HP-UX Reference

Release 11.0

User Commands

Section 1

Volume 1 of 5

Edition 1

HEWLETT® PACKARD

B2355-90166

E1097

Printed in: United States

Legal Notices

Printing History

The manual printing date and part number indicate its current edition. The printing date will change when a new edition is printed. Minor changes may be made at reprint without changing the printing date. the manual part number will change when extensive changes are made.

Manual updates may be issued between editions to correct errors or document product changes. To ensure that you receive the updated or new editions, you should subscribe to the appropriate product support service. See your HP sales representative for details.

First Edition: October 1997 (HP-UX Release 11.0)

Volume One
Table of Contents

Introduction

Section 1

Volume One
Table of Contents

Table of Contents
Volume One

Section 1: User Commands

Table of Contents
Volume One

Table of Contents
Volume One

Table of Contents
Volume One

Table of Contents
Volume One

Introduction to HP-UX Reference

Introduction to HP-UX Reference

NAME

Introduction - an introduction to the HP-UX operating system and the HP-UX Reference

INTRODUCTION

HP-UX is the Hewlett-Packard Company's implementation of an operating system that is compatible with various industry standards. It is based on the UNIX® System V Release 4 operating system and includes important features from the Fourth Berkeley Software Distribution.

Improvements include enhanced capabilities and other features, developed by HP to make HP-UX a very powerful, useful, and reliable operating system, capable of supporting a wide range of applications ranging from simple text processing to sophisticated engineering graphics and design. It can readily be used to control instruments and other peripheral devices. Real-time capabilities further expand the flexibility of HP-UX as a powerful tool for solving tough problems in design, manufacturing, business, and other areas where responsiveness and performance are important.

Extensive international language support enables HP-UX to interact with users in any of dozens of human languages. HP-UX interfaces easily with local area networks and resource-sharing facilities. By using industry-standard protocols, HP-UX provides flexible interaction with other computers and operating systems. Optional software products extend HP-UX capabilities into a broad range of specialized needs.

The *HP-UX Reference* is not a learning tool for beginners. It is primarily a reference tool that is most useful for experienced users of UNIX or UNIX-like systems. If you are not already familiar with UNIX or HP-UX, refer to the series of Beginner's Guides, tutorial manuals, and other learning documents supplied with your system or available separately. System implementation and maintenance details are explained in the *Managing Systems and Workgroups* manual.

MANPAGE ORGANIZATION

The contents of the *HP-UX Reference* and its on-line counterpart are a number of independent entries called **manpages**. These are also called **manual entries** or **reference pages**.

For convenient reference, the manpages are divided into eight specialized sections. The printed manual also has a table of contents for each volume and a composite index.

Each manpage consists of one or more printed pages, with the manpage name and section number printed in the upper corners. Manpages are arranged alphabetically within each section of the reference, except for the *intro* page at the beginning of each section. Manpages are referred to by name and section number, in the form *pagename*(*section*).

The manpages are available on-line through the **man** command if the manpages are present on the system. Refer to the *man*(1) manpage in Section 1 for more information.

Each page in the printed manual has two page numbers, printed at the bottom of the page. The center page number starts over with page 1 at the beginning of each new manpage; it is placed between two dashes in normal typeface. The number printed at the outside corner on each page sequences the printed pages within a section. Users usually locate manpages by the alphabetic headings at the top of the page as when reading a dictionary.

Some manpages describe two or more commands or routines. In such cases, the manpage is usually named for the first command or function that appears in the NAME section. Occasionally, a manpage name appears as a prefix to the NAME section. In such instances, the name describes the commands or functions in more general terms. For example, the *acct*(1M) manpage describes the **acctdisk**, **acctdusg**, **accton**, and **acctwtmp** commands, while the *string*(3C) manpage describes many character string functions.

The various sections are described as follows:

Volume Table of Contents (Printed Manual)

A complete listing of all manpages in the order they appear in each section, as well as alphabetically intermixed lists of all command, function, and feature names that are the different from the manpage where they appear

Section 1: User Commands

Programs that are usually invoked directly by users or from command language procedures (scripts).

Section 1M: System Administration Commands

Commands used for system installation and maintenance, including boot processes, crash recovery, system integrity testing, and other needs. Most commands in this section require the superuser privilege.

Section 2: System Calls

Entries into the HP-UX kernel, including the C-language interface. These topics are primarily of interest to programmers.

Section 3: Library Functions

Available subroutines that reside (in binary form) in various system libraries. These topics are primarily of interest to programmers.

Section 4: File Formats

The structure of various types of files, primarily of interest to administrators and programmers. For example, the link editor output file format is described in *a.out*(4). Files that are used only by a single command (such as intermediate files used by assemblers) are not described. C-language declarations corresponding to the formats in Section 4 can be found in the directories **`/usr/include`** and **`/usr/include/sys`**.

Section 5: Miscellaneous

A variety of information, such as descriptions of header files, character sets, macro packages, and other topics.

Section 7: Device Special Files

The characteristics of special (device) files that provide the link between HP-UX and system I/O devices. The names for each topic usually refer to the type of I/O device rather than to the names of individual special files.

Section 9: Introduction and Glossary

A general introduction (this one) and definitions of terms used in the HP-UX environment.

Composite Index (Printed Manual)

An alphabetical listing of keywords and topics based on the NAME section near the beginning of each manpage as well as other information, cross-referenced to manpage names and sections. The index also contains references to built-in features in the various command interpreters ("shells").

MANPAGE FORMATS

All manpages follow an established section heading format, but not all section headings are included in each manpage. A few manpages have self-explanatory specialized headings.

NAME

Gives the names of the commands, functions, or features and briefly states the purpose.

SYNOPSIS

Summarizes the syntax of the command or program entity. A few conventions are used:

`Constant-width` characters indicate literal characters that should be entered exactly as they appear. These characters appear in bold in the online manpages.

Italic strings represent variable elements that should be replaced with appropriate values.

Roman square brackets ([ ]) indicate that the contents are optional.

Roman braces ({ }) indicate a required element, usually in a choice.

Ellipses (...) indicate that the previous element can be repeated.

Note: An argument beginning with a dash (**`-`**), a plus sign (**`+`**), or an equal sign (**`=`**) is often defined as a command option, even if it appears in a position where a file name could appear. Therefore, it is unwise to have files names that begin with **`-`**, **`+`**, or **`=`**.

DESCRIPTION

Discusses the function and behavior of each entry.

EXTERNAL INFLUENCES

Information under this heading pertains to programming for various spoken languages. Typical entries indicate support for single- or multibyte characters, the effect of language-related environment variables on system behavior, and other related information.

NETWORKING FEATURES

Information under this heading is applicable only if you are using the network feature described there (such as NFS).

RETURN VALUE

Describes the values returned by function calls or in the return code by commands.

DIAGNOSTICS

Describes diagnostic information that may be produced. Self-explanatory messages are not listed.

ERRORS

Lists function error conditions (set in `errno`) and their corresponding error messages.

EXAMPLES

Provides examples of typical usage.

WARNINGS

Describes potential problems and deficiencies.

DEPENDENCIES

Describes variations in HP-UX operation that are related to the use of specific hardware or combinations of hardware.

AUTHOR

Indicates the origin of the software documented by the manpage. Unless noted otherwise, the source of an entry is System V.

FILES

Lists file names that are used or affected by the program or command.

SEE ALSO

Provides pointers to related manpages and other documentation.

STANDARDS CONFORMANCE

For each command or subroutine entry point addressed by one or more of the following industry standards, this section lists the standard specifications to which that HP-UX component conforms.

The various standards are:

AES	OSF Application Environment Specification
ANSI C	ANSI X3.159-1989
POSIX.1	IEEE Standard 1003.1-1988 (IEEE Computer Society) (Portable Operating System Interface for Computer Environments)
POSIX.2	IEEE Standard 1003.2-1990 (IEEE Computer Society) (Portable Operating System Interface for Computer Environments)
POSIX.4	IEEE Standard 1003.1b-1993 (IEEE Computer Society) (Portable Operating System Interface for Computer Environments)
FIPS 151-1	Federal Information Processing Standard 151-1 (National Institute of Standards and Technology)
FIPS 151-2	Federal Information Processing Standard 151-2 (National Institute of Standards and Technology)
SVID2	System V Interface Definition Issue 2
SVID3	System V Interface Definition Issue 3
XPG2	X/Open Portability Guide Issue 2 (X/Open, Ltd.)
XPG3	X/Open Portability Guide Issue 3 (X/Open, Ltd.)
XPG4	X/Open Portability Guide Issue 4 (X/Open, Ltd.)
XPG4.2	X/Open Portability Guide Issue 4 (X/Open, Ltd.) Version 2

GETTING STARTED WITH HP-UX

This is a very brief overview of how to use the HP-UX system: how to log in and log out, how to communicate through your machine, and how to run a program.

HP-UX uses **control characters** to perform certain functions. Control characters are generally shown in the form ^*x*, such as **^D** for Control-D. Hold down the **Control** (**Ctrl**) key while you press the character key.

Logging In

To log in you must have a valid user name and password, which can be obtained from your system administrator.

When a connection has been established, the system displays **login:** on your terminal. Type your user name and press the **Return** key. Enter your password (it is not echoed by the system) and press **Return**.

A list of copyright notices and a message-of-the-day may greet you before the first prompt.

It is important that you type your login name with lowercase letters, if possible. If you type uppercase letters, HP-UX assumes that your terminal cannot generate lowercase letters, and treats subsequent uppercase input as lowercase.

When you log in successfully, the system starts your login shell. The default is the POSIX shell, **/usr/bin/sh**. The POSIX shell (and its predecessors, the Korn and Bourne shells) use **$** as the default prompt. The C shell uses **%**.

See *login*(1) for more on login, *passwd*(1) to change your password, *chsh*(1) to change your login shell.

Logging Out

You can log out of the shells by typing an **exit** command or the **eof** (end-of-file) character (see the *Special Interactive Characters* subsection below). The shell terminates and the **login:** prompt appears again. (If you are using the C, Korn, or POSIX shells, respectively, see *csh*(1), *ksh*(1), or *sh-posix*(1) for information about the **ignoreeof** special command.)

How to Communicate Through Your Terminal

HP-UX gathers keyboard input characters and saves them in a buffer. The accumulated characters are not passed to the shell or other program until you type **Return**.

HP-UX terminal input/output is full-duplex. It has full read-ahead, which means that you can type at any time, even while a program is printing on your display or terminal. Of course, if you type during output, the output display will have the input characters interspersed in it. However, whatever you type will be saved and interpreted in the correct sequence. There is a limit to the amount of read-ahead, but it is generous and not likely to be exceeded unless the system is severely overloaded or operating abnormally. When the read-ahead limit is exceeded, the system throws away *all* the saved characters.

stty(1) tells you how to describe the characteristics of your terminal to the system. *profile*(4) explains how to accomplish this task automatically every time you log in.

Special Interactive Characters

A number of special characters are used to control the input and output of your terminal. These characters have defaults and can be redefined with the **stty** command (see *stty*(1)).

stty **Name**	**Default At Login** **Character (ASCII Name; Key Names)**	**Common** **Redefinition**
eof	**^D** (EOT)	
erase	**#**	**^H** (BS; Backspace)
kill	**@**	**^U** (NAK), **^X** (CAN)
intr	**^?** (DEL; Delete, Rub, Rubout)	**^C** (ETX)
quit	**^** (FS)	
start	**^Q** (DC1; X-ON)	
stop	**^S** (DC3; X-OFF)	

The **eof** character terminates "file" input from the terminal, as read by programs and scripts. By extension, **eof** can also terminate the shell (see the *Logging Out* subsection above).

The **kill** character deletes all characters typed before it on a terminal input line. The **erase** character erases the last character typed. Successive uses of **erase** will erase characters back to, but not beyond, the beginning of the input line.

The **intr** character generates an interrupt signal that bypasses the input buffer. This signal generally causes whatever program you are running to terminate. It can be used to stop a long printout that you don't want. However, programs can arrange either to ignore this signal altogether, or to be notified when it happens (instead of being terminated). For example, the **vi** editor catches interrupts and stops what it is doing, instead of terminating, so that an interrupt can be used to halt an editing operation without losing the file being edited.

The **quit** character generates a quit signal that bypasses the input buffer and most program traps and causes a running program to terminate. It can cause a core dump in the current directory.

The **stop** character can be used to pause output to the terminal. It is commonly used on video terminals to suspend output to the display while you read what is already being displayed. You can then resume output by typing the **start** character. When **stop** and **start** are used to suspend or resume output, they bypass the keyboard command-line buffer and are not passed to the program. However, any other characters typed on the keyboard are saved and used as input later in the program.

The **eof**, **erase**, and **kill** characters can be used as normal text characters if you escape them with a preceding \, as in **\\^D**. Therefore, to erase a \, you need two **erase**s.

The **intr**, **quit**, **start**, and **stop** characters cannot be escaped on the input line.

End-of-Line and Tab Characters

Besides adapting to the speed of the terminal, HP-UX tries to be intelligent as to whether you have a terminal with a **newline** (**line-feed**) key, or whether it must be simulated with a return/line-feed character pair. In the latter case, all incoming return characters are changed to line-feed characters (the standard line delimiter), and a return/line-feed pair is echoed to the terminal. If you get into the wrong mode, use the **stty** command to correct it (see *stty*(1)).

Tab characters are used freely in HP-UX source programs. If your terminal does not have the tab function, you can arrange to have tab characters changed into spaces during output, and echoed as spaces during input. The **stty** command sets or resets this mode. By default, the system assumes that tabs are set every eight character positions. The **tabs** command (see *tabs*(1)) can set tab stops on your terminal, if the terminal supports tabs.

How to Run a Program

When you have successfully logged into HP-UX, the shell monitors input from your terminal. The shell accepts typed lines from the terminal, splits them into command names and arguments, then executes the command. The command can be the name of a shell built-in, an executable script of commands, or an executable program. There is nothing special about system-provided commands, except that they are kept in directories where the shell can find them. You can also keep commands in your own directories and arrange for the shell to find them there.

The command name is the first word on an input line to the shell; the command and its arguments are separated from one another by blanks (one or more space and/or tab characters).

When a program terminates, the shell ordinarily regains control and prompts you to indicate that it is ready for another command. The shell has many other capabilities, which are described in detail in the appropriate manpages: *sh-posix*(1) for the POSIX shell, *ksh*(1) for the Korn shell, *sh-bourne*(1) for the Bourne shell, or *csh*(1) for the C shell.

The Current Directory

HP-UX has a file system arranged in a hierarchy of directories. When the system administrator gave you a user name, he or she also created a directory for you (ordinarily with the same name as your user name, and known as your *login* or *home* directory). When you log in, that directory becomes your *current* or *working* directory, and any file name you type is assumed to be in that directory by default. Because you are the owner of this directory, you have full permission to read, write, alter, or destroy its contents. The permissions you have for other directories and files will have been granted or denied to you by their respective owners, or by the system administrator. To change the current working directory use *cd*(1).

Path Names

To refer to files not in the current directory, you must use a path name. Full (absolute) path names begin with **/**, which is the name of the *root* directory of the whole file system. After the slash comes the name of each directory containing the next subdirectory (followed by a **/**), until finally the file name is reached (for example, **/usr/ae/filex** refers to file **filex** in directory **ae**, while **ae** is itself a subdirectory of **usr**; **usr** is a subdirectory of the root directory). See *glossary*(9) for a formal definition of **path name.**

If your current directory contains subdirectories, the path names of files in them begin with the name of the corresponding subdirectory (without a prefixed **/**). Generally, a path name can be used anywhere a file name is required.

Important commands that modify the contents of directories are **cp**, **mv**, and **rm** which respectively copy, move (that is, rename, relocate, or both), and remove files. To determine the status of files or the contents of directories, use the **ls** command. Use **mkdir** to make directories, **rmdir** to destroy them, and **mv** to rename them (see *cp*(1), *ls*(1), *mkdir*(1), *mv*(1), *rm*(1), and *rmdir*(1)).

Writing a Program

To enter the text of a source program into an HP-UX file, use a text editing program such as **vi**, **ex**, or **ed** (see *vi*(1), *ex*(1), and *ed*(1)). The three principal languages available under HP-UX are C (see *cc_bundled*(1) and *cc*(1)), FORTRAN (see *f77*(1)), and Pascal (see *pc*(1)). After the program text has been entered with the editor and written into a file (whose name has the appropriate suffix), you can give the name of that file to the appropriate language processor as an argument. Normally, the output of the language processor will be left in a file named **a.out** in the current directory. Since the results of a subsequent compilation may also be placed in **a.out**, thus overwriting the current output, you may want to use **mv** to give the output a unique name. If the program is written in assembly language, you will probably need to link library subroutines with it (see *ld*(1)). FORTRAN, C, and Pascal call the linker automatically.

When you have gone through this entire process without encountering any diagnostics, the resulting program can be run by giving its name to the shell in response to the prompt.

Your programs can receive arguments from the command line just as system programs do by using the *argc* and *argv* parameters. See the supplied C tutorial for details.

Text Processing

Almost all text is entered through a text editor. The editor preferred above all others provided with HP-UX is the **vi** editor. For batch-processing text files, the **sed** editor is very efficient. Other editors are used much less frequently. The **ex** editor is useful for handling certain situations while using **vi** but most other editors are rarely used except in various scripts.

The following editors are the same program masquerading under various names: **vi**, **view**, and **vedit** (see *vi*(1)) and **ex** and **edit** (see *ex*(1)). For information about the **sed** stream editor, see *sed*(1). The **ed** line editor is described in *ed*(1).

The commands most often used to display text on a terminal are **cat**, **more**, and **pr** (see *cat*(1), *more*(1), and *pr*(1)). The **cat** command simply copies ASCII text to the terminal, with no processing at all. The **more** command displays text on the terminal a screenful at a time, pausing for an acknowledgement from the user before continuing. The **pr** command paginates text, supplies headings, and has a facility for multicolumn output. **pr** is most commonly used in conjunction with the **lp** command (see *lp*(1)) to pipe formatted text to a line printer.

Interuser Communication

Certain commands provide interuser communication. Even if you do not plan to use them, it could be beneficial to learn about them, because someone else may direct them toward you. To communicate with another user that is currently logged in, you can use **write** to transfer text directly to that user's terminal display (if permission to do so has been granted by the other user). Otherwise, **elm**, **mailx**, or **mail** (in order of ease of use) can send a message to another user's mailbox. The user is then informed by HP-UX that mail has arrived (if currently logged in) or mail is present (when the user next logs in). Refer to *elm*(1), *mail*(1), *mailx*(1), and *write*(1) for explanations of how these commands are used.

ACKNOWLEDGEMENTS

UNIX is a registered trademark of The Open Group.

SEE ALSO

cat(1), cc_bundled(1), cd(1), chsh(1), cp(1), csh(1), ed(1), ex(1), f77(1), ksh(1), ld(1), login(1), lp(1), ls(1), mail(1), mailx(1), man(1), mkdir(1), more(1), mv(1), passwd(1), pc(1), pr(1), rm(1), rmdir(1), sed(1), sh(1), sh-bourne(1), sh-posix(1), stty(1), tabs(1), vi(1), write(1), a.out(4), profile(4), glossary(9).

Section 1

User Commands

Section 1

User Commands

NAME

intro - introduction to command utilities and application programs

DESCRIPTION

This section describes commands accessible by users, as opposed to system calls in Section (2) or library routines in Section (3), which are accessible by user programs.

Command Syntax

Unless otherwise noted, commands described in this section accept options and other arguments according to the following syntax:

name [*option* (*s*)] [*cmd_arg* (*s*)]

where the elements are defined as follows:

name Name of an executable file.

option One or more *option*s can appear on a command line. Each takes one of the following forms:

- *no_arg_letter*
A single letter representing an option without an argument.

- *no_arg_letters*
Two or more single-letter options combined into a single command-line argument.

- *arg_letter<>opt_arg*
A single-letter option followed by a required argument where:
arg_letter
is the single letter representing an option that requires an argument,
opt_arg
is an argument (character string) satisfying the preceding *arg_letter*,
<> represents optional white space.

cmd_arg Path name (or other command argument) *not* beginning with **-**, or **-** by itself indicating the standard input. If two or more *cmd_arg*s appear, they must be separated by white space.

Manual Entry Formats

All manual entries follow an established topic format, but not all topics are included in each entry.

NAME Gives the name(s) of the entry and briefly states its purpose.

SYNOPSIS Summarizes the use of the entry or program entity being described. A few conventions are used:

`Computer font` strings are literals, and are to be typed exactly as they appear in the manual (except for parameters in the SYNOPSIS section of entries in Sections 2 and 3).

Italic strings represent substitutable argument names and names of manual entries found elsewhere in the manual.

Square brackets [] around an argument name indicate that the argument is optional.

Ellipses (...) are used to show that the previous argument can be repeated.

A final convention is used by the commands themselves. An argument beginning with a dash (-), a plus sign (+), or an equal sign (=) is often taken to be some sort of option argument, even if it appears in a postion where a file name could appear. Therefore it is unwise to have file names that begin with -, +, or =.

DESCRIPTION Discusses the function and behavior of each entry.

EXTERNAL INFLUENCES
Information under this heading pertains to programming for various spoken languages. Typical entries indicate support for single- and/or multi-byte characters, the effect of language-related environment variables on system behavior, and other related information.

NETWORKING FEATURES	Information under this heading is applicable only if you are using the networking feature described there (such as NFS).
RETURN VALUE	Discusses various values returned upon completion of program calls.
DIAGNOSTICS	Discusses diagnostics indications that may be produced. Self-explanatory messages are not listed.
ERRORS	Lists error conditions and their corresponding error message or return value.
EXAMPLES	Provides examples of typical usage, where appropriate.
WARNINGS	Points out potential pitfalls.
DEPENDENCIES	Points out variations in HP-UX operation that are related to the user or specific hardware or hardware combinations.
AUTHOR	Indicate the origin of the software documented by the manual entry.
FILES	Lists file names that are built into the program or command.
SEE ALSO	Provides pointers to related topics.
BUGS	Discusses known bugs and deficiencies, occasionally suggesting fixes.
STANDARDS CONFORMANCE	This section lists the standard specifications to which the HP-UX component conforms.

RETURN VALUE

Upon termination, each command returns two bytes of status, one supplied by the system giving the cause for termination, and (in the case of "normal" termination) one supplied by the program (for descriptions, see *wait*(2) and *exit*(2)). The system-supplied byte is 0 for normal termination. The byte provided by the program is customarily 0 for successful execution and non-zero to indicate errors or failure such as incorrect parameters in the command line, or bad or inaccessible data. Values returned are usually called variously "exit code", "exit status", "return code", or "return value", and are described only where special conventions are involved.

WARNINGS

Some commands produce unexpected results when processing files containing null characters. These commands often treat text input lines as strings, and therefore become confused when they encounter a null character (the string terminator) within a line.

SEE ALSO

getopt(1), exit(2), wait(2), getopt(3C), hier(5), Introduction(9).

NAME

adb, adb64 - absolute debugger

SYNOPSIS

adb [**-w**] [**-I***dir*] [**-k**] [**-m**] [**-P***pid*] *objfil* [*corfil*]

adb64 [**-w**] [**-I***dir*] [**-k**] [**-m**] [**-P***pid*] *objfil* [*corfil*]

DESCRIPTION

The **adb** command executes a general-purpose debugging program that is sensitive to the underlying architecture of the processor and operating system on which it runs. It can be used to examine files and provide a controlled environment for executing HP-UX programs.

adb calls **adb64** to process 64 bit files.

objfil is normally an executable program file, or an HP-UX kernel (vmunix), preferably containing a symbol table; if not, the symbolic features of **adb** cannot be used, although the file can still be examined. The default for *objfil* is **a.out**.

corfil is assumed to be a core image file produced after executing *objfil* or an HP-UX crash file produced from the *objfil*. The default for *corfil* is **core**.

Requests to **adb** are read from standard input and **adb** responds on standard output. If the **-w** flag is present, *objfil* is created (if necessary) and opened for reading and writing, to be modified using **adb**. The **-I** option specifies a directory where files read with **$<** or **$<<** (see below) are sought; the default is **/usr/lib/adb**. **adb** ignores QUIT; INTERRUPT causes return to the next **adb** command.

The following options are also supported:

-k Allows **adb** to read *objfil* as an HP-UX kernal file and *corfil* as an HP-UX crash dump. This also allows virtual-to-physical address translation, useful for kernel debugging. In this case, *corfil* should be an HP-UX crash dump or **/dev/mem**. Without **-k** or **-m**, **adb** treats *objfil* as an application program file and *corfil* as an application core file.

When **adb** is invoked with this option, it sets up the context of the currently running process using space registers four through seven. A user specified address is dereferenced by combining it with the appropriate space register, depending on the quadrant in which the 32-bit address lies.

When the current radix is not (decimal) ten, the **-k** option allows **adb** to support the notion of long pointers or addresses in the form *space***.***offset*. Once a space is specified, all subsequent addresses are dereferenced using that space until the user enters another long address. If a space equal to (hexadecimal) 0xffffffff is used, **adb** reverts to the previous context and uses space registers four through seven to dereference 32-bit addresses.

-m Must be specified instead of **-k** when a core dump is written to multiple files by **savecore**.

When **-m** is used, *corfil* must be specified as the path name of the directory that contains system core dump files. For example, **savecore** normally saves system core dump files in a directory named **/var/adm/crash/core.***n*. (The trailing *n* in the path name is a number that increases by one every time **savecore** is run with the same *dirname*. See *savecore*(1M) for more information.)

Therefore, a command line using **-m** might look similar to the following:

```
adb -m /var/adm/crash/core.1/vmunix /var/adm/crash/core.1
```

Notice that when **-m** is specified on the command line, **-k** is not necessary.

-P*pid* Causes **adb** to adopt process *pid* as a "traced" process (see *ptrace*(2)). This option is helpful for debugging processes that were not originally run under the control of **adb**.

Requests to **adb** follow the form:

[*address*] [**,** *count*] [*command*] [**;**]

If *address* is present, *dot* is set to *address*. Initially *dot* is set to **0**. For most commands, *count* specifies the number of times the command is to be executed. The default *count* is **1**. *address* and *count* are expressions.

The interpretation of an address depends on the context in which it is used. If a subprocess is being debugged, addresses are interpreted in the address space of the subprocess. (For further details of address mapping see Addresses below.)

Expressions

Expressions are interpreted as follows:

.	The value of *dot*.
+	The value of *dot* increased by the current increment.
^	The value of *dot* decreased by the current decrement.
"	The last *address* typed.
integer	A number. The prefix **0** (zero) forces interpretation in octal radix; the prefixes **0d** and **0D** force interpretation in decimal radix; the prefixes **0x** and **0X** force interpretation in hexadecimal radix. Thus **020** = **0d16** = **0x10** = sixteen. If no prefix appears, the *default radix* is used; see the **$d** command. The radix is initialized to the base used in the assembly language for the processor involved. Note that a hexadecimal number whose most significant digit would otherwise be an alphabetic character must have a **0x** (or **0X**) prefix.
integer **.** *fraction*	A 32-bit floating-point number.
'*cccc***'**	The ASCII value of up to 4 characters. A backslash (\) can be used to escape a single quote (**'**).
< *name*	*name* can have the value of either a variable or a register. **adb** maintains a number of variables named by single letters or digits; see Variables below. If *name* is a register, the value of the register is obtained from the CORE_PROC segment in *corfil* (before the subprocess is initiated) or from the user area of the subprocess. Register names are implementation dependent; see the **$r** command.
symbol	A *symbol* is a sequence of uppercase or lowercase letters, underscores, or digits, not starting with a digit. A backslash (\) can be used to escape other characters. The value of the *symbol* is taken from the symbol table in *objfil*. An initial underscore (_) is prefixed to *symbol*, if needed.
_ *symbol*	If the compiler prefixes _ to an external symbol, it may be necessary to cite this name to distinguish it from a symbol generated in assembly language.
(*exp*)	The value of the expression *exp*.

The following are monadic operators:

*****exp*	The contents of the location addressed by *exp* in *corfil*.
@ *exp*	The contents of the location addressed by *exp* in *objfil*.
-*exp*	Integer negation.
~*exp*	Bitwise complement.

The following dyadic operators are left associative and are less binding than monadic operators:

*e1***+***e2*	Integer addition.
*e1***-***e2*	Integer subtraction.
*e1*******e2*	Integer multiplication.
*e1***%***e2*	Integer division.
*e1***&***e2*	Bitwise conjunction.
*e1***|***e2*	Bitwise disjunction.
*e1***#***e2*	*e1* rounded up to the next multiple of *e2*.

Commands

Most commands consist of an action character followed by a modifier or list of modifiers. The following action characters can take format specifiers. (The action characters **?** and **/** can be followed by *****; see Addresses for further details.)

?*f*	Locations starting at *address* in *objfil* are printed according to the format *f*. *dot* is incremented by the sum of the increments for each format letter. If a subprocess has been initiated, *address* references a location in the address space of the subprocess instead of *objfil*.
/*f*	Locations starting at *address* in *corfil* are printed according to the format *f* and *dot* is increased like **?**. If a subprocess has been initiated, *address* refers to a location in the address space of the subprocess instead of *corfil*.
=*f*	The value of *address* is printed in the styles indicated by the format *f*. (For **i** format **?** is printed for the parts of the instruction that refer to subsequent words.)

A *format* consists of one or more characters that specify a style of printing. Each format character can be preceded by an integer that indicates how many times the format is repeated. While stepping through a format, *dot* is increased by the amount given for each format character. If no format is given then the last format is used.

The following format characters are available:

o 2	Print 2 bytes in octal. All octal numbers output by **adb** are preceded by **0**.
O 4	Print 4 bytes in octal.
q 2	Print 2 bytes in signed octal.
Q 4	Print 4 bytes in signed octal.
d 2	Print 2 bytes in decimal.
D 4	Print 4 bytes in decimal.
x 2	Print 2 bytes in hexadecimal.
X 4	Print 4 bytes in hexadecimal.
u 2	Print 2 bytes as an unsigned decimal number.
U 4	Print 4 bytes as an unsigned decimal number.
f 4	Print the 32 bit value as a floating point number.
F 8	Print double floating point.
b 1	Print the addressed byte in hexadecimal.
B 1	Print the addressed byte in octal.
c 1	Print the addressed character (the sign bit is ignored).
C 1	Print the addressed character using the following escape convention. First, the sign bit is discarded, then character values **000** to **040** are printed as **@** followed by the corresponding character in the range **0100** to **0140**. The character **@** is printed as **@@**.
s *n*	Print the addressed characters until a zero character is reached.
S *n*	Print a string using the **@** escape convention. The value *n* is the length of the string including its zero terminator.
Y 4	Print 4 bytes in date format (see *ctime*(3C)).
i *n*	Print as machine instructions. The value of *n* is the number of bytes occupied by the instruction.
a 0	Print the value of *dot* in symbolic form.
p *n*	Print the addressed value in symbolic form. The value of *n* is a machine-dependent constant.
t 0	When preceded by an integer, moves to the next appropriate tab stop. For example, **8t** moves to the next 8-space tab stop.
r 0	Print a space.
n 0	Print a new-line character.
"..." 0	Print the enclosed string.

^ *dot* is decreased by the current increment. Nothing is printed.

+ *dot* is increased by 1. Nothing is printed.

- *dot* is decreased by 1. Nothing is printed.

new-line
Repeat the previous command with a *count* of 1. The value of *dot* continues from the end of the previous format, unlike a [**?/**] command with no *address*, which repeats the previous *address* value. New-line can also be used to repeat a **:s** or **:c** command; however, any arguments to the previous command are lost.

[**?/**]**l** *value mask*
Words starting at *dot* are masked with *mask* and compared with *value* until a match is found. If **L** is used, **adb** looks to match 4 bytes at a time instead of 2. If no match is found, *dot* is left unchanged; otherwise *dot* is set to the matched location. If *mask* is omitted **-1** is used.

[**?/**]**w** *value ...*
Write the 2-byte *value* into the addressed location. If the command is **W**, write 4 bytes. Odd addresses are not allowed when writing to the subprocess address space.

=m Toggle the address mapping of *corfil* between the initial map set up for a valid core file and the default mapping pair which the user can modify with **/m**. If the *corfil* was invalid, only the default mapping is available.

[**?/**]**m** *b1 e1 f1*[**?/**]
Record new values for (*b1*, *e1*, *f1*). If fewer than three expressions are given, the remaining map parameters are left unchanged. If the **?** or **/** is followed by *****, the second segment (*b2*, *e2*, *f2*) of the mapping is changed. If the list is terminated by **?** or **/**, the file (*objfil* or *corfil*, respectively) is used for subsequent requests. (For example, **/m?** causes **/** to refer to *objfil*.) A **/m** command switches the *corfil* mapping to the default mapping pair. For a valid core file, the **=m** command can be used to switch back to the initial mapping.

>*name*
Assign *dot* to the variable or register named.

! Call a shell to read the remainder of the line following **!**.

The following **$** commands take the form **$***modifier*:

$<*f* Read commands from the file *f*. If this command is executed in a file, further commands in the file are not seen. If a *count* is given, and is zero, the command is ignored. The value of the count is placed in variable 9 before the first command in *f* is executed.

$<<*f* Similar to **$<** except it can be used in a file of commands without causing the file to be closed. Variable 9 is saved when the command executes and is restored when it completes. Only five **$<<** files can be open at once.

$>*f* Send output to the file *f*, which is created if it does not already exist.

$r Print the general registers and the instruction addressed by the process counter. *dot* is set to the process counter contents.

$f Print the floating-point registers.

$b Print all breakpoints and their associated counts and commands.

$c C stack backtrace. If *address* is given, it is taken as the address of the current frame (instead of the normal stack frame pointer). If *count* is given, only the first *count* frames are printed.

$e The names and values of external variables are printed.

$w Set the page width for output to *address* (default 80).

$s Set the limit for symbol matches to *address*. The default is system dependent.

$o The default for all integers input is octal.

$d Set the default radix to *address* and report the new value. Note that *address* is interpreted in the (old) current radix. Thus **10$d** never changes the default radix. To make decimal the default radix, use **0d10$d**.

$x	The default for all integers input is hexadecimal.
$q	Exit from **adb**.
$v	Print all non-zero variables in the current radix.
$m	Print the address map. This includes both the initial and default maps for a valid *corfil* with an indication of which is currently active.
$new-line	Print the process id and register values.
$z	Print a list of signals and how they are handled. See **:z** for information on changing signal handling.

The available **:** commands manage subprocesses, and take the form **:***modifier*:

:b*c*	Set breakpoint at *address*. The breakpoint is executed *count*−1 times before causing a stop. Each time the breakpoint is encountered, the command *c* is executed. If this command sets *dot* to zero, the breakpoint causes a stop.
:d	Delete breakpoint at *address*. **:d*** deletes all breakpoints.
:r	Run *objfil* as a subprocess. If *address* is given explicitly, the program is entered at this point; otherwise the program is entered at its standard entry point. The value *count* specifies how many breakpoints are ignored before stopping. Arguments to the subprocess may be supplied on the same line as the command. An argument starting with **<** or **>** causes the standard input or output to be established for the command. All signals are turned on when entering the subprocess.
:e	Set up a subprocess as in **:r**; no instructions are executed.
:c*s*	Continue the subprocess with signal *s* (see *signal*(5)). If *address* is given, the subprocess continues at this address. If no signal is specified, the signal that caused the subprocess to stop is sent. Breakpoint skipping is the same as for **:r**.
:s*s*	As for **c** except that the subprocess is single stepped *count* times. If there is no current subprocess, *objfil* is run as a subprocess as for **:r**. In this case no signal can be sent; the remainder of the line is treated as arguments to the subprocess.
:S*s*	Same as **:c** except that a temporary breakpoint is set at the next instruction. Useful for stepping across subroutines.
:x *a* [*b*]...	Execute subroutine *a* with parameters [*b*]...
:k	Terminate the current subprocess, if any.
:z*d*	Change signal handling for a specified signal. Disposition *d* can be specified as:

+s	Stop process when signal is received.
-s	Do not stop process when signal is received.
+r	Report when signal is received.
-r	Do not report when signal is received.
+d	Deliver signal to the target process.
-d	Do not deliver signal to the target process.

For example, **0x10:z+d** enables delivering of signal number **0x10** to the target process. Use **$z** to display existing settings.

Variables

adb provides named and numbered variables. Named variables are set initially by **adb** but are not used subsequently. Numbered variables are reserved for communication as follows:

0	The last value printed.
1	The last offset part of an instruction source.
2	The previous value of variable 1.
9	The count on the last **$<** command.

On entry, the following named variables are set from the coreheaders in the *corfil*. If *corfil* does not appear to be a **core** file, these values are set from *objfil*.

b	The base address of the data segment.
d	The data segment size.
s	The stack segment size.
t	The text segment size.

The following variables are set from *objfil*.

e	The entry point.
m	The "magic" number as defined in **<magic.h>**.

Addresses

The file address associated with a written address is determined by a mapping described below; see **$m**. Both the *objfil* mapping and the default *corfil* mapping are represented by two triples (*b1, e1, f1*) and (*b2, e2, f2*). The initial mapping for a valid *corfil* contains a triple for each segment (coreheader).

The *file address* corresponding to a written *address* is calculated as follows:

If

$b1 \leq address < e1$, *file address* $= address + f1-b1$,

otherwise, if

$b2 \leq address < e2$, *file address* $= address + f2-b2$,

otherwise, the requested *address* is not valid. For a valid *corfil*, this pattern repeats as many times as there are segments (coreheaders) in the *corfil*, rather than twice. If **?** or **/** is followed by *****, only the second triple is used, or (when using the initial mapping of a valid *corfil*) only segments with a **CORE_STACK** coreheader.

The initial setting of both mappings is suitable for normal **a.out** and **core** files. If either file is not of the kind expected, **adb** sets *b1* to **0**, *e1* to the maximum file size, and *f1* to **0**; in this way the entire file can be examined with no address translation.

adb keeps all appropriate values as signed 32-bit integers so that it can be used on large files.

EXTERNAL INFLUENCES

International Code Set Support

Single- and multi-byte character code sets are supported.

RETURN VALUE

adb comments about inaccessible files, syntax errors, abnormal termination of commands, etc. It echoes **adb** when there is no current command or format. Exit status is **0**, unless the last command failed or returned non-zero status.

DEPENDENCIES

- Setting breakpoints in shared libraries is not supported.

  adb does not read the linker symbol table for shared libraries, and cannot access locations in shared libraries by name. In a stack backtrace (**$c**), **adb** does not know the names of shared library procedures.

  If the core file was created when the program was in a shared library function, the **$c** command does not work. When a stack backtrace for the core file encounters a shared library procedure on the stack it aborts at that point.

- A leading zero by itself is not recognized as a radix indicator. Use the prefixes **0o** or **0O** (zero-oh) to force interpretation in octal radix. The prefixes **0t** and **0T** are also accepted to force interpretation in decimal radix. Thus **0o20** = **0t16** = sixteen. A hexadecimal number whose most significant digit would otherwise be an alphabetic character may begin with a leading zero instead of **0x** (or **0X**), if the default radix is hexadecimal.

- The **$f** command prints floating point registers as 32-bit single precision and **$F** prints these registers as 64-bit doubles.

- **$R** prints all registers available to **adb** users.
- The **:x** and **:S** commands are not currently supported.
- **adb** can be used to inspect relocatable object files; it reads the symbol table and sets up the appropriate mappings for text and data. Note that relocatable object files do not necessarily contain an exact image of the initialized data; however, if this is the case, the data mapping is not set.

AUTHOR

adb was developed by AT&T and HP.

FILES

a.out
core
/dev/mem
/dev/kmem
/dev/swap

SEE ALSO

savecore(1M), ptrace(2), crt0(3), ctime(3C), end(3C), a.out(4), core(4), savecore(4), signal(5).

ADB Tutorial

NAME

adjust - simple text formatter

SYNOPSIS

adjust [**-b**] [**-c**|**-j**|**-r**] [**-m** *column*] [**-t** *tabsize*] [*files* ...]

DESCRIPTION

The **adjust** command is a simple text formatter for filling, centering, left and right justifying, or only right justifying text paragraphs, and is designed for interactive use. It reads the concatenation of input files (or standard input if none are given) and produces on standard output a formatted version of its input, with each paragraph formatted separately. If **-** is given as an input filename, **adjust** reads standard input at that point (use **- -** as an argument to separate **-** from options.)

adjust reads text from input lines as a series of words separated by space characters, tabs, or newlines. Text lines are grouped into paragraphs separated by blank lines. By default, text is copied directly to the output, subject only to simple filling (see below) with a right margin of 72, and leading spaces are converted to tabs where possible.

Options

The **adjust** command recognizes the following command-line options:

-b Do not convert leading space characters to tabs on output; (output contains no tabs, even if there were tabs in input).

-c Center text on each line. Lines are pre- and post-processed, but no filling is performed.

-j Justify text. After filling, insert spaces in each line as needed to right justify it (except in the last line of each paragraph) while keeping the justified left margin.

-r After filling text, adjust the indentation of each line for a smooth right margin (ragged left margin).

-m*column*

Set the right fill margin to the given column number, instead of 72. Text is filled, and optionally right justified, so that no output line extends beyond this column (if possible). If **-m0** is given, the current right margin of the first line of each paragraph is used for that and all subsequent lines in the paragraph.

By default, text is centered on column 40. With **-c**, the **-m** option sets the middle column of the centering “window”, but **-m0** auto-sets the right side as before (which then determines the center of the “window”).

-t*tabsize* Set the tab size to other than the default (eight columns).

Only one of the **-c**, **-j**, and **-r** options is allowed in a single command line.

Details

Before doing anything else to a line of input text, **adjust** first handles backspaces, rubbing out preceding characters in the usual way. Next, it ignores all non-printable characters except tab. It then expands all tabs to spaces.

For simple text filling, the first word of the first line of each paragraph is indented the same amount as in the input line. Each word is then carried to the output followed by one space. “Words” ending in *terminal_character*[*quote*][*closing_character*] are followed by two spaces, where *terminal_character* is any of **.**, **:**, **?**, or **!**; *quote* is a single closing quote (**'**) character or double-quote character ("), and *close* is any of **)**, **]**, or **}**. Here are some examples:

end. of? sentence.' sorts!" of.) words?"]

(**adjust** does not place two spaces after a pair of single closing quotes (**''**) following a *terminal_character*).

adjust starts a new output line whenever adding a word (other than the first one) to the current line would exceed the right margin.

adjust understands indented first lines of paragraphs (such as this one) when filling. The second and subsequent lines of each paragraph are indented the same amount as the second line of the input paragraph if there is a second line, else the same as the first line.

- `*` **adjust** also has a rudimentary understanding of tagged paragraphs (such as this one) when filling. If the second line of a paragraph is indented more than the first, and the first line has a word beginning at the same indentation as the second line, the input column position of the tag word or words (prior to the one matching the second line indentation) is preserved.

Tag words are passed through without change of column position, even if they extend beyond the right margin. The rest of the line is filled or right justified from the position of the first non-tag word.

When **-j** is given, **adjust** uses an intelligent algorithm to insert spaces in output lines where they are most needed, until the lines extend to the right margin. First, all one space word separators are examined. One space is added to each separator, starting with the one having the most letters between it and the preceding and following separators, until the modified line reaches the right margin. If all one space separators are increased to two spaces and more spaces must be inserted, the algorithm is repeated with two space separators, and so on.

Output line indentation is held to one less than the right margin. If a single word is larger than the line size (right margin minus indentation), that word appears on a line by itself, properly indented, and extends beyond the right margin. However, if **-r** is used, such words are still right justified, if possible.

If the current locale defines class names **ekinsoku** and **bkinsoku** (see *iswctype*(3C)), **adjust** formats the text in accordance with the **ekinsoku**/**bkinsoku** character classification and margin settings (see **-r**, **-j**, and **-m** options).

EXTERNAL INFLUENCES

Environment Variables

LANG provides a default value for the internationalization variables that are unset or null. If **LANG** is unset or null, the default value of "C" (see *lang*(5)) is used. If any of the internationalization variables contains an invalid setting, **adjust** will behave as if all internationalization variables are set to "C". See *environ*(5).

LC_ALL If set to a non-empty string value, overrides the values of all the other internationalization variables.

LC_CTYPE determines the interpretation of text as single and/or multi-byte characters, the classification of characters as printable, and the characters matched by character class expressions in regular expressions.

LC_MESSAGES determines the locale that should be used to affect the format and contents of diagnostic messages written to standard error and informative messages written to standard output.

NLSPATH determines the location of message catalogs for the processing of **LC_MESSAGES**.

International Code Set Support

Single- and multi-byte character code sets are supported.

DIAGNOSTICS

adjust complains to standard error and later returns a nonzero value if any input file cannot be opened (it skips the file). It does the same (but quits immediately) if the argument to **-m** or **-t** is out of range, or if the program is improperly invoked.

Input lines longer than **BUFSIZ** are silently split (before tab expansion) or truncated (afterwards). Lines that are too wide to center begin in column 1 (no leading spaces).

EXAMPLES

This command is useful for filtering text while in *vi*(1). For example,

```
!}adjust
```

reformats the rest of the current paragraph (from the current line down), evening the lines.

The **vi** command:

```
:map ^X {!}adjust -j^V^M
```

(where **^** denotes control characters) sets up a useful "finger macro". Typing **^X** (Ctrl-X) reformats the entire current paragraph.

adjust -m1 is a simple way to break text into separate words without white space, except for tagged-paragraph tags.

WARNINGS

This program is designed to be simple and fast. It does not recognize backslash to escape white space or other characters. It does not recognize tagged paragraphs where the tag is on a line by itself. It knows that lines end in newline or null, and how to deal with tabs and backspaces, but it does not do anything special with other characters such as form feed (they are simply ignored). For complex operations, standard text processors are likely to be more appropriate.

This program could be implemented instead as a set of independent programs, fill, center, and justify (with the **-r** option). However, this would be much less efficient in actual use, especially given the program's special knowledge of tagged paragraphs and last lines of paragraphs.

AUTHOR

adjust was developed by HP.

SEE ALSO

nroff(1).

NAME

admin - create and administer SCCS files

SYNOPSIS

admin -i[*name*] [**-n**] [**-b**] [**-a** *login*] ... [**-d** *flag*[*flag-val*]] ... [**-f** *flag*[*flag-val*]] ...
[**-m** *mrlist*] ... [**-r** *rel*] [**-t**[*name*]] [**-y**[*comment*]] *file* ...

admin -n [**-a** *login*] ... [**-d** *flag*[*flag-val*]] ... [**-f** *flag*[*flag-val*]] ... [**-m** *mrlist*] ...
[**-t**[*name*]] [**-y**[*comment*]] *file* ...

admin [**-a** *login*] ... [**-e** *login*] ... [**-d** *flag*[*flag-val*]] ... [**-m** *mrlist*] ...
[**-r** *rel*] [**-t**[*name*]] *file* ...

admin -h *file* ...

admin -z *file* ...

DESCRIPTION

The **admin** command is used to create new SCCS files and change the parameters of existing ones. Arguments to **admin**, which may appear in any order, (unless **--** is specified as an argument, in which case all arguments after **--** are treated as files) consist of option arguments, beginning with **-**, and named *file*s (note that SCCS file names must begin with the characters **s.**). If a named *file* does not exist, it is created and its parameters are initialized according to the specified option arguments. Parameters not initialized by an option argument are assigned a default value. If a named *file* does exist, parameters corresponding to specified option arguments are changed, and other parameters are left unaltered.

If *directory* is named instead of *file*, **admin** acts on each *file* in *directory*, except that non-SCCS files (the last component of the path name does not begin with **s.**) and unreadable files are silently ignored. If a name of **-** is given, the standard input is read, and each line of the standard input is assumed to be the name of an SCCS file to be processed. Again, non-SCCS files and unreadable files are silently ignored.

The **admin** option arguments apply independently to all named *file*s, whether one file or many. In the following discussion, each option is explained as if only one file is specified, although they affect single or multiple files identically.

Options

The **admin** command supports the following options and command-line arguments:

-n This option indicates that a new SCCS file is to be created.

-i[*name*] The *name* of a file from which the contents for a new SCCS file is to be taken. (if name is a binary file, then you must specify the -b option) The contents constitutes the first delta of the file (see the **-r** option for the delta numbering scheme). If the **-i** option is used but the file name is omitted, the text is obtained by reading the standard input until an end-of-file is encountered. If this option is omitted, the SCCS file is created with an empty initial delta. Only one SCCS file can be created by an **admin** command on which the **-i** option is supplied. Using a single **admin** to create two or more SCCS files requires that they be created empty (no **-i** option). Note that the **-i** option implies the **-n** option.

-b Encode the contents of name, specified to the **-i** option. This keyletter must be used if name is a binary file; otherwise, a binary file will not be handled properly by SCCS commands.

-r *rel* The release (*rel*) into which the initial delta is inserted. This option can be used only if the **-i** option is also used. If the **-r** option is not used, the initial delta is inserted into release 1. The level of the initial delta is always 1 (by default initial deltas are named 1.1).

-t[*name*] The *name* of a file from which descriptive text for the SCCS file is to be taken. If the **-t** option is used and **admin** is creating a new SCCS file (the **-n** and/or **-i** options are also used), the descriptive text file name must also be supplied. In the case of existing SCCS files:

- A **-t** option without a file name causes removal of descriptive text (if any) currently in the SCCS file.
- A **-t** option with a file name causes text (if any) in the named file to replace the descriptive text (if any) currently in the SCCS file.

-f *flag* This option specifies a *flag*, and possibly a value for the *flag*, to be placed in the SCCS file. Several **-f** options can be supplied on a single **admin** command line. The allowable *flag*s and their values are:

b	Allows use of the **-b** option on a **get** command (see *get*(1)) to create branch deltas.
c*ceil*	The highest release (i.e., "ceiling"), a number less than or equal to 9999, which can be retrieved by a **get** command for editing. The default value for an unspecified **c** flag is 9999.
f*floor*	The lowest release (i.e., "floor"), a number greater than 0 but less than 9999, which may be retrieved by a **get** command for editing. The default value for an unspecified **f** flag is 1.
d*SID*	The default delta number *SID* to be used by a **get** command (see *get*(1)).
i*str*	Causes the message: **No id keywords (cm7)** issued by **get** or **delta** to be treated as a fatal error (see *delta*(1)). In the absence of this flag, the message is only a warning. The message is issued if no SCCS identification keywords (see *get*(1)) are found in the text retrieved or stored in the SCCS file. If a value is supplied, the keywords must exactly match the given string. However, the string must contain a keyword, but must not contain embedded newlines.
j	Allows concurrent **get** commands for editing on the same *SID* of an SCCS file. This allows multiple concurrent updates to the same version of the SCCS file. Only one user can perform concurrent edits. Access by multiple users is usually accomplished by using a common login or a set user ID program (see *chmod*(1) and *exec*(2)).
l*list*	A *list* of releases to which deltas can no longer be made. (A **get -e** against one of these locked releases fails). The *list* has the following syntax: *list* **::=** *range* | *list* **,** *range* ; *range* **::=** *RELEASE NUMBER* **| a** . The character **a** in the *list* is equivalent to specifying *all releases* for the named SCCS file. Omitting any list is equivalent to **a**.
n	Causes **delta** to create a null delta in each of those releases being skipped (if any) when a delta is made in a *new* release (such as when making delta 5.1 after delta 2.7, release 3 and release 4 are skipped). These null deltas serve as **anchor points** so that branch deltas can be created from them later. The absence of this flag causes skipped releases to be nonexistent in the SCCS file, preventing branch deltas from being created from them in the future.
q*text*	User-definable text substituted for all occurrences of the **%Q%** keyword in SCCS file text retrieved by **get**.
m*mod*	The *module* name of the SCCS file substituted for all occurrences of the **%M%** keyword in SCCS file text retrieved by **get**. If the **m** flag is not specified, the value assigned is the name of the SCCS file with the leading **s.** removed.
t*type*	The *type* of module in the SCCS file substituted for all occurrences of **%Y%** keyword in SCCS file text retrieved by **get**.

v[*pgm*] Causes **delta** to prompt for Modification Request (*MR*) numbers as the reason for creating a delta. The optional value specifies the name of a (*MR*) number validity checking program (see *delta*(1)). (If this flag is set when creating an SCCS file, the **m** option must also be used even if its value is null).

x Causes **get** to create files with execute permissions.

-d *flag* Causes removal (deletion) of the specified *flag* from an SCCS file. The **-d** option can be specified only when processing existing SCCS files. Several **-d** options can be supplied on a single **admin** command line. See the **-f** option for allowable *flag* names.

l*list* A *list* of releases to be unlocked. See the **-f** option for a description of the **l** flag and the syntax of a *list*.

-a *login* A *login* name, or numerical HP-UX group ID, to be added to the list of users allowed to make deltas (changes) to the SCCS file. A group ID is equivalent to specifying all *login* names common to that group ID. Several **a** options can be used on a single **admin** command line. As many *login*s or numerical group IDs as desired can be on the list simultaneously. If the list of users is empty, anyone can add deltas. A *login* or group ID preceded by a **!** denies permission to make deltas.

-e *login* A *login* name or numerical group ID to be erased from the list of users allowed to make deltas (changes) to the SCCS file. Specifying a group ID is equivalent to specifying all *login* names common to that group ID. Several **e** options can be used on a single **admin** command line.

-y[*comment*] The *comment* text is inserted into the SCCS file as a comment for the initial delta in a manner identical to that of *delta*(1). Omission of the **-y** option results in a default comment line being inserted in the form:

date and time created *YY* **/** *MM* **/** *DD* **/** *HH* **/** *MM* **/** *SS* **by** *login*

The **-y** option is valid only if the **-i** and/or **-n** options are specified (i.e., a new SCCS file is being created).

-m *mrlist* The list of Modification Request (*MR*) numbers is inserted into the SCCS file as the reason for creating the initial delta, in a manner identical to *delta*(1). The **v** flag must be set and the (*MR*) numbers are validated if the **v** flag has a value (the name of an (*MR*) number validation program). Diagnostic messages occur if the **v** flag is not set or (*MR*) validation fails.

-h Causes **admin** to check the structure of the SCCS file (see *sccsfile*(4)), and to compare a newly computed checksum (the sum of all of the characters in the SCCS file except those in the first line) with the checksum that is stored in the first line of the SCCS file. Appropriate error diagnostics are produced.

This option inhibits writing on the file, thus canceling the effect of any other options supplied, and therefore is only meaningful when processing existing files.

-z The SCCS file checksum is recomputed and stored in the first line of the SCCS file (see **-h**, above).

Note that use of this option on a truly corrupted file can prevent future detection of the corruption.

Access Control Lists (ACLs)

Do not add optional ACL entries to SCCS files. SCCS removes them, possibly causing unexpected and undesirable access modes.

EXTERNAL INFLUENCES

Environment Variables

LC_CTYPE determines the interpretation of text as single- and/or multi-byte characters.

LC_MESSAGES determines the language in which messages are displayed.

If **LC_CTYPE** or **LC_MESSAGES** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of **C** (see *lang*(5)) is used instead of **LANG**. If any internationalization variable contains an invalid setting, **admin** behaves as if all internationalization variables are set to **C**. See *environ*(5).

International Code Set Support

Single-byte and multi-byte character code sets are supported.

DIAGNOSTICS

Use *sccshelp*(1) for explanations.

WARNINGS

SCCS files can be any length, but the number of lines in the text file itself cannot exceed 99 999 lines.

FILES

The last component of all SCCS file names must be of the form **s.***filename*. New SCCS files are given mode 444 (see *chmod*(1)). Write permission in the pertinent directory is required to create a file. All writing done by **admin** is to a temporary x-file, called **x.***filename*, (see *get*(1)), created with mode 444 if the **admin** command is creating a new SCCS file, or with the same mode as the SCCS file if it exists. After successful execution of **admin**, the SCCS file is removed (if it exists), and the x-file is renamed to the name of the SCCS file. This ensures that changes are made to the SCCS file only if no errors occurred.

It is recommended that directories containing SCCS files be mode 755 and that SCCS files themselves be mode 444. The mode of any given directory allows only the owner to modify SCCS files contained in that directory. The mode of the SCCS files prevents any modification at all except by SCCS commands.

If it should be necessary to patch an SCCS file for any reason, the mode can be changed to 644 by the owner, thus allowing the use of **vi** or any other suitable editor. *Care must be taken!* The edited file should *always* be processed by an **admin -h** to check for corruption followed by an **admin -z** to generate a proper checksum. Another **admin -h** is recommended to ensure the SCCS file is valid.

admin also makes use of a transient lock file called **z.***filename*), which is used to prevent simultaneous updates to the SCCS file by different users. See *get*(1) for further information.

SEE ALSO

delta(1), ed(1), get(1), sccshelp(1), prs(1), what(1), sccsfile(4), acl(5).

STANDARDS CONFORMANCE

admin: SVID2, SVID3, XPG2, XPG3, XPG4

NAME

answer - phone message transcription system

SYNOPSIS

answer [**-pu**]

DESCRIPTION

The **answer** interactive program helps you to transcribe telephone (and other) messages into electronic mail.

The program uses your personal **elm** alias database and the system **elm** alias database, allowing you to use aliases to address the messages.

Options

answer supports the following options:

-p Prompt for phone-slip-type message fields.

-u Allow addresses that are not aliases.

Operation

answer begins with the **Message to:** prompt. Enter a one-word alias or a two-word user name ("Words" are separated by spaces.) The user name is converted to an alias in the form *f_lastword*, where *f* is the first character of the first word, *lastword* is the second word, and all letters are shifted to lowercase. For example, **Dave Smith** is converted to the alias **d_smith**.

Without the **-u** option, the specified or converted alias must exist in the alias databases. With the **-u** option, if the processed "alias" is not in the alias databases, it is used for the address as is.

The fully expanded address is displayed.

With the **-p** option, you are asked for typical message slip data:

```
Caller:
of:
Phone:

TELEPHONED         -
CALLED TO SEE YOU  -
WANTS TO SEE YOU   -
RETURNED YOUR CALL -
PLEASE CALL        -
WILL CALL AGAIN    -
*****URGENT******  -
```

Enter the appropriate data. You can put just an **X** or nothing after the pertinent dash prompts, or you can type longer comments. Whatever you enter becomes part of the message. Lines with no added text are omitted from the message.

Finally, you are prompted for a message. Enter a message, if any, ending with a blank line. The message is sent and the **Message to:** prompt is repeated.

To end the program, enter any one of **bye**, **done**, **exit**, or **quit**, at the **Message to:** prompt.

EXAMPLES

User input is in normal type.

With No Options

This example shows a valid alias, an invalid user name, and a valid user name. In the invalid case, the converted alias is displayed in square brackets.

```
-----------------------------------------------------------------

Message to: oswald
address 'oswaldr@mycompany.com (Oswald Rarebit)'

Enter message for oswald ending with a blank line.
```

```
> And here is the message.
>

-----------------------------------------------------------------

Message to: albert einstein
Sorry, could not find 'albert einstein' [a_einstein] in list!

Message to: john jones
address 'mrjones@companybee.com (John P. Jones)'

Enter message for john jones ending with a blank line.

> A nice message
>

-----------------------------------------------------------------

Message to: quit
```

With the -u Option

If you enter **answer -u**, the previous error is treated differently.

```
-----------------------------------------------------------------

Message to: albert einstein
address 'a_einstein'

Enter message for albert einstein ending with a blank line.

> About your theory ...
>
```

With the -p Option

If you enter **answer -p**, the phone-slip prompts are added. The three lines with no added text are deleted from the message.

```
-----------------------------------------------------------------

Message to: George Dancer
address 'g_dancer@cup.hp.com (George B. Dancer)'

Caller: Harold Maudlin
of:     Terpsichore Inc.
Phone:  123 456 7890

TELEPHONED          - at 4:30pm
CALLED TO SEE YOU   -
WANTS TO SEE YOU    - X
RETURNED YOUR CALL  -
PLEASE CALL         - X
WILL CALL AGAIN     -
*****URGENT******   - very very!

Enter message for George Dancer ending with a blank line.

> He said that you would ...
>
```

FILES

$HOME/.elm/aliases	User alias database data table
$HOME/.elm/aliases.dir	User alias database directory table

$HOME/.elm/aliases.pag	User alias database hash table
$HOME/.elm/aliases.text	User alias source text
/var/mail/.elm/aliases	System alias database data table
/var/mail/.elm/aliases.dir	System alias database directory table
/var/mail/.elm/aliases.pag	System alias database hash table
/var/mail/.elm/aliases.text	System alias source text
/tmp/snd.*pid*	Outbound mail message edit buffer

AUTHOR

answer was developed by HP.

SEE ALSO

elm(1), newalias(1).

NAME

ar - create and maintain portable archives and libraries

SYNOPSIS

ar [-]*key* [-][*modifier* ...] [*posname*] *afile* [*name* ...]

DESCRIPTION

The **ar** command maintains groups of files combined into a single archive file. Its main use is to create and update library files as used by the link editor (see *ld*(1)). It can be used, however, for any similar purpose. The magic string and file headers used by **ar** consist of printable ASCII characters. If an archive is composed of printable files, the entire archive is printable.

Individual files are inserted without conversion into the archive file. When **ar** creates an archive, it creates headers in a format that is portable across all machines. See *ar*(4) for a detailed description of the portable archive format and structure. The archive symbol table (described in *ar*(4)) is used by the link editor to search repeatedly and efficiently through libraries of object files. An archive symbol table is created and maintained by **ar** only when the archive contains at least one object file. The archive symbol table is in a specially named file that is always the first file in the archive. This file is never mentioned or accessible to the user. Whenever **ar** is used to create or update the contents of an archive, the symbol table is rebuilt (unless the **z** modifier is used). The **s** modifier described below forces the symbol table to be rebuilt.

One *key* operation character from the set, **drqtpmx**, is required and can be optionally preceded by a hyphen (**-**). The required *key* operation character can be specified with one or more *modifier* characters from the set **abcfFilsuvzACT**. *posname* is used with the **r** and **m** key operations and the **a**, **b**, and **i** modifiers to specify a position in the archive. *afile* is the archive file. Constituent files in the archive file are specified by *name* arguments.

The following list describes the *key* operation characters:

d Delete the named files from the archive file.

r Replace the named files, or add a new file to the archive:

- If the **u** modifier is used with the operation character **r**, only those files with modification dates later than those of the corresponding member files are replaced.
- If an optional positioning character from the set **abi** is used, the *posname* argument must be present and specifies that new files are to be placed after (**a**) or before (**b** or **i**) *posname*. In the absence of a positioning character, new files are placed at the end.
- **ar** creates *afile* if it does not already exist.
- If no *name* is specified and:
  - the specified archive file does not exist, **ar** creates an empty archive file containing only the archive header (see *ar*(4)).
  - the archive contains one or more files whose names match names in the current directory, each matching archive file is replaced by the corresponding local file without considering which file may be newer unless the **u** modifier is also specified.

q Quickly append the named files to the end of the archive file. Positioning characters are invalid. The operation does not check to determine whether the added members are already in the archive. **ar** creates *afile* if it does not already exist.

t Print a table of contents of the archive file to the standard output. If no names are given, all files in the archive are described. If names are given, information about only those files appears.

p Print the named files in the archive to the standard output. If no names are specified, the contents of all files are printed in the order that they appear in the archive.

m Move the named files. By default, the files are moved to the end of the archive. If a positioning character is present, the *posname* argument must be present and, as in the **r** operation, *posname* specifies where the files are to be moved. Note that, when used with a positioning character, the files are moved in the same order that they currently appear in the archive, *not* in the order specified on the command line. See EXAMPLES.

x Extract the named files. If no names are given, all files in the archive are extracted. In neither case does **x** alter entries from the archive file.

The following list describes the optional *modifier* characters:

a Position the files after the existing positioning file specified by *posname* .

b Place the new files before the existing positioning file specified by *posname* .

c Suppress the message normally produced when *afile* is created. For **r** and **q** operations, **ar** normally creates *afile* if it does not already exist.

f Truncate the named file names to 14 bytes before performing operations on an archive. This modifier has been provided for compatibility with previous releases where file names up to a maximum of 14 bytes were supported. Longer file names were truncated. When used with the **r** operation, the first existing file that matches the truncated file name is replaced. The **f** modifier can also be used with other operations to allow the full file names to be specified, rather than the truncated file names. Also see the description of the **F** modifier.

i Place the new files before the existing positioning file specified by *posname* . Identical to the **b** modifier.

l Place temporary files in the local current working directory rather than in the directory specified by the environment variable **TMPDIR** or in the default directory **/var/tmp**. Only the **d**, **m**, **q**, and **r** operations and the **s** and **F** modifiers use temporary files.

s Regenerate the archive symbol table even if **ar** is not invoked with an operation that modifies the archive contents. This modifier is useful for restoring the archive symbol table after the **strip** command has been used on the archive (see *strip*(1)) or after the archive has been modified using the **z** modifier.

u Update the archive. (**r** operations only) Do not copy the local file to the archive unless the local file is newer than the corresponding existing file in the archive.

v Give a verbose file-by-file description of the creation or modification of an archive file to the standard output. When used with **t**, **v** gives a long listing of all information about the files. When used with the **d**, **m**, **p**, **q**, or **x** operations, the verbose modifier causes **ar** to print each *key* operation character and the file name associated with that operation. For the **r** operation, **ar** shows an **a** if it adds a new file or an **r** if it replaces an existing one. For the **p** operation, **ar** prints the name of the file to the standard output before the contents of the file are printed.

z Suppress the rebuilding of the symbol table when the archive is modified. This modifier is useful only to avoid long build times when creating a large archive piece-by-piece. If an existing archive contains a symbol table, the **z** modifier will cause it to be invalidated. If a file name longer than 15 bytes is given the entire archive is rewritten. To rebuild the symbol table, either use the **ranlib** command (see *ranlib*(1)), or invoke **ar** again with the **s** modifier.

A Suppress warning messages regarding optional access control list entries. **ar** does not archive optional access control list entries in a file's access control list (see *acl*(5)). Normally, a warning message is printed for each file having optional access control list entries.

C Prevent extracted files from replacing files with the same name. The **C** modifier can only be used with the **x** operation.

F Truncate the entire archive. The **F** modifier causes the entire archive to be rewritten such that all file names within the archive are truncated to 14 bytes, even when **ar** does not modify the archive contents. The long name table will be removed (see *ar*(4)). This modifier has been provided for compatibility with previous releases where file names up to a maximum of 14 bytes were supported. Also see the description of the **f** modifier.

T Truncate file names whose archive names are longer than those supported by the file system. By default, files with names longer than those supported by the file system will not be extracted and will cause an error. The **T** modifier can only be used with the **x** operation.

Only the following combinations are meaningful; no other combination of modifiers with operations have any effect on the operation:

d:	**v**, **f**, **F**, **l**
m:	**v**, **f**, **F**, **l**, and **a** | **b** | **i**
r:	**v**, **f**, **F**, **l**, **c**, **A**, **u**, and **a** | **b** | **i**
q:	**v**, **f**, **F**, **l**, **c**, **A**, **z**
t:	**v**, **f**, **F**, **s**

p: **v**, **f**, **F**, **s**
x: **v**, **f**, **F**, **s**, **C**, **T**

EXTERNAL INFLUENCES

Environment Variables

The following internationalization variables affect the execution of **ar**:

LANG
Determines the locale category for native language, local customs and coded character set in the absence of **LC_ALL** and other **LC_*** environment variables. If **LANG** is not specified or is set to the empty string, a default of **C** (see *lang*(5)) is used instead of **LANG**.

LC_ALL
Determines the values for all locale categories and has precedence over **LANG** and other **LC_*** environment variables.

LC_CTYPE
Determines the locale category for character handling functions.

LC_MESSAGES
Determines the locale that should be used to affect the format and contents of diagnostic messages written to standard error.

LC_NUMERIC
Determines the locale category for numeric formatting.

LC_TIME
Determines the format and contents of date and time formatting.

NLSPATH
Determines the location of message catalogues for the processing of **LC_MESSAGES**.

If any internationalization variable contains an invalid setting, **ar** behaves as if all internationalization variables are set to **C**. See *environ*(5).

In addition, the following environment variable affects **ar**:

TMPDIR
Specifies a directory for temporary files (see *tmpnam*(3S)). The **l** modifier overrides the **TMPDIR** variable, and **TMPDIR** overrides **/var/tmp**, the default directory.

DIAGNOSTICS

phase error on *file name*
The named file was modified by another process while **ar** was copying it into the archive. When this happens, **ar** exits and the original archive is left untouched.

ar write error: file system error message
ar could not write to a temporary file or the final output file. If **ar** was trying to write the final output file, the original archive is lost.

ar reports **cannot create** *file***.a**, where *file***.a** is an **ar**-format archive file, even if *file***.a** already exists. This message is triggered when *file***.a** is write-protected or inaccessible.

EXAMPLES

Create a new file (if one does not already exist) in archive format with its constituents entered in the order indicated:

```
ar r newlib.a f3 f2 f1 f4
```

Replace files **f2** and **f3** such that the new copies follow file **f1**, and **f3** follows **f2**:

```
ar ma f1 newlib.a f2 f3
ar ma f2 newlib.a f3
ar r newlib.a f2 f3
```

The archive is then ordered:

```
newlib.a:  f1 f2' f3' f4
```

where the single quote marks indicate updated files. The first command says "move **f2** and **f3** after **f1** in *newlib*.a", thus creating the order:

```
f1 f3 f2 f4
```

Note that the relative order of **f2** and **f3** has not changed. The second command says "move **f3** after **f2** in *newlib.a*", creating the order:

```
f1 f2 f3 f4
```

The third command then replaces files **f2** and **f3**. Since files **f2** and **f3** both already existed in the archive, this sequence of commands could not be simply replaced by:

```
ar ra f1 newlib.a f2 f3
```

because the previous position and relative order of **f2** and **f3** in the archive are preserved (no matter how the files are specified on the command line), producing the following archive:

```
newlib.a:  f3' f2' f1 f4
```

WARNINGS

If you are a user who has appropriate privileges, **ar** can alter any archive file, even if it is write-protected.

If the same file is mentioned twice in an argument list, it might be put in the archive twice.

If multiple copies of a file exist in an archive, **ar** matches the first occurrence of the file in the archive.

ar automatically creates an archive symbol table, a task performed in early HP-UX versions by **ranlib**. Use of the **z** modifier either suppresses generation of the symbol table, or invalidates it if it exists. The **ranlib** command can be used to rebuild the symbol table if an archive was built with the **z** modifier.

FILES

/var/tmp/ar*	Temporary files

SEE ALSO

System Tools:

ld(1)	Invoke the link editor

Miscellaneous:

acl(5)	Access control lists
a.out(4)	Assembler, compiler, and linker output
ar(4)	Archive format
lorder(1)	Find the ordering relation for object files or archive libraries
ranlib(1)	Regenerate an archive symbol table
strip(1)	Strip symbol and line number information from an object file
tmpnam(3S)	Create a name for a temporary file

STANDARDS CONFORMANCE

ar: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

NAME

as - assembler (Precision Architecture)

SYNOPSIS

as [[*option*]... [*file*] ...]...

DESCRIPTION

The **as** command assembles source text from files or standard input and produces a relocatable object file suitable for the link editor, **ld** (see *ld*(1).

Source text is read from standard input only if no *file* argument is given. Standard input cannot be a device file, such as a terminal. The *option* and *file* arguments can be intermingled on the command line. Every specified *option* applies to every specified *file*, or standard input. The source files are concatenated to form a single input stream.

If the **-o** *objfile* option is not specified, the **.s** suffix (if any) is stripped from the end of the last source file name and **.o** is appended to the name to form the name of the default object code output file.

as output is not optimized. **as** creates a relocatable object file that must be processed by **ld** before it can be successfully executed (see *ld*(1)).

The **cc** compiler normally runs the C preprocessor **cpp** (see *cpp*(1)), then invokes **as** to assemble the **.s** files together with **/usr/lib/pcc_prefix.s**, and subsequently invokes **ld**.

Arguments

as recognizes the following arguments.

file	A text file contain assembler source code.
option	An option described below in Options.

Options

as recognizes the following values for the *option* argument.

-e	Permit an unlimited number of errors to be tolerated before the assembly process is abandoned. By default, one hundred errors are allowed before the assembler aborts.
-f	Set the default value for the **.CALLINFO** directive to **CALLS**, The normal default value for a **.CALLINFO** that omits the **CALLER**, **CALLS** or **NO_CALLS** parameter is **NO_CALLS**.
-l	Write a listing of the program to standard output after assembly. This listing shows the offsets of instructions and actual values for fields.
-o *objfile*	Name the output object file *objfile* instead of using the default **.o** suffix on the file name of the last *file* specified.
-p *number*	Set the default privilege level for an **.EXPORT** directive to *number*. By default, all user-level procedures are exported at privilege level 3.
-s	Set the output file suffix to **.ss** instead of **.o**. The file will have a format suitable for conversion to the ROM burning programs.
-u	Do not create unwind descriptors. To avoid the need for the **.CALLINFO** directive, the **.ENTER** and **.LEAVE** directives must not have been used.
-v *xrfile*	Write cross-reference data to the file named *xrfile*.
-V	Print the version number of the assembler program to standard error before assembling the source text.
-w[*number*]	Either suppress all warning messages if no *number* is supplied or suppress just the warning *number* provided. Multiple **-w***number* options can be used to suppress additional warning messages.
+DA*architecture*	Assemble code for the *architecture* specified. The use of this option is discouraged. The preferred method for selecting the *architecture* is to have a **.LEVEL** directive contained within the assembly source file.

The assembler uses the following precedence to determine the target architecture.

1. Use the **.LEVEL** directive within the assembly source file.
2. Use the **+DA** command-line specification.
3. Use the default architecture of **PA_RISC_1.0**.

+z,+Z Both of these options are used in the building of shared libraries. For a more complete discussion regarding these options, see the manual *HP-UX Linker and Libraries User's Guide*.

EXTERNAL INFLUENCES

International Code Set Support

Single- and multibyte character code sets are supported.

DIAGNOSTICS

When syntactic or semantic errors occur, a single-line diagnostic is written to standard error, that includes the file name and the line number where it occurred. The format is as follows:

as: "*filename***",line** *line***: error** *error***:** *message*
  source = *source-line*

WARNINGS

as does not invoke *cpp*(1) or *m4*(1) to perform macro processing.

FILES

/usr/include/hard_reg.hr	Hardware register definitions
/usr/include/soft_reg.h	Software calling convention register definitions
/usr/include/std_space.h	Standard space and subspace definition
/usr/lib/nls/msg/C/as.cat	Assembler error message catalog
/usr/lib/pcc_prefix.s	Space, subspace and register definitions
*file***.o**	Object file

SEE ALSO

adb(1), cc(1), cc_bundled(1), ld(1), nm(1), xdb(1), crt0(3).

Assembly Language Reference Manual,
Precision Architecture and Instruction Reference Manual,
Procedure Calling Conventions Reference Manual.

NAME
asa - interpret ASA carriage control characters

SYNOPSIS
asa [*files*]

DESCRIPTION
asa interprets the output of FORTRAN programs that utilize ASA carriage control characters. It processes either the *files* whose names are given as arguments, or the standard input if **-** is specified or if no file names are given. The first character of each line is assumed to be a control character. The following control characters are interpreted as indicated:

(blank)	Output a single new-line character before printing.
(space)	(XPG4 only.) The rest of the line will be output without change.
0	Output two new-line characters before printing.
0	(XPG4 only.) A <newline> shall be output, then the rest of the input line.
1	Output a new-page character before printing.
+	Overprint previous line.
+	(XPG4 only.) The <newline> of the previous line shall be replaced with one or more implementaions-defined characters that causes printing to return to column position 1, followed by the rest of the input line. If the + is the first character in the input, it shall have the same effect as <space>.

Lines beginning with other than the above characters are treated the same as lines beginning with a blank. The first character of a line is *not* printed. If any such lines appear, an appropriate diagnostic is sent to standard error. This program forces the first line of each input file to start on a new page.

(XPG4 only.) The action of the asa utility is unspecified upon encountering any character other than those listed above as the first character in a line.

To view the output of FORTRAN programs which use ASA carriage control characters and have them appear in normal form, **asa** can be used as a filter:

```
a.out | asa | lp
```

The output, properly formatted and paginated, is then directed to the line printer. FORTRAN output previously sent to a file can be viewed on a user terminal screen by using:

asa *file*

EXTERNAL INFLUENCES
Environment Variables
LC_CTYPE determines the interpretation of text within file as single- and/or multi-byte characters.

LC_MESSAGES determines the language in which messages are displayed.

If **LC_CTYPE** or **LC_MESSAGES** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**.

If any internationalization variable contains an invalid setting, **asa** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support
Single- and multi-byte character code sets are supported.

SEE ALSO
efl(1), f77(1), fsplit(1), ratfor(1).

STANDARDS CONFORMANCE
asa: XPG4, POSIX.2

NAME

at, batch - execute batched commands immediately or at a later time

SYNOPSIS

Enter commands from standard input to run at a specified time:

at [**-m**] [**-q** *queue*] **-t**
spectime"
commands
eof

at [**-m**] [**-q** *queue*]
time [*date*] [*next*
timeunit | + *count timeunit*]"
commands
eof

Enter commands from a file to run at a specified time:

at -f *job-file* [**-m**] [**-q** *queue*] **-t** *spectime*

at -f *job-file* [**-m**] [**-q** *queue*] *time* [*date*] [**next** *timeunit* | **+***count timeunit*]

List scheduled jobs:

at -d *job-id* ...

at -l [*job-id* ...]

at -l -q *queue*

Cancel (remove) a scheduled job:

at -r *job-id* ...

Enter commands from standard input to run as a batch process:

batch
commands
eof

Enter commands from a file to run as a batch process:

batch < *job-file*

DESCRIPTION

The **at** and **batch** commands schedule jobs for execution by the **cron** daemon (see *cron*(1M)).

at schedules a job for execution at a specified time. **at** can also list (**-l**) or remove (**-r**) existing scheduled **at** and **batch** jobs.

batch schedules a job for execution immediately, or as soon as system load levels permit.

You can enter commands into a job in one of the following ways:

- From the keyboard on separate lines immediately after the **at** or **batch** command line, followed by the currently defined *eof* (end-of-file) character to end the input. The default *eof* is Ctrl-**D**. It can be redefined in your environment (see *stty*(1)).
- With the **-f** option of the **at** command to read input from a script file.
- From output piped from a preceding command.

Options and Arguments

at recognizes the following options and arguments.

commands	One or more HP-UX commands that can be executed as a shell script by **at** or **batch**.
eof	End-of-file character. The default is Ctrl-**D** unless defined otherwise in your environment.
job-file	The path name of an existing file.

job-id — The job identifier reported by **at** or **batch** when the job was originally scheduled.

-d *job-id* ... — Displays the contents of the specified job. An unprivileged user is restricted to display information only on jobs that the user owns. A user with the appropriate privileges is able to display information about all jobs.

-f *job-file* — Read in the commands contained in *job-file* instead of using standard input.

-l [*job-id* ...] — List the jobs specified. If no *job-id*s are given, all jobs are listed.

-m — Send mail to the invoking user after the job has run, announcing its completion. Unless redirected elsewhere within the job, standard output and standard error produced by the job are automatically mailed to the user as well.

-q *queue* — Submit the specified job to the *queue* indicated (see *queuedefs*(4)). Queues **a**, **b**, and **d** through **y** can be used. **at** uses queue **a** by default. **batch** always uses queue **b**. All queues except **b** require a *time* or a **-t** specification. **at -qb** is equivalent to **batch**.

-r *job-id* ... — Remove the jobs specified by each *job-id*.

-t *spectime* — Define the absolute time to start the job.

spectime — A date and time in the format:

[[*CC*]*YY*]*MMDDhhmm* [**.** *ss*]

where the decimal digit pairs are as follows:

CC	The first two digits of the year (**19**, **20**).
YY	The second two digits of the year (**69**–**99**, **00**–**68**). See WARNINGS.
MM	The month of the year (**01**–**12**).
DD	The day of the month (**01**–**31**).
hh	The hour of the day (**00**–**23**).
mm	The minute of the hour (**00**–**59**).
ss	The second of the minute (**00**–**61**).

If both *CC* and *YY* are omitted, the default is the current year.

If *CC* is omitted and *YY* is in the range **69**–**99**, *CC* defaults to **19**. Otherwise, **CC** defaults to **20**.

The range for *ss* provides for two leap seconds. If *ss* is **60** or **61**, and the resulting time, as affected by the **TZ** environment variable, does not refer to a leap second, the time is set to the whole minute following *mm*.

If *ss* is omitted, it defaults to **00**.

time [*date*] — Define the base time for starting the job.

time — A time specified as one, two, or four digits. One- and two-digit numbers represent hours; four digits represent hours and minutes.

Alternately, *time* can be specified as two numbers separated by a colon (**:**), a single quote (**’**), the letter *h* (**h**), a period (**.**), or a comma (**,**). Spaces may be present between the separator and digits representing minutes. If defined in *langinfo*(5), special time unit characters can be used.

am or **pm** can be appended to indicate morning or afternoon. Otherwise, a 24-hour clock is understood. For example, **0815**, **8:15**, **8’15**, **8h15**, **8.15**, and **8,15** are read as 15 minutes after eight in the morning. The suffixes **zulu** and **utc** can be used to specify Coordinated Universal Time (UTC), equivalent to Greenwich Mean Time (GMT).

The special names **midnight**, **noon**, and **now** are also recognized.

date — A day of the week (fully spelled out or abbreviated) or a date consisting of a day, a month, and optionally a year. The day and year fields must be numeric, and the month can be either fully spelled out, abbreviated, or numeric. The locales of these three fields are in the order of month, date and year (such as month/date/year). They are separated by punctuation marks such as slash (**/**), hyphen (**-**), period (**.**), and comma (**,**). If defined in *langinfo*(5), special date unit characters can be present. If a given date is ambiguous (such as **2/5**), the **D_T_FMT** string (if defined in *langinfo*(5)) is used to resolve the ambiguity.

Two special days, **today** and **tomorrow**, are also recognized. If no *date* is given, **today** is assumed if the given time is greater than the current time; **tomorrow** is assumed if it is less.

If the given month is less than the current month (and no year is given), next year is assumed. Two-digit years in the range 69 to 99 are expanded to 1969 to 1999; in the range 00 to 68, to 2000 to 2068.

next *timeunit* | **+** *count timeunit*

Delay the execution date and time by a specific number of time units after the base time specified by *time* [*date*].

count A decimal number. **next** is equivalent to **+1**.

timeunit A time unit, one of the following: **minutes**, **hours**, **days**, **weeks**, **months**, or **years**, or their singular forms.

How Jobs Are Processed

When a job is accepted, **at** and **batch** print a message to standard error in the form:

job *job-id* **at** *execution-date*

where *job-id* is the job identifier in the form *jobnumber***.***queue*, such as **756284400.a**, and *execution-date* is the date and time when the job will be released for execution.

If your login shell is not the POSIX shell (**/usr/bin/sh**), the commands also print a warning message:

```
warning: commands will be executed using /usr/bin/sh
```

at jobs default to queue **a**. **batch** jobs always go in queue **b**. See the **-q** option.

An **at** or **batch** job consists of a two-part script stored in **/var/spool/cron/atjobs** that can be executed by the POSIX shell.

The first part sets up the environment to match the environment when the **at** or **batch** command was issued. This includes the current shell environment variables, current directory, **umask**, and **ulimit** (see *ulimit*(2), *umask*(1), and *proto*(4)). Open file descriptors, traps, and priority are lost.

The second part consists of the commands that you entered.

When **cron** dispatches the job, it starts a POSIX shell to execute the script.

The number of jobs executing from a queue at any time is controlled by parameters in the file **/var/adm/cron/queuedefs** (see *queuedefs*(4)).

Standard output and standard error from the job are mailed to the user unless they are redirected elsewhere within the job.

Scheduled jobs are immune to the **SIGHUP** hangup signal, and remain scheduled if the user logs off.

Users are permitted to use the **at** and **batch** commands if their user names appear in the file **/usr/lib/cron/at.allow**. If that file does not exist, users can use **at** and **batch** if their names *do not* appear in the file **/usr/lib/cron/at.deny**. If neither file exists, only superuser is allowed to submit jobs. If only **at.deny** exists but is empty, all users can use **at** and **batch**. The **allow**/**deny** files consist of one user name per line.

All users can list and remove their own jobs. Users with appropriate privileges can list and remove jobs other than their own.

EXTERNAL INFLUENCES

Environment Variables

LC_TIME determines the format and contents of date and time strings.

LC_MESSAGES determines the language in which messages are displayed.

LC_MESSAGES also determines the language in which the words **days**, **hours**, **midnight**, **minutes**, **months**, **next**, **noon**, **now**, **today**, **tomorrow**, **weeks**, **years**, and their singular forms can also be specified.

IF **LC_TIME** or **LC_MESSAGES** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**.

If any internationalization variable contains an invalid setting, all internationalization variables default to "C" (see *environ*(5)).

International Code Set Support

Single- and multi-byte character code sets are supported.

RETURN VALUE

The exit code is set to one of the following:

0	Successful completion
1	Failure

DIAGNOSTICS

at produces self-explanatory messages for syntax errors and out-of-range times.

warning: commands will be executed using /usr/bin/sh

If your login shell is not the POSIX shell (**/usr/bin/sh**), **at** and **batch** produce a warning message as a reminder that **at** and **batch** jobs are executed using **/usr/bin/sh**.

EXAMPLES

The following commands show three different ways to run a POSIX shell script file named **delayed-job** five minutes from now:

```
at -f delayed-job now + 5 minutes
cat delayed-job | at now + 5 minutes
at now + 5 minutes <delayed-job
```

Run a typical HP-UX command (**nroff** in this case) when system load levels permit, and redirect standard output and standard error to files:

```
batch
nroff source-file >output-file 2>error-file
eof        (the default is Ctrl-D)
```

Run a job contained in **future** in the home directory at 12:20 a.m. on December 27, 2013:

```
at -f $HOME/future -t201312271220.00
```

Redirect standard error to a pipe (useful in a shell procedure). Note that the sequence of the output redirection specifications is significant. Standard error is redirected to where standard output is going; standard output is redirected to a file; the original "standard output" (which now consists of the former standard error) is piped to the **mail** program.

```
batch <<!!     (sets eof temporarily to !!)
nroff input-file 2>&1 1> output-file | mail loginid
!!
```

Run a job contained in **jobfile** in the home directory at 5:00 a.m. next Tuesday:

```
at -f $HOME/jobfile 5am tuesday next week
```

Run the same job at 5:00 a.m. one week from next Tuesday (i.e., 2 Tuesdays in advance):

```
at -f $HOME/jobfile 5am tuesday + 2 weeks
```

Add a command to the file named **weekly-run** in directory **jobs** in the home directory so that it automatically reschedules itself every time it runs. This example reschedules itself every Thursday at 1900 (7:00 p.m.):

```
echo "sh $HOME/jobs/weekly-run" | at 1900 thursday next week
```

The following commands show several forms recognized by **at** and include native language usage:

```
at 0815 Jan 24
at 8:15 Jan 24
at 9:30am tomorrow
at now + 1 day
at -f job 5 pm Friday
at 17:40 Tor.                # in Danish
at 17h46 demain              # in French
at 5:30 26. Feb. 1988        # in German
at 12:00 26-02               # in Finnish
```

WARNINGS

If the *date* argument begins with a number and the *time* argument is also numeric without a suffix, the *time* argument should be a four-digit number that can be correctly interpreted as hours and minutes.

If you use both **next** and **+***count* within a single **at** command, the first operator is accepted and the trailing operator is silently ignored.

If you use both **-t** and *time* ... in the same command, the first specified is accepted and the second is silently ignored.

If the FIFO used to communicate with **cron** fills up, **at** is suspended until **cron** has read sufficient messages from the FIFO to make room for the message **at** is trying to write. This condition can occur if **at** is writing messages faster than **cron** can process them or if **cron** is not executing.

Scheduled processes are run in the background. Any script file that calls itself will cause the user or the system to run out of available processes.

If the execution-time request for a job duplicates the execution time of a currently scheduled job, the new job time is set to the next available second.

at will not schedule jobs whose start time precedes the current Epoch (00:00:00 January 1, 1970 UTC).

at will not schedule jobs beyond the year 2030.

DEPENDENCIES

HP Process Resource Manager

If the optional HP Process Resource Management (PRM) software is installed and configured, jobs are launched in the initial process resource group of the user that scheduled the job. The user's initial group is determined at the time the job is started, not when the job is scheduled. If the user's initial group is not defined, the job runs in the user default group (**PRMID=1**). See *prmconfig*(1) for a description of how to configure HP PRM, and *prmconf*(4) for a description of how the user's initial process resource group is determined.

AUTHOR

at was developed by AT&T and HP.

FILES

/usr/bin/sh	POSIX shell
/var/adm/cron	Main **cron** directory
/var/adm/cron/.proto	This file contains a set of shell commands which are added to the **at** job file to make the environment for the **at** job same as the current environment. See *proto(4).*
/usr/lib/cron/at.allow	List of allowed users
/usr/lib/cron/at.deny	List of denied users
/var/adm/cron/queuedefs	Scheduling information
/var/spool/cron/atjobs	Spool area

SEE ALSO

crontab(1), kill(1), mail(1), nice(1), ps(1), sh(1), stty(1), cron(1M), proto(4), queuedefs(4), hpnls(5).

HP Process Resource Manager:
prmconfig(1), prmconf(4) in *HP Process Resource Manager User's Guide*.

STANDARDS CONFORMANCE

`at`: SVID2, SVID3, XPG2, XPG3, XPG4

`batch`: SVID2, SVID3, XPG2, XPG3, XPG4

NAME

attributes - describe an audio file

SYNOPSIS

/opt/audio/bin/attributes *filename*

DESCRIPTION

This command provides information about an audio file, including file format, data format, sampling rate, number of channels, data length and header length.

EXAMPLE

The following is an example of using **attributes** on an audio file supplied with HP-UX.

```
$ /opt/audio/bin/attributes /opt/audio/sounds/welcome.au

File Name: /opt/audio/sounds/welcome.au
File Type: NeXT/Sun
Data Format: Mu-law
Sampling Rate: 22050
Channels: Mono
Duration: 1.972 seconds
Bits per Sample: 8
Header Length: 40 bytes
Data Length: 43492 bytes
```

AUTHOR

attributes was developed by HP.

Sun is a trademark of Sun MicroSystems, Inc.

NeXT is a trademark of NeXT Computers, Inc.

SEE ALSO

audio(5), asecure(1M), aserver(1M), convert(1), send_sound(1).

Using the Audio Developer's Kit

NAME

awk - pattern-directed scanning and processing language

SYNOPSIS

awk [**-F***fs*] [**-v** *var***=***value*] [*program* | **-f** *progfile* ...] [*file* ...]

DESCRIPTION

awk scans each input *file* for lines that match any of a set of patterns specified literally in *program* or in one or more files specified as **-f** *progfile*. With each pattern there can be an associated action that is to be performed when a line in a *file* matches the pattern. Each line is matched against the pattern portion of every pattern-action statement, and the associated action is performed for each matched pattern. The file name **-** means the standard input. Any *file* of the form *var***=***value* is treated as an assignment, not a filename. An assignment is evaluated at the time it would have been opened if it were a filename, unless the **-v** option is used.

An input line is made up of fields separated by white space, or by regular expression **FS**. The fields are denoted **$1**, **$2**, ...; **$0** refers to the entire line.

Options

awk recognizes the following options and arguments:

-F *fs*	Specify regular expression used to separate fields. The default is to recognize space and tab characters, and to discard leading spaces and tabs. If the **-F** option is used, leading input field separators are no longer discarded.
-f *progfile*	Specify an awk program file. Up to 100 program files can be specified. The pattern-action statements in these files are executed in the same order as the files were specified.
-v *var***=***value*	Cause *var***=***value* assignment to occur before the **BEGIN** action (if it exists) is executed.

Statements

A pattern-action statement has the form:

pattern **{** *action* **}**

A missing **{** *action* **}** means print the line; a missing pattern always matches. Pattern-action statements are separated by new-lines or semicolons.

An action is a sequence of statements. A statement can be one of the following:

```
if( expression ) statement [ else statement ]
while( expression ) statement
for( expression ; expression ; expression ) statement
for( var in array ) statement
do statement while( expression )
break
continue
{[ statement  ... ]}
expression                          # commonly  var = expression
print [ expression-list ] [ > expression ]
printf format [, expression-list ] [ > expression ]
return [ expression ]
next                                # skip remaining patterns on this input line.
delete array [ expression ]         # delete an array element.
exit [ expression ]                 # exit immediately; status is expression.
```

Statements are terminated by semicolons, newlines or right braces. An empty *expression-list* stands for **$0**. String constants are quoted (**" "**), with the usual C escapes recognized within. Expressions take on string or numeric values as appropriate, and are built using the operators **+**, **-**, *****, **/**, **%**, **^** (exponentiation), and concatenation (indicated by a blank). The operators **++**, **- -**, **+=**, **-=**, ***=**, **/=**, **%=**, **^=**, ****=**, **>**, **>=**, **<**, **<=**, **==**, **!=**, and **?:** are also available in expressions. Variables can be scalars, array elements (denoted *x***[***i***]**) or fields. Variables are initialized to the null string. Array subscripts can be any string, not necessarily numeric (this allows for a form of associative memory). Multiple subscripts such as **[** *i***,***j***,***k* **]** are permitted. The constituents are concatenated, separated by the value of **SUBSEP**.

The **print** statement prints its arguments on the standard output (or on a file if **>***file* or **>>***file* is present or on a pipe if **|***cmd* is present), separated by the current output field separator, and terminated by the output record separator. *file* and *cmd* can be literal names or parenthesized expressions. Identical string values in different statements denote the same open file. The **printf** statement formats its expression list according to the format (see *printf*(3)).

Built-In Functions

The built-in function **close(***expr***)** closes the file or pipe **expr** opened by a **print** or **printf** statement or a call to **getline** with the same string-valued *expr*. This function returns zero if successful, otherwise, it returns non-zero.

The customary functions **exp**, **log**, **sqrt**, **sin**, **cos**, **atan2** are built in. Other built-in functions are:

blength[**(**[*s*]**)**]	Length of its associated argument (in bytes) taken as a string, or of **$0** if no argument.
length[**(**[*s*]**)**]	Length of its associated argument (in characters) taken as a string, or of **$0** if no argument.
rand()	Returns a random number between zero and one.
srand([*expr*]**)**	Sets the seed value for *rand*, and returns the previous seed value. If no argument is given, the time of day is used as the seed value; otherwise, *expr* is used.
int(*x***)**	Truncates to an integer value
substr(*s***,** *m* [**,** *n*]**)**	Return the at most *n*-character substring of *s* that begins at position *m*, numbering from 1. If *n* is omitted, the substring is limited by the length of string *s*.
index(*s***,** *t* **)**	Return the position, in characters, numbering from 1, in string *s* where string *t* first occurs, or zero if it does not occur at all.
match(*s***,** *ere* **)**	Return the position, in characters, numbering from 1, in string *s* where the extended regular expression *ere* occurs, or 0 if it does not. The variables **RSTART** and **RLENGTH** are set to the position and length of the matched string.
split(*s***,** *a*[**,** *fs*]**)**	Splits the string *s* into array elements *a***[1]**, *a***[2]**, ..., *a***[***n***]**, and returns *n*. The separation is done with the regular expression *fs*, or with the field separator **FS** if *fs* is not given.
sub(*ere***,** *repl* [**,** *in*]**)**	Substitutes *repl* for the first occurrence of the extended regular expression *ere* in the string *in*. If *in* is not given, **$0** is used.
gsub	Same as **sub** except that all occurrences of the regular expression are replaced; **sub** and **gsub** return the number of replacements.
sprintf(*fmt***,** *expr***, ...)**	String resulting from formatting *expr* ... according to the *printf*(3S) format *fmt*
system(*cmd***)**	Executes *cmd* and returns its exit status
toupper(*s***)**	Converts the argument string *s* to uppercase and returns the result.
tolower(*s***)**	Converts the argument string *s* to lowercase and returns the result.

The built-in function **getline** sets **$0** to the next input record from the current input file; **getline <** *file* sets **$0** to the next record from *file*. **getline** *x* sets variable *x* instead. Finally, *cmd* **| getline** pipes the output of *cmd* into **getline**; each call of **getline** returns the next line of output from *cmd*. In all cases, **getline** returns 1 for a successful input, 0 for end of file, and −1 for an error.

Patterns

Patterns are arbitrary Boolean combinations (with **! || &&**) of regular expressions and relational expressions. **awk** supports Extended Regular Expressions as described in *regexp*(5). Isolated regular expressions in a pattern apply to the entire line. Regular expressions can also occur in relational expressions, using the operators **~** and **!~**. **/***re***/** is a constant regular expression; any string (constant or variable) can be used as a regular expression, except in the position of an isolated regular expression in a pattern.

A pattern can consist of two patterns separated by a comma; in this case, the action is performed for all lines from an occurrence of the first pattern though an occurrence of the second.

A relational expression is one of the following:

expression matchop regular-expression
expression relop expression
expression **in** *array-name*
(*expr,expr,...***)** *in* **array-name**

where a relop is any of the six relational operators in C, and a matchop is either **~** (matches) or **!~** (does not match). A conditional is an arithmetic expression, a relational expression, or a Boolean combination of the two.

The special patterns **BEGIN** and **END** can be used to capture control before the first input line is read and after the last. **BEGIN** and **END** do not combine with other patterns.

Special Characters

The following special escape sequences are recognized by **awk** in both regular expressions and strings:

Escape	Meaning
\a	alert character
\b	backspace character
\f	form-feed character
\n	new-line character
\r	carriage-return character
\t	tab character
\v	vertical-tab character
\*nnn*	1- to 3-digit octal value *nnn*
\x*hhh*	1- to n-digit hexadecimal number

Variable Names

Variable names with special meanings are:

FS	Input field separator regular expression; a space character by default; also settable by option **-F***fs*.
NF	The number of fields in the current record.
NR	The ordinal number of the current record from the start of input. Inside a **BEGIN** action the value is zero. Inside an **END** action the value is the number of the last record processed.
FNR	The ordinal number of the current record in the current file. Inside a **BEGIN** action the value is zero. Inside an **END** action the value is the number of the last record processed in the last file processed.
FILENAME	A pathname of the current input file.
RS	The input record separator; a newline character by default.
OFS	The **print** statement output field separator; a space character by default.
ORS	The **print** statement output record separator; a newline character by default.
OFMT	Output format for numbers (default **%.6g**). If the value of **OFMT** is not a floating-point format specification, the results are unspecified.
CONVFMT	Internal conversion format for numbers (default **%.6g**). If the value of **CONVFMT** is not a floating-point format specification, the results are unspecified.
SUBSEP	The subscript separator string for multi-dimensional arrays; the default value is "\034"
ARGC	The number of elements in the **ARGV** array.
ARGV	An array of command line arguments, excluding options and the *program* argument numbered from zero to **ARGC**-1.

The arguments in **ARGV** can be modified or added to; **ARGC** can be altered. As each input file ends, **awk** will treat the next non-null element of **ARGV**, up to the current value of **ARGC**-1, inclusive, as the name of the next input file. Thus, setting an element of **ARGV** to null means that it will not be treated as an input file. The name **-** indicates the standard input. If an argument matches the format of an *assignment* operand, this argument will be treated as an assignment rather than a *file* argument.

ENVIRON Array of environment variables; subscripts are names. For example, if environment variable **V=thing**, **ENVIRON["V"]** produces **thing**.

RSTART The starting position of the string matched by the **match** function, numbering from 1. This is always equivalent to the return value of the **match** function.

RLENGTH The length of the string matched by the **match** function.

Functions can be defined (at the position of a pattern-action statement) as follows:

```
function foo(a, b, c) { ...; return x }
```

Parameters are passed by value if scalar, and by reference if array name. Functions can be called recursively. Parameters are local to the function; all other variables are global.

Note that if pattern-action statements are used in an HP-UX command line as an argument to the **awk** command, the pattern-action statement must be enclosed in single quotes to protect it from the shell. For example, to print lines longer than 72 characters, the pattern-action statement as used in a script (**-f** *progfile* command form) is:

```
length > 72
```

The same pattern action statement used as an argument to the **awk** command is quoted in this manner:

```
awk 'length > 72'
```

EXTERNAL INFLUENCES

Environment Variables

LANG Provides a default value for the internationalization variables that are unset or null. If **LANG** is unset or null, the default value of "C" (see *lang*(5)) is used. If any of the internationalization variables contains an invalid setting, **awk** will behave as if all internationalization variables are set to "C". See *environ*(5).

LC_ALL If set to a non-empty string value, overrides the values of all the other internationalization variables.

LC_CTYPE Determines the interpretation of text as single and/or multi-byte characters, the classification of characters as printable, and the characters matched by character class expressions in regular expressions.

LC_NUMERIC Determines the radix character used when interpreting numeric input, performing conversion between numeric and string values and formatting numeric output. Regardless of locale, the period character (the decimal-point character of the POSIX locale) is the decimal-point character recognized in processing **awk** programs (including assignments in command-line arguments).

LC_COLLATE Determines the locale for the behavior of ranges, equivalence classes and multi-character collating elements within regular expressions.

LC_MESSAGES
Determines the locale that should be used to affect the format and contents of diagnostic messages written to standard error and informative messages written to standard output.

NLSPATH Determines the location of message catalogues for the processing of **LC_MESSAGES.**

PATH Determines the search path when looking for commands executed by **system(cmd)**, or input and output pipes.

In addition, all environment variables will be visible via the **awk** variable **ENVIRON**.

International Code Set Support

Single- and multi-byte character code sets are supported except that variable names must contain only ASCII characters and regular expressions must contain only valid characters.

DIAGNOSTICS

awk supports up to 199 fields (**$1**, **$2**, ..., **$199**) per record.

EXAMPLES

Print lines longer than 72 characters:

```
length > 72
```

Print first two fields in opposite order:

```
{ print $2, $1 }
```

Same, with input fields separated by comma and/or blanks and tabs:

```
BEGIN { FS = ",[ \t]*|[ \t]+" }
      { print $2, $1 }
```

Add up first column, print sum and average:

```
        { s += $1 }"
END     { print "sum is", s, " average is", s/NR }
```

Print all lines between start/stop pairs:

```
/start/, /stop/
```

Simulate **echo** command (see *echo*(1)):

```
BEGIN   {                              # Simulate echo(1)
        for (i = 1; i < ARGC; i++) printf "%s ", ARGV[i]
        printf "\n"
        exit }
```

AUTHOR

awk was developed by AT&T, IBM, OSF, and HP.

SEE ALSO

lex(1), sed(1).

A. V. Aho, B. W. Kernighan, P. J. Weinberger: *The AWK Programming Language*, Addison-Wesley, 1988.

STANDARDS CONFORMANCE

awk: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

NAME

banner - make posters in large letters

SYNOPSIS

banner *strings*

DESCRIPTION

banner prints its arguments (each up to 10 characters long) in large letters on the standard output.

Each argument is printed on a separate line. Note that multiple-word arguments must be enclosed in quotes in order to be printed on the same line.

EXAMPLES

Print the message "Good luck Susan" in large letters on the screen:

```
banner "Good luck" Susan
```

The words **Good luck** are displayed on one line, and **Susan** is displayed on a second line.

WARNINGS

This command is likely to be withdrawn from X/Open standards. Applications using this command might not be portable to other vendors' platforms.

SEE ALSO

echo(1).

STANDARDS CONFORMANCE

banner: SVID2, SVID3, XPG2, XPG3

NAME

basename, dirname - extract portions of path names

SYNOPSIS

basename *string* [*suffix*]

dirname [*string*]

DESCRIPTION

basename deletes any prefix ending in **/** and the *suffix* (if present in *string*) from *string*, and prints the result on the standard output. If *string* consists entirely of slash characters, *string* is set to a single slash character. If there are any trailing slash characters in *string*, they are removed. If the suffix operand is present but not identical to the characters remaining in *string*, but it is identical to a suffix of the characters remaining in *string*, the suffix is removed from *string*. **basename** is normally used inside command substitution marks (**`**...**`**) within shell procedures.

dirname delivers all but the last level of the path name in *string*. If *string* does not contain a directory component, **dirname** returns **.**, indicating the current working directory.

EXTERNAL INFLUENCES

Environment Variables

LC_CTYPE determines the interpretation of string and, in the case of basename, suffix as single and/or multi-byte characters.

If **LC_CTYPE** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**. If any internationalization variable contains an invalid setting, **basename** and **dirname** behave as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported.

EXAMPLES

The following shell script, invoked with the argument **/usr/src/cmd/cat.c**, compiles the named file and moves the output to a file named **cat** in the current directory:

```
cc $1
mv a.out ‘basename $1 .c‘
```

The following example sets the shell variable **NAME** to **/usr/src/cmd**:

```
NAME=‘dirname /usr/src/cmd/cat.c‘
```

RETURNS

basename and **dirname** return one of the following values:

0	Successful completion.
1	Incorrect number of command-line arguments.

SEE ALSO

expr(1), sh(1).

STANDARDS CONFORMANCE

basename: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

dirname: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

NAME

bc - arbitrary-precision arithmetic language

SYNOPSIS

bc [**-c**] [**-l**] [*file* ...]

DESCRIPTION

bc is an interactive processor for a language that resembles C but provides unlimited-precision arithmetic. It takes input from any files given, then reads the standard input.

Options:

bc recognizes the following command-line options:

-c	Compile only. **bc** is actually a preprocessor for **dc** which **bc** invokes automatically (see *dc*(1)). Specifying **-c** prevents invoking *dc*, and sends the *dc* input to standard output.
-l	causes an arbitrary-precision math library to be predefined. As a side effect, the scale factor is set.

Program Syntax:

L	a single letter in the range **a** through **z**;
E	expression;
S	statement;
R	relational expression.

Comments:

Comments are enclosed in **/*** and ***/**.

Names:

Names include:

simple variables: L
array elements: L [E]
The words **ibase**,**obase**, and **scale**
stacks: L

Other Operands

Other operands include:

Arbitrarily long numbers with optional sign and decimal point.

(E)

sqrt (E)

length (E)	number of significant decimal digits
scale (E)	number of digits right of decimal point

L (E , ... , E)

Strings of ASCII characters enclosed in quotes (").

Arithmetic Operators:

Arithmetic operators yield an E as a result and include:

+ - * / % ^	(**%** is remainder (not mod, see below); **^** is power).
++ --	(prefix and append; apply to names)
= += -= *= /= %= ^=	

Relational Operators

Relational operators yield an R when used as **E** *op* **E**:

== <= >= != < >

Statements

E
{ S ; ... ; S }
if (R) S
while (R) S
for (E ; R ; E) S
null statement
break
quit

Function Definitions:

```
define L ( L ,..., L ) {
          auto L, ... , L
          S; ... S
          return ( E )
}
```

Functions in –l Math Library:

Functions in the **-l** math library include:

s(x)	sine
c(x)	cosine
e(x)	exponential
l(x)	log
a(x)	arctangent
j(n,x)	Bessel function

All function arguments are passed by value. Trigonometric angles are in radians where 2 pi radians = 360 degrees.

The value of a statement that is an expression is printed unless the main operator is an assignment. No operators are defined for strings, but the string is printed if it appears in a context where an expression result would be printed. Either semicolons or new-lines can separate statements. Assignment to *scale* influences the number of digits to be retained on arithmetic operations in the manner of *dc*(1). Assignments to **ibase** or **obase** set the input and output number radix respectively, again as defined by *dc*(1).

The same letter can be used simultaneously as an array, a function, and a simple variable. All variables are global to the program. "Auto" variables are pushed down during function calls. When using arrays as function arguments or defining them as automatic variables, empty square brackets must follow the array name.

The **%** operator yields the remainder at the current scale, not the integer modulus. Thus, at scale 1, **7 % 3** is .1 (one tenth), not 1. This is because (at scale 1) **7 / 3** is 2.3 with .1 as the remainder.

EXAMPLES

Define a function to compute an approximate value of the exponential function:

```
scale = 20
define e(x){
        auto a, b, c, i, s
        a = 1
        b = 1
        s = 1
        for(i=1; 1==1; i++){
                a = a*x
                b = b*i
                c = a/b
                if(c == 0) return(s)
                s = s+c
        }
}
```

Print approximate values of the exponential function of the first ten integers.

```
for(i=1; i<=10; i++) e(i)
```

WARNINGS

There are currently no **&&** (AND) or **||** (OR) comparisons.

The **for** statement must have all three expressions.

quit is interpreted when read, not when executed.

bc's parser is not robust in the face of input errors. Some simple expression such as 2+2 helps get it back into phase.

The assignment operators: **=+ =- =* =/ =%** and **=^** are obsolete. Any occurences of these operators cause a syntax error with the exception of **=-** which is interpreted as **=** followed by a unary minus.

Neither entire arrays nor functions can be passed as function parameters.

FILES

/usr/bin/dc	desk calculator executable program
/usr/lib/lib.b	mathematical library

SEE ALSO

bs(1), dc(1).

bc tutorial in *Number Processing Users Guide*

STANDARDS CONFORMANCE

bc: XPG4, POSIX.2

NAME

bdiff - diff for large files

SYNOPSIS

bdiff *file1 file2* [*n*] [**-s**]

DESCRIPTION

bdiff compares two files and produces output identical to what would be produced by **diff** (see *diff*(1)), specifying changes that must be made to make the files identical. **bdiff** is designed for handling files that are too large for **diff**, but it can be used on files of any length.

bdiff processes files as follows:

- Ignore lines common to the beginning of both files.
- Split the remainder of each file into *n*-line segments, then execute **diff** on corresponding segments. The default value of *n* is 3500.

Command-Line Arguments

bdiff recognizes the following command-line arguments:

file1
file2 Names of two files to be compared by **bdiff**. If *file1* or *file2* (but not both) is **-**, standard input is used instead.

n If a numeric value is present as the third argument, the files are divided into *n*-line segments before processing by **diff**. Default value for *n* is 3500. This option is useful when 3500-line segments are too large for processing by **diff**.

-s Silent option suppresses diagnostic printing by **bdiff**, but does not suppress possible error messages from **diff**). If the *n* and **-s** arguments are both used, the *n* argument must precede the **-s** option on the command line or it will not be properly recognized.

EXTERNAL INFLUENCES

Environment Variables

LC_MESSAGES determines the language in which messages are displayed.

If **LC_MESSAGES** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**.

If any internationalization variable contains an invalid setting, **bdiff** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported.

DIAGNOSTICS

both files standard input (bd2)
Standard input was specified for both files. Only one file can be specified as standard input.

non-numeric limit (bd4)
A non-numeric value was specified for the *n* (third) argument.

EXAMPLES

Find differences between two large files: **file1** and **file2**, and place the result in a new file named **diffs_1.2**.

```
bdiff file1 file2 >diffs_1.2
```

Do the same, but limit file length to 1400 lines; suppress error messages:

```
bdiff file1 file2 1400 -s >diffs_1.2
```

WARNINGS

bdiff produces output identical to output from **diff**, and makes the necessary line-number corrections so that the output looks like it was processed by **diff**. However, depending on where the files are split, **bdiff** may or may not find a fully minimized set of file differences.

FILES

`/tmp/bd??????`

SEE ALSO

diff(1).

NAME

bfs - big file scanner

SYNOPSIS

bfs [**-**] *name*

DESCRIPTION

bfs is similar to **ed** except that it is read-only (see *ed*(1)) **bfs** can handle files with up to 32K – 1 lines; each line can contain up to 512 characters, including the new-line character. **bfs** is usually more efficient than **ed** for scanning a file, since the file is not copied to a buffer. Historically, this command was most useful for identifying sections of a large file where **csplit** could be used to divide it into more manageable pieces for editing (see *csplit*(1)). However, most editors now support files larger than the above-mentioned limits.

Normally, the size of the file being scanned is printed, as is the size of any file written with the **w** command. The optional **-** suppresses printing of sizes. Input is prompted with ***** if **P** and a carriage-return are typed, as in **ed**. Prompting can be turned off again by inputting another **P** and pressing Return. Note that messages are given in response to errors if prompting is turned on.

bfs supports the Basic Regular Expression (RE) syntax (see *regexp*(5)) with the addition that a null RE (e.g., **//**) is equivalent to the last RE encountered. All address expressions described under **ed** are supported. In addition, regular expressions can be surrounded with two symbols besides **/** and **?**: **>** indicates downward search without wrap-around, and **<** indicates upward search without wrap-around. There is a slight difference in mark names: only the letters **a** through **z** can be used, and all 26 marks are remembered.

The **e**, **g**, **v**, **k**, **n**, **p**, **q**, **w**, **=**, **!** and null commands operate as described under **ed**. Commands such as **---**, **+++-**, **+++=**, **-12**, and **+4p** are accepted. Note that **1,10p** and **1,10** both print the first ten lines. The **f** command only prints the name of the file being scanned; there is no *remembered* file name. The **w** command is independent of output diversion, truncation, or crunching (see the **xo**, **xt**, and **xc** commands, below). The following additional commands are available:

xf *file* Further commands are taken from the named *file*. When an end-of-file is reached, an interrupt signal is received or an error occurs, reading resumes with the file containing the **xf**. **Xf** commands may be nested to a depth of 10.

xo [*file*] Further output from the **p** and null commands is diverted to the named *file*, which, if necessary, is created mode 666. If *file* is missing, output is diverted to the standard output. Note that each diversion causes truncation or creation of the file.

: *label* This positions a *label* in a command file. *label* is terminated by a new-line, and blanks between the **:** and the start of *label* are ignored. This command can also be used to insert comments into a command file, since labels need not be referenced.

(**.,.**)**xb/***regular expression***/***label*

A jump (either upward or downward) is made to *label* if the command succeeds. It fails under any of the following conditions:

1. Either address is not between **1** and **$**.
2. The second address is less than the first.
3. The regular expression does not match at least one line in the specified range, including the first and last lines.

On success, **.** is set to the line matched and a jump is made to *label*. This command is the only one that does not issue an error message on bad addresses. Thus it can be used to test whether addresses are bad before other commands are executed. Note that the command

xb/*label*

is an unconditional jump.

The **xb** command is allowed only if it is read from someplace other than a terminal. If it is read from a pipe only a downward jump is possible.

xn List the marks currently in use (marks are set by the **k** command).

xt *number* Output from the **p** and null commands is truncated to at most *number* characters. The initial number is 255.

xv[*digit*][*spaces*][*value*]

The variable name is the specified *digit* following the **xv**. **xv5100** or **xv5 100** both assign the value **100** to the variable **5**. **Xv61,100p** assigns the value **1,100p** to the variable **6**. To reference a variable, put a **%** in front of the variable name. For example, using the above assignments for variables **5** and **6**:

```
1,%5p
1,%5
%6
```

all print the first 100 lines.

```
g/%5/p
```

globally searches for the characters **100** and prints each line containing a match. To escape the special meaning of **%**, a **** must precede it. For example, to match and list lines in a program file that contain **printf()** format strings specifying characters, decimal integers, or strings, the following could be used:

```
g/".*\%[cds]/p
```

Another feature of the **xv** command is that the first line of output from an HP-UX command can be stored into a variable. The only requirement is that the first character of *value* be an **!**. For example:

```
.w junk
xv5!cat junk
!rm junk
!echo "%5"
xv6!expr %6 + 1
```

each put the current line into variable **5**, print it, and increment the variable **6** by one. To escape the special meaning of **!** as the first character of *value*, precede it with a ****.

```
xv7\!date
```

stores the value **!date** into variable **7**.

xbz *label*
xbn *label*

These two commands test the last saved *return code* from the execution of an HP-UX system command (**!***command*) for a zero or non-zero value, respectively, and cause a branch to the specified label. The two examples below both search for the next five lines containing the string **size**.

First example:

```
xv55
: l
/size/
xv5!expr %5 - 1
!if [ %5 != 0 ] ; then exit 2 ; fi
xbn l
```

Second Example:

```
xv45
: l
/size/
xv4!expr %4 - 1
!if [ %4 = 0 ] ; then exit 2 ; fi
xbz l
```

xc [*switch*]

If *switch* is **1**, output from the **p** and null commands is crunched; if *switch* is **0** it isn't. Without an argument, **xc** reverses *switch*. Initially *switch* is set for no crunching. Crunched output has strings of tabs and blanks reduced to one blank, and blank lines suppressed.

EXTERNAL INFLUENCES

Environment Variables

LC_COLLATE determines the collating sequence used in evaluating regular expressions.

LC_CTYPE determines the classification of characters as letters, and the characters matched by character class expressions in regular expressions.

If **LC_COLLATE** or **LC_CTYPE** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**. If any internationalization variable contains an invalid setting, **bfs** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single-byte character code sets are supported.

DIAGNOSTICS

? for errors in commands, if prompting is turned off. Self-explanatory error messages when prompting is on.

SEE ALSO

csplit(1), ed(1), lang(5), regexp(5).

NAME

bs - a compiler/interpreter for modest-sized programs

SYNOPSIS

bs [*file* [*args*]]

DESCRIPTION

bs is a remote descendant of BASIC and SNOBOL4 with some C language added. **bs** is designed for programming tasks where program development time is as important as the resulting speed of execution. Formalities of data declaration and file/process manipulation are minimized. Line-at-a-time debugging, the **trace** and **dump** statements, and useful run-time error messages all simplify program testing. Furthermore, incomplete programs can be debugged; *inner* functions can be tested before *outer* functions have been written, and vice versa.

If *file* is specified on the command-line, it is used for input before any input is taken from the keyboard. By default, statements read from *file* are compiled for later execution. Likewise, statements entered from the keyboard are normally executed immediately (see **compile** and **execute** below). Unless the final operation is assignment, the result of an immediate expression statement is printed.

bs programs are made up of input lines. If the last character on a line is a ****, the line is continued. **bs** accepts lines of the following form:

statement
label statement

A label is a *name* (see below) followed by a colon. A label and a variable can have the same name.

A **bs** statement is either an expression or a keyword followed by zero or more expressions. Some keywords (**clear**, **compile**, **!**, **execute**, **include**, **ibase**, **obase**, and **run**) are always executed as they are compiled.

Statement Syntax:

expression The expression is executed for its side effects (value, assignment, or function call). The details of expressions follow the description of statement types below.

break **break** exits from the innermost **for**/**while** loop.

clear Clears the symbol table and compiled statements. **clear** is executed immediately.

compile [*expression*]
Succeeding statements are compiled (overrides the immediate execution default). The optional expression is evaluated and used as a file name for further input. A **clear** is associated with this latter case. **compile** is executed immediately.

continue **continue** transfers to the loop-continuation of the current **for**/**while** loop.

dump [*name*] The name and current value of every non-local variable is printed. Optionally, only the named variable is reported. After an error or interrupt, the number of the last statement is displayed. The user-function trace is displayed after an error or **stop** that occurred in a function.

edit A call is made to the editor selected by the **EDITOR** environment variable if it is present, or *ed*(1) if **EDITOR** is undefined or null. If the *file* argument is present on the command line, *file* is passed to the editor as the file to edit (otherwise no file name is used). Upon exiting the editor, a **compile** statement (and associated **clear**) is executed giving that file name as its argument.

exit [*expression*]
Return to system level. The expression is returned as process status.

execute Change to immediate execution mode (an interrupt has a similar effect). This statement does not cause stored statements to execute (see **run** below).

for *name* **=** *expression expression statement*
for *name* **=** *expression expression*
...
next

for *expression* **,** *expression* **,** *expression statement*
for *expression* **,** *expression* **,** *expression*
...
next The **for** statement repetitively executes a statement (first form) or a group of statements (second form) under control of a named variable. The variable takes on the value of the first expression, then is incremented by one on each loop, not to exceed the value of the second expression. The third and fourth forms require three expressions separated by commas. The first of these is the initialization, the second is the test (true to continue), and the third is the loop-continuation action (normally an increment).

fun *f* **(** [*a*, ...] **)** [v, ...]
...
nuf **fun** defines the function name, arguments, and local variables for a user-written function. Up to ten arguments and local variables are allowed. Such names cannot be arrays, nor can they be I/O associated. Function definitions cannot be nested. Calling an undefined function is permissible; see function calls below.

freturn A way to signal the failure of a user-written function. See the interrogation operator (**?**) below. If interrogation is not present, **freturn** merely returns zero. When interrogation *is* active, **freturn** transfers to that expression (possibly by-passing intermediate function returns).

goto *name* Control is passed to the internally stored statement with the matching label.

ibase *n* **ibase** sets the input base (radix) to *n*. The only supported values for *n* are the constants **8**, **10** (the default), and **16**. Hexadecimal values 10-15 are entered as **a**-**f**. A leading digit is required (i.e., **f0a** must be entered as **0f0a**). **ibase** (and **obase** discussed below) are executed immediately.

if *expression statement*
if *expression*
...
[**else**...]
fi The statement (first form) or group of statements (second form) is executed if the expression evaluates to non-zero. The strings **0** and "" (null) evaluate as zero. In the second form, an optional **else** provides for a second group of statements to be executed when the first group is not. The only statement permitted on the same line with an **else** is an **if**; only other **fi**s can be on the same line with a **fi**. The concatenation of **else** and **if** into an **elif** is supported. Only a single **fi** is required to close an **if** ... **elif** ... [**else** ...] sequence.

include *expression*
expression must evaluate to a file name. The file must contain **bs** source statements. Such statements become part of the program being compiled. **include** statements cannot be nested.

obase *n* **obase** sets the output base to *n* (see **ibase** above).

onintr *label*
onintr **onintr** provides program control of interrupts. In the first form, control passes to the label given, just as if a **goto** had been executed at the time **onintr** was executed. The effect of the statement is cleared after each interrupt. In the second form, an interrupt causes **bs** to terminate.

return [*expression*]
The expression is evaluated and the result is passed back as the value of a function call. If no expression is given, zero is returned.

run The random number generator is reset. Control is passed to the first internal statement. If the **run** statement is contained in a file, it should be the last statement.

`stop` Execution of internal statements is stopped. **`bs`** reverts to immediate mode.

`trace` [*expression*]
The **`trace`** statement controls function tracing. If the expression is null (or evaluates to zero), tracing is turned off. Otherwise, a record of user-function calls/returns is printed. Each **`return`** decrements the **`trace`** *expression* value.

`while` *expression statement*
`while` *expression*
...
`next` **`while`** is similar to **`for`** except that only the conditional expression for loop-continuation is given.

`!` *shell command*
An immediate escape to the shell.

`#` ... This statement is ignored (treated as a comment).

Expression Syntax:

name A name is used to specify a variable. Names are composed of a letter (uppercase or lowercase) optionally followed by letters and digits. Only the first six characters of a name are significant. Except for names declared in *fun* statements, all names are global to the program. Names can take on numeric (double float) values, string values, or can be associated with input/output (see the built-in function **`open( )`** below).

name ([expression [, expression] ...])
Functions can be called by a name followed by the arguments in parentheses separated by commas. Except for built-in functions (listed below), the name must be defined with a *fun* statement. Arguments to functions are passed by value. If the function is undefined, the call history to the call of that function is printed, and a request for a return value (as an expression) is made. The result of that expression is taken to be the result of the undefined function. This permits debugging programs where not all the functions are yet defined. The value is read from the current input file.

name [expression [, expression] ...]
This syntax is used to reference either arrays or tables (see built-in *table* functions below). For arrays, each expression is truncated to an integer and used as a specifier for the name. The resulting array reference is syntactically identical to a name; **`a[1,2]`** is the same as **`a[1][2]`**. The truncated expressions are restricted to values between 0 and 32 767.

number A number is used to represent a constant value. A number is written in Fortran style, and contains digits, an optional decimal point, and possibly a scale factor consisting of an **`e`** followed by a possibly signed exponent.

string Character strings are delimited by " characters. The **`\`** escape character allows the double quote (**`\`**"), new-line (**`\n`**), carriage return (**`\r`**), backspace (\b), and tab (**`\t`**) characters to appear in a string. Otherwise, **`\`** stands for itself.

(expression) Parentheses are used to alter the normal order of evaluation.

(expression , expression [, expression ...]) [expression]
The bracketed expression is used as a subscript to select a comma-separated expression from the parenthesized list. List elements are numbered from the left, starting at zero.

The expression:

```
    ( False, True )[ a == b ]
```

has the value **`True`** if the comparison is true.

`?` expression The interrogation operator tests for the success of the expression rather than its value. At the moment, it is useful for testing end-of-file (see examples in the *Programming Tips* section below), the result of the **`eval`** built-in function, and for checking the return from user-written functions (see **`freturn`**). An interrogation "trap" (end-of-file, etc.) causes an immediate transfer to the most recent interrogation, possibly skipping assignment statements or intervening function levels.

`-` expression The result is the negation of the expression.

++ name	Increments the value of the variable (or array reference). The result is the new value.
-- name	Decrements the value of the variable. The result is the new value.
!expression	The logical negation of the expression. Watch out for the shell escape command.
expression	*operator* expression Common functions of two arguments are abbreviated by the two arguments separated by an operator denoting the function. Except for the assignment, concatenation, and relational operators, both operands are converted to numeric form before the function is applied.

Binary Operators (in increasing precedence):

=	**=** is the assignment operator. The left operand must be a name or an array element. The result is the right operand. Assignment binds right to left, all other operators bind left to right.
_	_ (underscore) is the concatenation operator.
& |	**&** (logical AND) has result zero if either of its arguments are zero. It has result one if both of its arguments are non-zero; **|** (logical OR) has result zero if both of its arguments are zero. It has result one if either of its arguments is non-zero. Both operators treat a null string as a zero.
< <= > >= == !=	The relational operators (**<**: less than, **<=**: less than or equal, **>**: greater than, **>=**: greater than or equal, **==**: equal to, **!=**: not equal to) return one if their arguments are in the specified relation, or return zero otherwise. Relational operators at the same level extend as follows: **a>b>c** is equivalent to *a*>*b* & *b*>*c*. A string comparison is made if both operands are strings.
+ -	Add and subtract.
*** / %**	Multiply, divide, and remainder.
^	Exponentiation.

Built-in Functions:

Dealing with arguments

arg(*i***)**	is the value of the *i*-th actual parameter on the current level of function call. At level zero, *arg* returns the *i*-th command-line argument (*arg*(0) returns **bs**).
narg( )	returns the number of arguments passed. At level zero, the command argument count is returned.

Mathematical

abs(*x***)**	is the absolute value of *x*.
atan(*x***)**	is the arctangent of *x*. Its value is between $-\pi/2$ and $\pi/2$.
ceil(*x***)**	returns the smallest integer not less than *x*.
cos(*x***)**	is the cosine of *x* (radians).
exp(*x***)**	is the exponential function of *x*.
floor(*x***)**	returns the largest integer not greater than *x*.
log(*x***)**	is the natural logarithm of *x*.
rand( )	is a uniformly distributed random number between zero and one.
sin(*x***)**	is the sine of *x* (radians).
sqrt(*x***)**	is the square root of *x*.

String operations

size(*s***)**	the size (length in bytes) of *s* is returned.
format(*f***,** *a***)**	returns the formatted value of *a*. *f* is assumed to be a format specification in the style of *printf*(3S). Only the **%...f**, **%...e**, and **%...s** types are safe. Since it is not always

possible to know whether **a** is a number or a string when the **format** call is coded, coercing **a** to the type required by **f** by either adding zero (for **e** or **f** format) or concatenating (_) the null string (for **s** format) should be considered.

index(*x*, *y***)**
returns the number of the first position in *x* that any of the characters from *y* matches. No match yields zero.

trans(*s*, *f*, *t***)**
Translates characters of the source *s* from matching characters in *f* to a character in the same position in *t*. Source characters that do not appear in *f* are copied to the result. If the string *f* is longer than *t*, source characters that match in the excess portion of *f* do not appear in the result.

substr(*s*, *start*, *width***)**
returns the sub-string of *s* defined by the *start*ing position and *width*.

match(*string*, *pattern***)**

mstring(*n***)** The *pattern* is a regular expression according to the Basic Regular Expression definition (see *regexp*(5)). **mstring** returns the *n*-th (1 <= *n* <= 10) substring of the subject that occurred between pairs of the pattern symbols **\(** and **\)** for the most recent call to *match*. To succeed, patterns must match the beginning of the string (as if all patterns began with ^). The function returns the number of characters matched. For example:

```
match("a123ab123", ".*\([a-z]\)") == 6
mstring(1) == "b"
```

File handling

open(*name*, *file*, *function***)**
close(*name***)**
name argument must be a **bs** variable name (passed as a string). For the **open**, the *file* argument can be:

1. a 0 (zero), 1, or 2 representing standard input, output, or error output, respectively;
2. a string representing a file name; or
3. a string beginning with an **!** representing a command to be executed (via **sh -c**). The *function* argument must be either **r** (read), **w** (write), **W** (write without new-line), or **a** (append). After a **close**, *name* reverts to being an ordinary variable. If *name* was a pipe, a **wait()** is executed before the close completes (see *wait*(2)). The **bs exit** command does not do such a wait. The initial associations are:

```
open("get", 0, "r")
open("put", 1, "w")
open("puterr", 2, "w")
```

Examples are given in the following section.

access(*s*, *m***)**
executes **access()** (see *access*(2)).

ftype(*s***)** returns a single character file type indication: **f** for regular file, **p** for FIFO (i.e., named pipe), **d** for directory, **b** for block special, or **c** for character special.

Tables

table(*name*, *size***)**
A table in **bs** is an associatively accessed, single-dimension array. "Subscripts" (called keys) are strings (numbers are converted). The *name* argument must be a **bs** variable name (passed as a string). The *size* argument sets the minimum number of elements to be allocated. **bs** prints an error message and stops on table overflow. The result of *table* is *name*.

item(*name*, *i***)**
key() The **item** function accesses table elements sequentially (in normal use, there is no orderly progression of key values). Where the **item** function accesses values, the **key** function accesses the "subscript" of the previous **item** call. It fails (or in the absence of an

interrogate operator, returns null) if there was no valid subscript for the previous **item** call. The *name* argument should not be quoted. Since exact table sizes are not defined, the interrogation operator should be used to detect end-of-table; for example:

```
table("t", 100)
    ...
# If word contains "party", the following expression adds one
# to the count of that word:
++t[word]
    ...
# To print out the the key/value pairs:
for i = 0, ?(s = item(t, i)), ++i if key() put = key()_":"_s
```

If the interrogation operator is not used, the result of **item** is null if there are no further elements in the table. Null is, however, a legal "subscript".

iskey(*name*, *word***)**
iskey tests whether the key *word* exists in the table *name* and returns one for true, zero for false.

Odds and ends

eval(*s***)** The string argument is evaluated as a **bs** expression. The function is handy for converting numeric strings to numeric internal form. **eval** can also be used as a crude form of indirection, as in:

```
    name = "xyz" eval("++"_ name)
```

which increments the variable *xyz*. In addition, **eval** preceded by the interrogation operator permits the user to control **bs** error conditions. For example:

```
    ?eval("open(\"x\", \"XXX\", \"r\")")
```

returns the value zero if there is no file named **XXX** (instead of halting the user's program). The following executes a **goto** to the label **L** (if it exists):

```
    label="L"
    if !(?eval("goto "_ label)) puterr = "no label"
```

plot(*request*, *args***)**
If the **tplot** command is available, the **plot** function produces output on devices recognized by **tplot**. The *requests* are as follows:

Call	**Function**
plot(0, *term***)**	causes further *plot* output to be piped into *tplot* with an argument of **-T***term*. *term* can be up to 40 characters in length.
plot(1)	"erases" the plotter.
plot(2, *string***)**	labels the current point with *string*.
plot(3, *x1, y1, x2, y2***)**	draws the line between (*x1*,*y1*) and (*x2*,*y2*).
plot(4, *x, y, r***)**	draws a circle with center (*x*,*y*) and radius *r*.
plot(5, *x1, y1, x2, y2, x3, y3***)**	draws an arc (counterclockwise) with center (*x1*,*y1*) and endpoints (*x2*,*y2*) and (*x3*,*y3*).
plot(6)	is not implemented.
plot(7, *x, y***)**	makes the current point (*x*,*y*).
plot(8, *x, y***)**	draws a line from the current point to (*x*,*y*).
plot(9, *x*, *y***)**	draws a point at (*x*,*y*).
plot(10, *string***)**	sets the line mode to *string*.
plot(11, *x1, y1, x2, y2***)**	makes (*x1*,*y1*) the lower left corner of the plotting area and (*x2*,*y2*) the upper right corner of the plotting area.

plot(12, *x1*, *y1*, *x2*, *y2***)** causes subsequent x (y) coordinates to be multiplied by *x1* (*y1*) and then added to *x2* (*y2*) before they are plotted. The initial scaling is **plot(12, 1.0, 1.0, 0.0, 0.0)**.

Some requests do not apply to all plotters. All requests except zero and twelve are implemented by piping characters to *tplot*.

Each statement executed from the keyboard re-invokes **tplot**, making the results unpredictable if a complete picture is not done in a single operation. Plotting should thus be done either in a function or a complete program, so all the output can be directed to **tplot** in a single stream.

last() in immediate mode, **last** returns the most recently computed value.

EXTERNAL INFLUENCES

Environment Variables

LC_COLLATE determines the collating sequence used in evaluating regular expressions.

LC_CTYPE determines the characters matched by character class expressions in regular expressions.

If **LC_COLLATE** or **LC_CTYPE** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**. If any internationalization variable contains an invalid setting, **bs** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single-byte character code sets are supported.

EXAMPLES

Using **bs** as a calculator (**$** is the shell prompt):

```
$ bs
# Distance (inches) light travels in a nanosecond.
186000 * 5280 * 12 / 1e9
11.78496
   ...
# Compound interest (6% for 5 years on $1,000).
int = .06 / 4
bal = 1000
for i = 1 5*4 bal = bal + bal*int
bal - 1000
346.855007
   ...
exit
```

The outline of a typical **bs** program:

```
# initialize things:
var1 = 1
open("read", "infile", "r")
   ...
# compute:
while ?(str = read)
   ...
next
# clean up:
close("read")
   ...
# last statement executed (exit or stop):
exit
# last input line:
run
```

Input/Output examples:

```
# Copy file oldfile to file newfile.
open("read", "oldfile", "r")
open("write", "newfile", "w")
   ...
while ?(write = read)
   ...
# close "read" and "write":
close("read")
close("write")

# Pipe between commands.
open("ls", "!ls *", "r")
open("pr", "!pr -2 -h 'List'", "w")
while ?(pr = ls) ...
   ...
# be sure to close (wait for) these:
close("ls")
close("pr")
```

WARNINGS

The graphics mode (**plot** ...) is not particularly useful unless the **tplot** command is available on your system.

bs is not tolerant of some errors. For example, mistyping a **fun** declaration is difficult to correct because a new definition cannot be made without doing a **clear**. The best solution in such a case is to start by using the **edit** command.

SEE ALSO

ed(1), sh(1), access(2), printf(3S), stdio(3S), lang(5), regexp(5).

See Section (3M) for a further description of the mathematical functions.

pow() is used for exponentiation — see *exp*(3M));

bs uses the Standard I/O package.

NAME

cal - print calendar

SYNOPSIS

cal [[*month*] *year*]

DESCRIPTION

cal prints a calendar for the specified year. If a month is also specified, a calendar just for that month is printed. If neither is specified, a calendar for the present month is printed. *year* can be between 1 and 9999. *month* is a decimal number between 1 and 12. The calendar produced is a Gregorian calendar.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the locale to use for the locale categories when both **LC_ALL** and the corresponding environment variable (beginning with **LC_**) do not specify a locale. If **LANG** is not set or is set to the empty string, a default of "C" (see *lang*(5)) is used.

LC_CTYPE determines the locale for interpretation of sequences of bytes of text data as characters (e.g., single- verses multibyte characters in arguments and input files).

LC_TIME determines the format and contents of the calendar.

TZ determines the timezone used to calculate the value of the current month.

If any internationalization variable contains an invalid setting, **cal** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported.

EXAMPLES

The command:

```
cal 9 1850
```

prints the calendar for September, 1850 on the screen as follows:

```
          September 1850
  S     M    Tu     W    Th     F     S
  1     2     3     4     5     6     7
  8     9    10    11    12    13    14
 15    16    17    18    19    20    21
 22    23    24    25    26    27    28
 29    30
```

However, for XPG4 the output looks like below:

```
                     Sep 1850
 Sun   Mon   Tue   Wed   Thu   Fri   Sat
  1     2     3     4     5     6     7
  8     9    10    11    12    13    14
 15    16    17    18    19    20    21
 22    23    24    25    26    27    28
 29    30
```

WARNINGS

The year is always considered to start in January even though this is historically naive.

Beware that **cal 83** refers to the early Christian era, not the 20th century.

STANDARDS CONFORMANCE

cal: SVID2, SVID3, XPG2, XPG3, XPG4

NAME

calendar - reminder service

SYNOPSIS

calendar [**-**]

DESCRIPTION

calendar consults the file **calendar** in the current directory and prints out lines containing today's or tomorrow's date anywhere in the line. On weekends, "tomorrow" extends through Monday.

When a **-** command-line argument is present, **calendar** searches for the file **calendar** in each user's home directory, and sends any positive results to the user by **mail** (see *mail*(1)). Normally this is done daily in the early morning hours under the control of **cron** (see *cron*(1M)). When invoked by **cron**, **calendar** reads the first line in the **calendar** file to determine the user's environment.

Language-dependent information such as spelling and date format (described below) are determined by the user-specified **LANG** statement in the **calendar** file. This statement should be of the form **LANG=***language* where *language* is a valid language name (see *lang*(5)). If this line is not in the **calendar** file, the action described in the EXTERNAL INFLUENCES Environment Variable section is taken.

calendar is concerned with two fields: month and day. A month field can be expressed in three different formats: a string representing the name of the month (either fully spelled out or abbreviated), a numeric month, or an asterisk (representing any month). If the month is expressed as a string representing the name of the month, the first character can be either upper-case or lower-case; other characters must be lower-case. The spelling of a month name should match the string returned by calling **nl_langinfo()** (see *nl_langinfo*(3C)). The day field is a numeric value for the day of the month.

Month-Day Formats

If the month field is a string, it can be followed by zero or more blanks. If the month field is numeric, it must be followed by either a slash (**/**) or a hyphen (**-**). If the month field is an asterisk (*****), it must be followed by a slash (**/**). The day field can be followed immediately by a blank or non-digit character.

Day-Month Formats

The day field is expressed as a numeral. What follows the day field is determined by the format of the month. If the month field is a string, the day field must be followed by zero or one dot (**.**) followed by zero or more blanks. If the month field is a numeral, the day field must be followed by either a slash (**/**) or a hyphen (**-**). If the month field is an asterisk, the day field must be followed by a slash (**/**).

EXTERNAL INFLUENCES

Environment Variables

LC_TIME determines the format and contents of date and time strings when no **LANG** statement is specified in the **calendar** file.

LANG determines the language in which messages are displayed.

If **LC_TIME** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**. If any internationalization variable contains an invalid setting, **calendar** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported.

EXAMPLES

The following **calendar** file illustrates several formats recognized by *calendar*:

```
LANG=en_US.roman8
Friday, May 29th: group coffee meeting
meeting with Boss on June 3.
3/30/87 - quarter end review
4-26 Management council meeting at 1:00 pm
It is first of the month ( */1 ); status report due.
```

In the following **calendar** file, dates are expressed according to European English usage:

```
LANG=en_GB.roman8
On 20 Jan. code review
```

```
Jim’s birthday is on the 3. February
30/3/87 - quarter end review
26-4 Management council meeting at 1:00 pm
It is first of the month ( 1/* ); status report due.
```

WARNINGS

To get reminder service, either your calendar must be public information or you must run **calendar** from your personal **crontab** file, independent of any **calendar -** run systemwide. Note that if you run **calendar** yourself, the calendar file need not reside in your home directory.

calendar’s extended idea of “tomorrow” does not account for holidays.

This command is likely to be withdrawn from X/Open standards. Applications using this command might not be portable to other vendors’ platforms.

AUTHOR

calendar was developed by AT&T and HP.

FILES

calendar
/tmp/cal*
/usr/lbin/calprog to figure out today’s and tomorrow’s dates
/usr/bin/crontab
/etc/passwd

SEE ALSO

cron(1M), nl_langinfo(3C), mail(1), environ(5).

STANDARDS CONFORMANCE

calendar: SVID2, SVID3, XPG2, XPG3

NAME

cat - concatenate, copy, and print files

SYNOPSIS

cat [**-benrstuv**] *file* ...

DESCRIPTION

cat reads each *file* in sequence and writes it on the standard output. Thus:

cat *file*

prints *file* on the default standard output device;

cat *file1 file2* **>** *file3*

concatenates *file1* and *file2*, and places the result in *file3*.

If **-** is appears as a *file* argument, **cat** uses standard input. To combine standard input and other files, use a combination of **-** and *file* arguments.

Options

cat recognizes the following options:

-b Omit line numbers from blank lines when **-n** option is specified. If this option is specified, the **-n** option is automatically selected.

-e Print a **$** character at the end of each line (prior to the new-line). If this option is specified, the **-v** option is automatically selected.

-n Display output lines preceded by line numbers, numbered sequentially from 1.

-r Replace multiple consecutive empty lines with one empty line, so that there is never more than one empty line between lines containing characters.

-s Silent option. **cat** suppresses error messages about non-existent files, identical input and output, and write errors. Normally, input and output files cannot have identical names unless the file is a special file.

-t Print each tab character as **^I** and form feed character as **^L**. If this option is specified, the **-v** option is automatically selected.

-u Do not buffer output (handle character-by-character). Normally, output is buffered.

-v Cause non-printing characters (with the exception of tabs, new-lines and form-feeds) to be printed visibly. Control characters are printed using the form **^***X* (Ctrl-*X*), and the DEL character (octal 0177) is printed as **^?** (see *ascii*(5)). Single-byte control characters whose most significant bit is set, are printed using the form **M-^***x*, where *x* is the character specified by the seven low order bits. All other non-printing characters are printed as **M-***x*, where *x* is the character specified by the seven low order bits. This option is influenced by the **LC_CTYPE** environment variable and its corresponding code set.

EXTERNAL INFLUENCES

Environment Variables

LANG provides a default value for the internationalization variables that are unset or null. If **LANG** is unset or null, the default value of "C" (see *lang*(5)) is used. If any of the internationalization variables contains an invalid setting, **cat** will behave as if all internationalization variables are set to "C". See *environ*(5).

LC_ALL If set to a non-empty string value, overrides the values of all the other internationalization variables.

LC_CTYPE determines the interpretation of text as single and/or multi-byte characters, the classification of characters as printable, and the characters matched by character class expressions in regular expressions.

LC_MESSAGES determines the locale that should be used to affect the format and contents of diagnostic messages written to standard error and informative messages written to standard output.

NLSPATH determines the location of message catalogues for the processing of **LC_MESSAGES**.

International Code Set Support

Single- and multi-byte character code sets are supported.

RETURN VALUE

Exit values are:

0	Successful completion.
>0	Error condition occurred.

EXAMPLES

To create a zero-length file, use any of the following:

cat */dev/null* **>** *file*
cp */dev/null file*
touch *file*

The following prints **^I** for all the occurrences of tab character in *file1*

cat -t *file1*

To suppress error messages about files that do not exist, use:

cat -s *file1 file2 file3* **>** *file*

If *file2* does not exist, the above command concatenates *file1* and *file3* without reporting the error on *file2*. The result is the same if **-s** option is not used, except that **cat** displays the error message.

To view non-printable characters in *file2*, use:

cat -v *file2*

WARNINGS

Command formats such as

cat *file1 file2* **>** *file1*

overwrites the data in *file1* before the concatenation begins, thus destroying the file. Therefore, be careful when using shell special characters.

SEE ALSO

cp(1), more(1), pg(1), pr(1), rmnl(1), ssp(1).

STANDARDS CONFORMANCE

cat: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

(Bundled C Compiler - Limited Functionality)

NAME

cc - bundled C compiler

SYNOPSIS

cc [*options*] *files*

DESCRIPTION

This manual page describes the Bundled C compiler. See *cc*(1), online only, for a description of the ANSI-compliant HP-UX manual page.

This **cc** accepts several types of arguments as *files*:

.c Suffix
: Arguments whose names end with **.c** are understood to be C source files. Each is compiled and the resulting object file is left in a file having the corresponding base name, **.o** instead of **.c**. However, if a single C file is compiled and linked, all in one step, the **.o** file is deleted.

.s Suffix
: Arguments whose names end with **.s** are understood to be assembly source files and are assembled, producing a **.o** file for each **.s** file.

.i Suffix
: Arguments whose names end with **.i** are assumed to be the output of **cpp** (see the **-P** option below). They are compiled without invoking **cpp** (see *cpp*(1)). Each object file is left in a file having the corresponding base name, but suffixed with **.o** instead of **.i**.

-l*x* Form
: Arguments of the form **-l***x* cause the linker to search the library **lib***x***.sl** or **lib***x***.a** in an attempt to resolve currently unresolved external references. Because a library is searched when its name is encountered, placement of a **-l** is significant. If a file contains an unresolved external reference, the library containing the definition must be placed *after* the file on the command line. See *ld*(1) for further details.

-l:lib*x***.***suffix* Form
: Arguments of the form **-l:lib***x***.***suffix* cause the linker to search the library **lib***x***.sl** or **lib***x***.a** (depending on *suffix*) in an attempt to resolve currently unresolved external references. It is similar to the **-l** option except the current state of the **-Wl,-a** option is not important.

Other Suffixes
: All other arguments, such as those whose names end with **.o** or **.a**, are taken to be relocatable object files that are to be included in the link operation.

Arguments and options can be passed to the compiler through the **CCOPTS** environment variable as well as on the command line. The compiler reads the value of **CCOPTS** and divides these options into two sets; options that appear before a vertical bar (**|**), and options that appear after the vertical bar. The first set of options are placed before any of the command-line parameters to **cc**; the second set of options are placed after the command-line parameters to **cc**. If the vertical bar is not present, all options are placed before the command-line parameters. For example (in *sh*(1) notation),

```
CCOPTS="-v | -lmalloc"
export CCOPTS
cc -w prog.c
```

is equivalent to

```
cc -v -w prog.c -lmalloc
```

When set, the **TMPDIR** environment variable specifies a directory to be used by the compiler for temporary files, overriding the default directory **/var/tmp**.

Options

The following options are the only options which are recognized by the bundled C compiler.

-c
: Suppress the link edit phase of the compilation, and force an object (**.o**) file to be produced for each **.c** file, even if only one program is compiled. Object files produced from C programs must be linked before being executed.

-C
: Prevent the preprocessor from stripping C-style comments. See *cpp*(1) for details.

-D*name*=*def*
-D*name* Define *name* to the preprocessor, as if by '#define'. See *cpp*(1) for details.

-E Run only **cpp** on the named C or assembly files, and send the result to the standard output.

-I*dir* Change the algorithm used by the preprocessor for finding include files to also search in directory *dir*. See *cpp*(1) for details.

-l*x* Refer to the **DESCRIPTION** section.

-L *dir* Change the algorithm used by the linker to search for **lib***x***.sl** or **lib***x***.a**. The **-L** option causes **cc** to search in *dir* before searching in the default locations. See *ld*(1) for details.

-o*outfile* Name the output file from the linker *outfile*. The default name is **a.out**.

-P Run only **cpp** on the named C files and leave the result on corresponding files suffixed **.i**. The **-P** option is also passed along to **cpp**.

+R*num* Allow only the first *num* **register** variables to actually have the **register** class. Use this option when the register allocator issues the message:

```
out of general registers
```

-s Cause the output of the linker to be stripped of symbol table information. See *strip*(1) for more details. The use of this option prevents the use of a symbolic debugger on the resulting program. See *ld*(1) for more details.

-S Compile the named C files, and leave the assembly language output on corresponding files suffixed **.s**.

-t*x***,***name* Substitute subprocess *x* with *name* where *x* is one or more of a set of identifiers indicating the subprocess(es). This option works in two modes: 1) if *x* is a single identifier, *name* represents the full path name of the new subprocess; 2) if *x* is a set of identifiers, *name* represents a prefix to which the standard suffixes are concatenated to construct the full path names of the new subprocesses.

The *x* can take one or more of the values:

p	Preprocessor (standard suffix is **cpp**)
c	Compiler (standard suffix is **ccom**)
a	Assembler (standard suffix is **as**)
l	Linker (standard suffix is **ld**)

-U*name* Remove any initial definition of *name* in the preprocessor. See *cpp*(1) for details.

-v Enable verbose mode, which produces a step-by-step description of the compilation process on the standard error.

-V Cause each invoked subprocess to print its version information to stdout.

-w Suppress warning messages.

-W*x***,***arglist* Pass the comma-separated argument(s) in *arglist* to subprocess *x*. The **-W** option specification allows additional, implementation-specific options to be recognized by the compiler driver. For example,

```
-Wl,-a,archive
```

causes the linker to link with archive libraries instead of with shared libraries. See *ld*(1) for details.

The *x* can assume one of the following values:

p	Preprocessor
a	Assembler
l	Linker

Any other options encountered generate a warning to standard error.

Other arguments are assumed to be C-compatible object programs, typically produced by an earlier **cc** run, or perhaps libraries of C-compatible routines. These programs, together with the results of any compilations specified, are linked (in the order given) to produce an executable program with the name **a.out**.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages are displayed.

If **LC_CTYPE** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of **C** (see *lang*(5)) is used. If any internationalization variable contains an invalid setting, **cc** behaves as if all internationalization variables are set to **C**. See *environ*(5).

International Code Set Support

Single byte and multibyte character code sets are supported.

DIAGNOSTICS

The diagnostics produced by C itself are intended to be self-explanatory. Occasionally, messages may be produced by the preprocessor, assembler, or the link editor.

If any errors occur before **cc** is completed, a nonzero value is returned. Otherwise, zero is returned.

EXAMPLES

The example below compiles the C file **prog.c** to create a **prog.o** file, then invokes the **ld** link editor to link **prog.o** and **procedure.o** with all of the C startup routines in **/usr/ccs/lib/crt0.o** and library routines from the C library **libc.sl** or **libc.a**. The resulting executable program is placed in file **prog**:

```
cc prog.c procedure.o -o prog
```

WARNINGS

Options not recognized by **cc** are not passed on to the link editor. The option **-Wl,***arg* can be used to pass any such option to the link editor.

By default, the return value from a C program is completely random. The only two guaranteed ways to return a specific value are to explicitly call **exit()** (see *exit*(2)) or leave the function **main()** with a **return** *expression* **;** construct.

FILES

file.c	Input file
file.o	Object file
a.out	Linked executable output file
/var/tmp/ctm*	Temporary files used by the compiler
/usr/ccs/bin/as	Assembler (see *as*(1))
/usr/ccs/bin/ld	Link editor (see *ld*(1))
/usr/ccs/lib/crt0.o	Runtime startup
/usr/lib/libc.a	Standard C library (archive version), see *HP-UX Reference* Section (3)
/usr/lib/libc.sl	Standard C library (shared version), see *HP-UX Reference* Section (3)
/usr/include	Standard directory for **#include** files

Bundled C Compiler Files

/usr/ccs/bin/cc	C driver
/usr/ccs/lbin/ccom	C compiler
/usr/lib/nls/msg/$LANG/cc.cat	C compiler message catalog
/usr/ccs/lbin/cpp	C preprocessor

SEE ALSO

System Tools

as(1)	Translate assembly code to machine code.
cpp(1)	Invoke the C language preprocessor.
ld(1)	Invoke the link editor.
cc(1)	The ANSI-compliant C compiler on HP-UX.

Miscellaneous

matherr(3M)	Trap math errors.
fpgetround(3M)	Floating-point mode control functions.
strip(1)	Strip symbol and line number information from an object file.
crt0(3)	Execution startup routine.
end(3C)	Symbol of the last locations in program.
exit(2)	Termination of a process.

Tutorials and Standards Documents

B. W. Kernighan and D. M. Ritchie, *The C Programming Language*, Prentice-Hall, 1978.

NAME

cd - change working directory

SYNOPSIS

cd [*directory*]

DESCRIPTION

If *directory* is not specified, the value of shell parameter **HOME** is used as the new working directory. If *directory* specifies a complete path starting with **/**, **.**, **..**, *directory* becomes the new working directory. If neither case applies, **cd** tries to find the designated directory relative to one of the paths specified by the **CDPATH** shell variable. **CDPATH** has the same syntax as, and similar semantics to, the **PATH** shell variable. **cd** must have execute (search) permission in *directory*.

cd exists only as a shell built-in command because a new process is created whenever a command is executed, making **cd** useless if written and processed as a normal system command. Moreover, different shells provide different implementations of **cd** as a built-in utility. Features of **cd** as described here may not be supported by all the shells. Refer to individual shell manual entries for differences.

If **cd** is called in a subshell or a separate utility execution environment such as:

```
find . -type d -exec cd {}; -exec foo {};
```

(which invokes **foo** on accessible directories) **cd** does not affect the current directory of the caller's environment. Another usage of **cd** as a stand-alone command is to obtain the exit status of the command.

EXTERNAL INFLUENCES

International Code Set Support

Single- and multi-byte character code sets are supported.

EXAMPLES

Change the current working directory to the **HOME** directory from any location in the file system:

```
cd
```

Change to new current working directory **foo** residing in the current directory:

```
cd foo
```

or

```
cd ./foo
```

Change to directory **foobar** residing in the current directory's parent directory:

```
cd ../foobar
```

Change to the directory whose absolute pathname is **/usr/local/lib/work.files**:

```
cd /usr/local/lib/work.files
```

Change to the directory **proj1/schedule/staffing/proposals** relative to home directory:

```
cd $HOME/proj1/schedule/staffing/proposals
```

VARIABLES

The following environment variables affect the execution of cd:

HOME	The name of the home directory, used when no directory operand is specified.
CDPATH	A colon-separated list of pathnames that refer to directories. If the directory operand does not begin with a slash (**/**) character, and the first component is not dot or dot-dot, **cd** searches for *directory* relative to each directory named in the **CDPATH** variable, in the order listed. The new working directory is set to the first matching directory found. An empty string in place of a directory pathname represents the current directory. If **CDPATH** is not set, it is treated as if it was an empty string.

RETURN VALUE

Upon completion, **cd** exits with one of the following values:

0	The directory was successfully changed.
>0	An error occurred. The working directory remains unchanged.

SEE ALSO

csh(1), pwd(1), ksh(1), sh-posix(1), sh(1), chdir(2).

STANDARDS CONFORMANCE

cd: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

NAME

cdc - change the delta commentary of an SCCS delta

SYNOPSIS

cdc -r *SID* [**-m**[*mrlist*]] [**-y**[*comment*]] *files*

DESCRIPTION

The **cdc** command changes the **delta commentary**, for the *SID* specified by the **-r** option, of each named SCCS file.

Delta commentary is defined to be the Modification Request (MR) and comment information normally specified via the *delta*(1) command (**-m** and **-y** options).

If a directory is named, **cdc** behaves as if each file in the directory were specified as a named file, except that non-SCCS files (last component of the path name does not begin with **s.**) and unreadable files are silently ignored. If a name of **-** is given, the standard input is read (see WARNINGS); each line of the standard input is taken to be the name of an SCCS file to be processed.

Options

Arguments to **cdc**, which can appear in any order, consist of *option* arguments and file names.

All of the described *option* arguments apply independently to each named file:

-r*SID* Used to specify the *S*CCS *ID*entification (SID) string of a delta for which the delta commentary is to be changed.

-m[*mrlist*] If the SCCS file has the **v** option set (see *admin*(1)), a list of MR numbers to be added and/or deleted in the delta commentary of the *SID* specified by the **-r** option *may* be supplied. A null MR list has no effect.

MR entries are added to the list of MRs in the same manner as that of *delta*(1). To delete an MR, precede the MR number with the character **!** (see EXAMPLES). If the MR to be deleted is currently in the list of MRs, it is removed and changed into a "comment" line. A list of all deleted MRs is placed in the comment section of the delta commentary and preceded by a comment line stating that they were deleted.

If **-m** is not used and the standard input is a terminal, the prompt **MRs?** is issued on the standard output before the standard input is read; if the standard input is not a terminal, no prompt is issued. The **MRs?** prompt always precedes the **comments?** prompt (see **-y** option).

MRs in a list are separated by blanks and/or tab characters. An unescaped new-line character terminates the MRs list.

Note that if the **v** option has a value (see *admin*(1)), it is treated as the name of a program (or shell procedure) that validates the correctness of the MR numbers. If a non-zero exit status is returned from the MR number validation program, **cdc** terminates and the delta commentary remains unchanged.

-y[*comment*] Arbitrary text used to replace the *comment* or *comment*s already existing for the delta specified by the **-r** option. Previous comments are kept and preceded by a comment line stating that they were changed. A null *comment* has no effect.

If **-y** is not specified and the standard input is a terminal, the prompt **comments?** is issued on the standard output before standard input is read; if standard input is not a terminal, no prompt is issued. An unescaped new-line character terminates the *comment* text.

The exact permissions necessary to modify the SCCS file are documented in *get*(1). Simply stated, they are either:

- If you made the delta, you can change its delta commentary, or
- If you own the file and directory, you can modify the delta commentary.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages are displayed.

International Code Set Support

Single- and multi-byte character code sets are supported.

DIAGNOSTICS

Use *sccshelp*(1) for explanations.

EXAMPLES

Add **bl78-12345** and **bl79-00001** to the MR list, remove **bl77-54321** from the MR list, and add the comment **trouble** to delta 1.6 of **s.file**:

```
cdc -r1.6 -m"bl78-12345 !bl77-54321 bl79-00001" -ytrouble s.file
```

The following does the same thing:

```
cdc -r1.6 s.file
MRs? !bl77-54321 bl78-12345 bl79-00001
comments? trouble
```

WARNINGS

If SCCS file names are supplied to the **cdc** command via the standard input (**-** on the command line), the **-m** and **-y** options must also be used.

FILES

x-file	See *delta*(1).
z-file	See *delta*(1).

SEE ALSO

admin(1), delta(1), get(1), sccshelp(1), prs(1), sccsfile(4). rcsfile(4), acl(5), rcsintro(5).

NAME

chacl - add, modify, delete, copy, or summarize access control lists (ACLs) of files

SYNOPSIS

/usr/bin/chacl *acl file* ...

chacl -r *acl file* ...

chacl -d *aclpatt file* ...

chacl -f *fromfile tofile* ...

chacl -[**z** | **Z** | **F**] *file*...

DESCRIPTION

chacl extends the capabilities of *chmod*(1), by enabling the user to grant or restrict file access to additional specific users and/or groups. Traditional file access permissions, set when a file is created, grant or restrict access to the file's owner, group, and other users. These file access permissions (eg., **rwxrw-r--**) are mapped into three base access control list entries: one entry for the file's owner (*u*.**%,** *mode*), one for the file's group (**%.***g*, *mode*), and one for other users (**%.%,** *mode*).

chacl enables a user to designate up to thirteen additional sets of permissions (called optional access control list (ACL) entries) which are stored in the access control list of the file.

To use *chacl*, the owner (or superuser) constructs an *acl*, a set of *(user.group, mode)* mappings to associate with one or more files. A specific user and group can be referred to by either name or number; any user (*u*), group (*g*), or both can be referred to with a **%** symbol, representing *any* user or group. The **@** symbol specifies the file's owner or group.

Read, write, and execute/search (**rwx**) *modes* are identical to those used by *chmod*; symbolic operators (*op*) add (**+**), remove (**-**), or set (**=**) access rights. The entire *acl* should be quoted if it contains whitespace or special characters. Although two variants for constructing the *acl* are available (and fully explained in *acl*(5)), the following syntax is suggested:

> *entry* [, *entry*] ...

where the syntax for an *entry* is

> *u.g op mode*[*op mode*] ...

By default, **chacl** modifies existing ACLs. It adds ACL entries or modifies access rights in existing ACL entries. If *acl* contains an ACL entry already associated with a file, the entry's mode bits are changed to the new value given, or are modified by the specified operators. If the file's ACL does not already contain the specified entry, that ACL entry is added. **chacl** can also remove all access to files. Giving it a null *acl* argument means either "no access" (when using the **-r** option) or "no changes."

For a summary of the syntax, run **chacl** without arguments.

If *file* is specified as **-**, **chacl** reads from standard input.

Options

chacl recognizes the following options:

-r Replace old ACLs with the given ACL. All optional ACL entries are first deleted from the specified files's ACLs, their base permissions are set to zero, and the new ACL is applied. If *acl* does not contain an entry for the owner (*u*.**%**), the group (**%.***g*), or other (**%.%**) users of a file, that base ACL entry's mode is set to zero (no access). The command affects all of the file's ACL entries, but does not change the file's owner or group ID.

In *chmod*(1), the "modify" and "replace" operations are distinguished by the syntax (string or octal value). There is no corollary for ACLs because they have a variable number of entries. Hence **chacl** modifies specific entries by default, and optionally replaces all entries.

-d Delete the specified entries from the ACLs on all specified files. The *aclpatt* argument can be an exact ACL or an ACL pattern (see *acl*(5)). **chacl -d** updates each file's ACL only if entries are deleted from it.

If you attempt to delete a base ACL entry from any file, the entry remains but its access mode is set to zero (no access). If you attempt to delete a non-existent ACL entry from a file (that is, if an ACL entry pattern matches no ACL entry), **chacl** informs you of the error, continues, and eventually returns non-zero.

-f *fromfile tofile*
: Copy the ACL from *fromfile* to the specified *tofile*, transferring ownership, if necessary (see *acl*(5), *chown*(2), or *chownacl*(3C)). *fromfile* can be **-** to represent standard input.

  This option implies the **-r** option. If the owner and group of *fromfile* are identical to those of *tofile*, **chacl -f** is identical to:

  ```
  chacl -r ‘lsacl fromfile‘ tofile ...
  ```

  To copy an ACL without transferring ownership, the above command is suggested instead of **chacl -f**.

-z Delete (“zap”) all optional entries in the specified file’s ACLs, leaving only base entries.

-Z Delete (“zap”) all optional entries in the specified file’s ACLs, and set the access modes in all base entries to zero (no access). This is identical to replacing the old ACL with a null ACL:

  ```
  chacl -r ’’ file ...
  ```

  or using *chmod*(1), which deletes optional entries as a side effect:

  ```
  chmod 0 file ...
  ```

-F Incorporate (“fold”) optional ACL entries into base ACL entries. The base ACL entry’s permission bits are altered, if necessary, to reflect the caller’s effective access rights to the file; all optional entries, if any, are deleted.

  For ordinary users, only the access mode of the owner base ACL entry can be altered. Unlike **getaccess**, the write bit is not turned off for a file on a read-only file system or a shared-text program being executed (see *getaccess*(1)).

  For super-users, only the execute mode bit in the owner base ACL entry might be changed, only if the file is not an regular file or if an execute bit is not already set in a base ACL entry mode, but is set in an optional ACL entry mode.

acl also can be obtained from a string in a file:

```
chacl ‘cat file‘ files ...
```

Using **@** in *acl* to represent “file owner or group” can cause **chacl** to run more slowly because it must reparse the ACL for each file (except with the **-d** option).

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages are displayed.

If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**. If any internationalization variable contains an invalid setting, **chacl** behaves as if all internationalization variables are set to "C". See *environ*(5).

RETURN VALUE

If **chacl** succeeds, it returns a value of zero.

If **chacl** encounters an error before it changes any file’s ACL, it prints an error message to standard error and returns 1. Such errors include invalid invocation, invalid syntax of *acl* (*aclpatt*), a given user name or group name is unknown, or inability to get an ACL from *fromfile* with the **-f** option.

If **chacl** cannot execute the requested operation, it prints an error message to standard error, continues, and later returns 2. This includes cases when a file does not exist, a file’s ACL cannot be altered, more ACL entries would result than are allowed, or an attempt is made to delete a non-existing ACL entry.

EXAMPLES

The following command adds read access for user **jpc** in any group, and removes write access for any user in the files’s groups, for files **x** and **y**.

```
chacl "jpc.%+r, %.@-w" x y
```

This command replaces the ACL on the file open as standard input and on file **test** with one which only allows the file owner read and write access.

```
    chacl -r '(@.%,rw-)' - test
```

Delete from file **myfile** the specific access rights, if any, for user 165 in group 13. Note that this is different from adding an ACL entry that restricts access for that user and group. The user's resulting access rights depend on the entries remaining in the ACL. The command also deletes all entries for user **jpc** that have a read bit turned on (the asterisk can be used as a wildcard in the ACL pattern for user, group, or access mode):

```
    chacl -d '165.13, jpc.*+r' myfile
```

Copy the ACL from **oldfile** to **slow/hare** and **fast/tortoise**.

```
    chacl -f oldfile slow/hare fast/tortoise
```

Delete the optional ACL entries, if any, on the file open as standard input.

```
    chacl -z -
```

Deny all access to all files in the current directory whose names start with **a**, **b**, or **c**:

```
    chacl -Z [a-c]*
```

Incorporate the optional ACL entries of a file (**fun.stuff**) into the base ACL entries:

```
    chacl -F fun.stuff
```

WARNINGS

An ACL string cannot contain more than 16 unique entries, even though converting **@** symbols to user or group names and combining redundant entries might result in fewer than 16 entries for some files.

DEPENDENCIES

chacl will fail when the target file resides on a file system which does not support ACLs.

NFS

Only the **-F** option is supported on remote files.

AUTHOR

chacl was developed by HP.

SEE ALSO

chmod(1), getaccess(1), lsacl(1), getacl(2), setacl(2), acl(5), glossary(9).

NAME

chatr - change program's internal attributes

SYNOPSIS

PA32 SOM chatr

chatr [**-nqsMN**] [**-l** *library*] [**-B** *bind*] [**+b** *flag*] [**+k** *flag*] [**+l** *library*] [**+pd** *size*] [**+pi** *size*] [**+r** *flag*] [**+s** *flag*] *file* ...

PA64 ELF chatr

There are two possible syntactic forms that can be used to invoke PA64 **chatr.** The first syntactic form, which is compatible with the SOM **chatr,** is used for backward compatibility, and for easy manipulation of ordinary files that only have a single text and a single data segment:

chatr [**-s**] [**-l** *library*] [**-B** *mode*] [**+b** *flag*] [**+cd** *flag*] [**+ci** *flag*] [**+k** *flag*] [**-l** *library*] [**+md** *flag*] [**+mi** *flag*] [**+pd** *size*] [**+pi** *size*] [**+s** *flag*] [**+z** *flag*] *files*...

The second syntactic form provides the ability to explicitly specify segments to be modified:

chatr [**-s**] [**-B** *mode*] [**+s** *flag*] [**+k** *flag*] [**+dz** *flag*] [[**+si** *index* | +sa *address* | +sall]... [**+z** *flag*] [**+r** *flag]* [**+p** *size]* [**+m** *flag*] [**+c** *flag]* [**-B** *mode*] [**+s** *flag*] [**+k** *flag*]... *file*

DESCRIPTION

PA32 SOM chatr

chatr, by default, prints each *file*'s magic number and file attributes to the standard output.

Options

chatr recognizes the following options:

-l *library* Indicate that the specified shared library is subject to run-time path lookup if directory path lists are provided (see **+s** and **+b**).

-n Change *file* from demand-loaded to shared.

-q Change *file* from shared to demand-loaded.

-M Change *file* from EXEC_MAGIC to SHMEM_MAGIC. (This option is an interim solution until 64-bit addressability is available with a true 64-bit kernel.)

-N Change *file* from SHMEM_MAGIC to EXEC_MAGIC. (This option is an interim solution until 64-bit addressability is available with a true 64-bit kernel.)

-s Perform its operation silently.

-B *bind* Select run-time binding behavior of a program using shared libraries. One of the major binding modes **immediate** or **deferred** must be specified. One or more of the binding modifiers **nonfatal**, **verbose**, or **restricted** can also be specified, each with a separate option. See the *HP-UX Linker and Libraries User's Guide* manual for a description of binding modes.

+b *flag* Control whether the directory path list stored when the program was built can be used to locate shared libraries needed by the program. The two flag values, **enable** and **disable**, respectively enable and disable use of the path list. See the **+s** option.

+l *library* Indicate that the specified shared library is not subject to run-time path lookup if directory path lists are provided (see **+s** and **+b**).

+pd *size* Request a particular virtual memory page size that should be used for data. Sizes of **4K, 16K, 64K, 256K, 1M, 4M, 16M, 64M, 256M,** and **L** are supported. A size of **L** will result in using the largest page size available. The actual page size may vary if the requested size cannot be fulfilled.

+pi *size* Request a particular virtual memory page size that should be used for instructions. See the **+pd** option for additional information.

+k *flag* Request kernel assisted branch prediction. The flags **enable** and **disable** turn this request on and off, respectively.

+r *flag* Request static branch prediction when executing this program. The flags **enable** and **disable** turn this request on and off, respectively.

+s *flag* Control whether the directory path list specified with the **SHLIB_PATH** environment variable can be used to locate shared libraries needed by the program. The two flag values, **enable** and **disable**, respectively enable and disable use of the environment variable. If both **+s** and **+b** are used, their relative order on the command line indicates which path list will be searched first. See the **+b** option.

The term **shared** applies to the magic number **SHARE_MAGIC** while the term **demand-loaded** applies to the magic number **DEMAND_MAGIC**. See *magic*(4) and the *HP-UX Linker and Libraries Online User Guide* for more information.

Upon completion, **chatr** prints the file's old and new values to standard output unless **-s** is specified.

The **+pd** and **+pi** options only provide a hint for the virtual memory page size. The actual page sizes may vary. Under certain conditions, page size hints of **L** may result in better performance, depending on the specific memory requirements of the application.

The performance of some applications may benefit from static branch prediction, others may not. The **+r** option provides a hint for using or avoiding this feature.

PA64 ELF chatr

PA64 **chatr** is similar to SOM **chatr** but supports some new options and obsoletes some old ones. New options:

+dz	Enables or disables lazy swap allocation for dynamically allocated segments (such as the stack or heap). Valid only with the second command line form.
+p	Set the page size for a specified segment. Only valid with the second command line form.
+md	Set the modification bit for the file's data segment(s). Only valid with the first command line form.
+mi	Set the modification bit for the file's text segment(s). Only valid with the first command line form.
+m	Set the modification bit for a specified segment. Only valid with the second command line form.
+cd	Set the code bit for the file's data segment(s). Only valid with the first command line form.
+ci	Set the code bit for the file's text segments(s). Only valid with the first command line form.
+c	Set the code bit for a specified segment. Only valid with the second command line form.
+z	Enable lazy swap on all data segments (using the first command syntax) or on a specific segment (using the second command syntax). May not be used with non-data segments.
+si	Identify a segment using a segment index number. Valid only with the second command line form.
+sa	Identify a segment using an address. Valid only with the second command line form.
+sall	Use all segments in the file for a set of attribute modifications. Valid only with the second command line form.

Obsolete options: **-n**, **-q**, **-M**, **-N**, **-l**, **+l**.

RETURN VALUE

chatr returns zero on success. If the command line contents is syntactically incorrect, or one or more of the specified files cannot be acted upon, **chatr** returns the number of files whose attributes could not be modified. If no files are specified, **chatr** returns decimal 255.

EXTERNAL INFLUENCES

Environment Variables

The following internationalization variables affect the execution of **chatr**:

LANG	Determines the locale category for native language, local customs and coded character set in the absence of **LC_ALL** and other **LC_*** environment variables. If **LANG** is not specified or is set to the empty string, a default of **C** (see *lang*(5)) is used instead of **LANG**.
LC_ALL	Determines the values for all locale categories and has precedence over **LANG** and other **LC_*** environment variables.

LC_CTYPE	Determines the locale category for character handling functions.
LC_MESSAGES	Determines the locale that should be used to affect the format and contents of diagnostic messages written to standard error.
LC_NUMERIC	Determines the locale category for numeric formatting.
NLSPATH	Determines the location of message catalogues for the processing of **LC_MESSAGES**.

If any internationalization variable contains an invalid setting, **chatr** behaves as if all internationalization variables are set to **C**. See *environ*(5).

In addition, the following environment variable affects **chatr**:

TMPDIR	Specifies a directory for temporary files (see *tmpnam*(3S)).

EXAMPLES

Change **a.out** to demand-loaded

```
chatr -q a.out
```

Change binding mode of program file that uses shared libraries to immediate and nonfatal. Also enable usage of **SHLIB_PATH** environment variable:

```
chatr -B immediate -B nonfatal +s enable a.out
```

Disallow run-time path lookup for the shared library **/usr/lib/libc.sl** that the shared library **libfoo.sl** depends on:

```
chatr +l /usr/lib/libc.sl libfoo.sl
```

NOTES

SHMEM_MAGIC is an interim solution until 64-bit addressability is available with a true 64-bit kernel.

SHMEM_MAGIC will not be supported on future HP implementations of 64-bit architectures (beyond PA2.0). Programs that need larger than 1.75 GB of shared memory on those architectures will have to be recompiled (as 64-bit executables) for those architectures.

Programs that are compiled as 64-bit executables on any 64-bit HP implementation (including PA 2.0) cannot be marked as SHMEM_MAGIC nor do they need to be as they will already have access to more than 1.75 GB of shared memory.

The additional 1 GB of shared memory that is available over other types of executables can be availed of only for system V shared memory and not other forms of shared memory (like memory mapped files).

AUTHOR

chatr was developed by HP.

SEE ALSO

System Tools:

ld(1)	invoke the link editor
cachectl(3C)	flush and/or purge the cache

Miscellaneous:

a.out(4)	assembler, compiler, and linker output
magic(4)	magic number for HP-UX implementations

Texts and Tutorials:

HP-UX Linker and Libraries User's Guide

NAME

checknr - check nroff/troff files

SYNOPSIS

checknr [**-s**] [**-f**] [**-a.***x1***.***y1***.***x2***.***y2* ... **.***xn***.***yn*] [**-c.***x1***.***x2***.***x3*...c **.***xn*] [*file* ...]

DESCRIPTION

checknr searches a list of **nroff** or **troff** input files for certain kinds of errors involving mismatched opening and closing delimiters and unknown commands. If no files are specified, **checknr** searches the standard input. **checknr** looks for the following:

- Font changes using **\fx** ... **\fP**.
- Size changes using **\sx** ... **\s0**.
- Macros that come in *open* ... *close* forms, such as the **.TS** and **.TE** macros, which must appear in matched pairs.

checknr knows about the **ms** and **me** macro packages.

Options

checknr recognizes the following options:

-a Define additional macro pairs in the list. **-a** is followed by groups of six characters, each group defining a pair of macros. Each six characters consist of a period, the first macro name, another period, and the second macro name. For example, to define the pairs **.BS** and **.ES**, and **.XS** and **.XE**, use:

```
-a.BS.ES.XS.XE
```

No spaces are allowed between the option and its arguments.

-c Define commands that **checknr** would otherwise interpret as undefined.

-f Ignore **\f***x* font changes.

-s Ignore **\s***x* size changes.

EXTERNAL INFLUENCES

International Code Set Support

Single-byte character code sets are supported.

DIAGNOSTICS

checknr complains about unmatched delimiters, unrecognized commands, and bad command syntax.

EXAMPLES

Check file **sorting** for errors that involve mismatched opening and closing delimiters and unknown commands, but disregard errors caused by font changes:

```
checknr -f sorting
```

WARNINGS

checknr is designed for use on documents prepared with the intent of using **checknr**, much the same as **lint** is used. It expects a certain document writing style for **\f**... and **\s**... commands, in which each **\f***x* is terminated with **\fP** and each **\s***x* is terminated with **\s0**. Although text files format properly when the next font or point size is coded directly instead of using **\fP** or **\s0**, such techniques produce complaints from *checknr*. If files are to be examined by *checknr*, the **\fP** and **\s0** delimiting conventions should be used.

-a cannot be used to define single-character macro names.

checknr does not recognize certain reasonable constructs such as conditionals.

AUTHOR

checknr was developed by the University of California, Berkeley.

SEE ALSO

checkeq(1), lint(1), nroff(1).

NAME

chfn - change user information; used by **finger**

SYNOPSIS

chfn [*login-name*]

chfn -r files [*login-name*]

chfn -r nis [*login-name*]

chfn -r nisplus [*login-name*]

chfn -r dce [*login-name*]

DESCRIPTION

The **chfn** command changes the user information that is stored in the **repository** for the current logged-in user or for the user specified by *login-name* (see *passwd*(1)).

The information is organized as four comma-separated subfields within the reserved (5th) field of the password file entry. It consists of the user's full name, location code, office phone number, and home phone number, in that order. This information is used by the **finger** command and other programs (see *finger*(1)).

chfn prompts you for each subfield. The prompt includes a default value, which is enclosed in brackets. Accept the default value by pressing the Return key. To enter a blank subfield, type the word **none**.

The DCE repository (**-r dce**) is only available if Integrated Login has been configured, see *auth.adm*(1M). If Integrated Login has been configured, other considerations apply. A user with appropriate DCE privileges is capable of modifying a user's finger (gecos) information; this is not dependent upon superuser privileges.

If the repository is not specified; i.e., **chfn** [*login-name*], the finger information is changed in the passwd file only.

Run **finger** after running **chfn** to make sure the information was processed correctly.

Options

The following option is recognized:

-r Specify the repository to which the operation is to be applied. Supported repositories include **files**, **nis**, **nisplus**, and **dce**.

Subfield Values

Name Up to 1022 printing characters.

The **finger** command and other utilities expand an **&** found anywhere in this subfield by substituting the login name for it and shifting the first letter of the login name to uppercase. (**chfn** does not alter the input **&**.)

Location Up to 1022 printing characters.

Office Phone Up to 25 printing characters.

finger inserts appropriate hyphens if the value is all digits.

Home Phone Up to 25 printing characters.

finger inserts appropriate hyphens if the value is all digits.

Security Restrictions

You must have the **owner** kernel authorization and the **syslo** sensitivity label to run **chfn**.

You must have appropriate privileges to use the optional *login-name* argument to change another user's information.

EXAMPLES

The following is a sample run. The user's input is shown in regular type.

```
Name [Tracy Simmons]:
Location (Ex: 47U-P5) []: 42L-P1
Office Phone (Ex: 1632) [77777]: 71863
```

```
    Home Phone (Ex: 9875432) [4085551546]: none
```

WARNINGS

The encoding of office and extension information is installation-dependent.

For historical reasons, the user's name, etc., are stored in the **/etc/passwd** file. This is an inappropriate place to store the information.

Because two users may try to write the **passwd** file at once, a synchronization method was developed. On rare occasions, **chfn** prints a message that the password file is busy. When this occurs, **chfn** sleeps for a short time, then tries to write to the **passwd** file again.

AUTHOR

chfn was developed by the University of California, Berkeley.

FILES

/etc/passwd
/etc/ptmp

NOTES

The **chfn** command is a hard link to **passwd** command. When **chfn** is executed actually the **passwd** command gets executed with appropriate arguments to change the user gecos information in the *repository* specified in command line. If no *repository* is specified the gecos information is changed in **/etc/passwd** file.

SEE ALSO

chsh(1), finger(1), passwd(1), passwd(4).

NAME

chkey - change user's secure RPC key pair

SYNOPSIS

chkey [**-p**] [**-s nisplus** | **nis** | **files**]

DESCRIPTION

chkey is used to change a user's secure RPC public key and secret key pair. **chkey** prompts for the old secure-rpc password and verifies that it is correct by decrypting the secret key. If the user has not already keylogged in, **chkey** registers the secret key with the local *keyserv*(1M) daemon. If the secure-rpc password does not match the login password, **chkey** prompts for the login password. **chkey** uses the login password to encrypt the user's secret Diffie-Hellman (192 bit) cryptographic key.

chkey ensures that the login password and the secure-rpc password are kept the same.

The key pair can be stored in the **/etc/publickey** file, (see *publickey*(4)), NIS **publickey** map or NIS+ **cred.org_dir** table. If a new secret key is generated, it will be registered with the local *keyserv*(1M) daemon.

If the source of the **publickey** is not specified with the **-s** option, **chkey** consults the **publickey** entry in the name service switch configuration file (see *nsswitch.conf*(4)). If the **publickey** entry specifies one and only one source, then **chkey** will change the key in the specified name service. However, if multiple name services are listed, **chkey** can not decide which source to update and will display an error message. The user should specify the source explicitly with the **-s** option.

Non root users are not allowed to change their key pair in the **/etc/publickey** file.

Options

-p	Re-encrypt the existing secret key with the user's login password.
-s nisplus	Update the NIS+ database.
-s nis	Update the NIS database.
-s files	Update the **files** database.

AUTHOR

chkey was developed by Sun Microsystems, Inc.

FILES

/etc/nsswitch.conf
/etc/publickey

SEE ALSO

keylogin(1), keylogout(1), keyserv(1M), newkey(1M), nisaddcred(1M), nsswitch.conf(4), publickey(4).

NAME

chmod - change file mode access permissions

SYNOPSIS

/usr/bin/chmod [**-A**] [**-R**] *symbolic_mode_list file* ...

Obsolescent form:

/usr/bin/chmod [**-A**] [**-R**] *numeric_mode file* ...

DESCRIPTION

The **chmod** command changes the permissions of one or more *file*s according to the value of *symbolic_mode_list* or *numeric_mode*. You can display the current permissions for a file with the **ls -l** command (see *ls*(1)).

Symbolic Mode List

A *symbolic_mode_list* is a comma-separated list of operations in the following form. Whitespace is not permitted.

[*who*]*op*[*permission*][**,**...]

The variable fields can have the following values:

who One or more of the following letters:

- **u** Modify permissions for user (owner).
- **g** Modify permissions for group.
- **o** Modify permissions for others.
- **a** Modify permissions for all users (**a** is equivalent to **ugo**).

op Required; one of the following symbols:

- **+** Add *permission* to the existing file mode bits of *who*.
- **-** Delete *permission* from the existing file mode bits of *who*.
- **=** Replace the existing mode bits of *who* with *permission*.

permission One or more of the following letters:

- **r** Add or delete the read permission for *who*.
- **w** Add or delete the write permission for *who*.
- **x** Add or delete the execute file (search directory) permission for *who*.
- **s** Add or delete the set-owner-ID-on-file-execution or set-group-ID-on-file-execution permission for *who*. Useful only if **u** or **g** is expressed or implied in *who*.
- **t** Add or delete the save-text-image-on-file-execution (sticky bit) permission. Useful only if **u** is expressed or implied in *who*. See *chmod*(2).
- **X** Conditionally add or delete the execute/search permission as follows:
  - If *file* is a directory, add or delete the search permission to the existing file mode for *who*. (Same as **x**.)
  - If *file* is not a directory, and the current file permissions include the execute permission (**ls -l** displays an **x** or an **s**) for at least one of user, group, or other, then add or delete the execute file permission for *who*.
  - If *file* is not a directory, and no execute permissions are set in the current file mode, then do not change any execute permission.

Or one only of the following letters:

- **u** Copy the current user permissions to *who*.
- **g** Copy the current group permissions to *who*.
- **o** Copy the current other permissions to *who*.

The operations are performed in the order specified, and can override preceding operations specified in the same command line.

If *who* is omitted, the **r**, **w**, **x**, and **X** permissions are changed for all users if the changes are permitted by the current file mode creation mask (see *umask*(1)). The **s** and **t** permissions are changed as if **a** was specified in *who*.

Omitting *permission* is useful only when used with **=** to delete all permissions.

Numeric Mode (Obsolescent)

Absolute permissions can be set by specifying a *numeric_mode*, an octal number constructed from the logical OR (sum) of the following mode bits:

Miscellaneous mode bits:

4000	(= **u=s**)	Set user ID on file execution (file only)
2000	(= **g=s**)	Set group ID on file execution (file only)
1000	(= **u=t**)	Set sticky bit; see *chmod*(2)

Permission mode bits:

0400	(= **u=r**)	Read by owner
0200	(= **u=w**)	Write by owner
0100	(= **u=x**)	Execute (search in directory) by owner
0040	(= **g=r**)	Read by group
0020	(= **g=w**)	Write by group
0010	(= **g=x**)	Execute/search by group
0004	(= **o=r**)	Read by others
0002	(= **o=w**)	Write by others
0001	(= **o=x**)	Execute/search by others

Options

-A Preserve any optional access control list (ACL) entries associated with the file. By default, in conformance with the IEEE Standard POSIX 1003.1-1988, optional ACL entries are deleted. For information about access control lists, see *acl*(5).

-R Recursively change the file mode bits. For each *file* operand that names a directory, **chmod** alters the file mode bits of the named directory and all files and subdirectories in the file hierarchy below it.

Only the owner of a file, or a user with appropriate privileges, can change its mode.

Only a user having appropriate privileges can set (or retain, if previously set) the sticky bit of a regular file.

In order to set the set-group-ID bit, the group of the file must correspond to your current group ID.

If **chmod** is used on a symbolic link, the mode of the file referred to by the link is changed.

EXTERNAL INFLUENCES

Environment Variables

LC_MESSAGES determines the language in which messages are displayed.

If **LC_MESSAGES** is not specified or is null, it defaults to the value of **LANG**. If **LANG** is not specified or is null, it defaults to **C** (see *lang*(5)).

If any internationalization variable contains an invalid setting, all internationalization variables default to **C**. See *environ*(5).

International Code Set Support

Single- and multibyte character code sets are supported.

RETURN VALUE

Upon completion, **chmod** returns one of the following values:

0	Successful completion.
>0	An error condition occurred.

EXAMPLES

Deny write permission to others:

chmod o-w *file*

Make a file executable by everybody:

chmod a+x *file*

Assign read and execute permission to everybody, and set the set-user-ID bit:

chmod a=rx,u+s *file*

Assign read and write permission to the file owner, and read permission to everybody else:

`chmod u=rw,go=r` *file*

or the obsolescent form:

`chmod 644` *file*

Traverse a directory subtree making all regular files readable by user and group only, and all executables and directories executable (searchable) by everyone:

`chmod -R ug+r,o-r,a+X` *pathname*

If the current value of **`umask`** is **`020`** (**`umask -S`** displays **`u=rwx,g=rx,o=rwx`**; do not change write permission for group) and the current permissions for file **`mytest`** are **`444`** (**`a=r`**), displayed by **`ls -l`** as **`-r--r--r--`**, then the command

`chmod +w mytest`

sets the permissions to **`646`** (**`uo=rw,g=r`**), displayed by **`ls -l`** as **`-rw-r--rw-`**.

If the current value of **`umask`** is **`020`** (**`umask -S`** displays **`u=rwx,g=rx,o=rwx`**; do not change write permission for group) and the current permissions for file **`mytest`** are **`666`** (**`a=rw`**), displayed by **`ls -l`** as **`-rw-rw-rw-`**, then the command

`chmod -w mytest`

sets the permissions to **`464`** (**`uo=r,g=rw`**), displayed by **`ls -l`** as **`-r--rw-r--`**.

DEPENDENCIES

The **`-A`** option causes **`chmod`** to fail on file systems that do not support ACLs.

AUTHOR

`chmod` was developed by AT&T and HP.

SEE ALSO

chacl(1), ls(1), umask(1), chmod(2), acl(5).

STANDARDS CONFORMANCE

`chmod`: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

NAME

chown, chgrp - change file owner or group

SYNOPSIS

chown [**-h**] [**-R**] *owner*[**:***group*] *file* ...

chgrp [**-h**] [**-R**] *group file* ...

DESCRIPTION

The **chown** command changes the owner ID of each specified *file* to *owner* and optionally the group ID of each specified *file* to *group*.

The **chgrp** command changes the group ID of each specified *file* to *group*.

owner can be either a decimal user ID or a login name found in the **/etc/passwd** file.

group can be either a decimal group ID or a group name found in the **/etc/group** file.

In order to change the owner or group, you must own the file and have the CHOWN privilege (see *setprivgrp*(1M)). If either command is invoked on a regular file by other than the superuser, the set-user-ID and set-group-ID bits of the file mode (04000 and 02000 respectively) are cleared. Note that a given user's or group's ability to use this command can be restricted by **setprivgrp** (see *setprivgrp*(1M)).

Access Control Lists – HFS File Systems Only

Users can permit or deny specific individuals and groups to access a file by setting optional ACL entries in the file's access control list (see *acl*(5)). When using **chown** in conjunction with ACLs, if the new owner and/or group of a file does not have an optional ACL entry corresponding to *user***.%** and/or **%.***group* in the file's access control list, the file's access permission bits remain unchanged. However, if the new owner and/or group is already designated by an optional ACL entry of *user***.%** and/or **%.***group* in the file's ACL, **chown** sets the corresponding file access permission bits (and the corresponding base ACL entries) to the permissions contained in that entry.

Options

chown and **chgrp** recognize the following options:

-h Change the owner or group of a symbolic link.

By default, the owner or group of the target file that a symbolic link points to is changed. With **-h**, the target file that the symbolic link points to is not affected. If the target file is a directory, and you specify **-h** and **-R**, recursion does not take place.

-R Recursively change the owner or group. For each *file* operand that names a directory, the owner or group of the directory and all files and subdirectories in the file hierarchy below it are changed.

EXTERNAL INFLUENCES

Environment Variables

LC_MESSAGES determines the language in which messages are displayed.

If **LC_MESSAGES** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**.

If any internationalization variable contains an invalid setting, **chown** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported.

RETURN VALUE

chown and **chgrp** return the following values:

0	Successful completion.
>0	An error condition occurred.

EXAMPLES

The following command changes the owner of the file **jokes** to **sandi**:

```
chown sandi jokes
```

The following command searches the directory **design_notes** and changes each file in that directory to owner **mark** and group **users**:

```
chown -R mark:users design_notes
```

WARNINGS

The default operation of **chown** and **chgrp** for symbolic links has changed as of HP-UX release 10.0. Use the **-h** option to get the former default operation.

FILES

/etc/group
/etc/passwd

SEE ALSO

chmod(1), setprivgrp(1M), chown(2), group(4), passwd(4), acl(5).

STANDARDS CONFORMANCE

chown: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

chgrp: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

NAME

chsh - change default login shell

SYNOPSIS

chsh *login-name* [*shell*]

chsh -r files *login-name* [*shell*]

chsh -r nisplus *login-name* [*shell*]

chsh -r nis *login-name* [*shell*]

chsh -r dce *login-name* [*shell*]

DESCRIPTION

The **chsh** command changes the login-shell for a user's login name in the repository (see *passwd*(1)).

The DCE repository (**-r dce**) is only available if Integrated Login has been configured, see *auth.adm*(1M). If Integrated Login has been configured, other considerations apply. A user with appropriate DCE privileges is capable of modifying a user's shell; this is not dependent upon superuser privileges.

If the repository is not specified; i.e., **chsh** [*login-name*], the login shell is changed in passwd file only.

Run **finger** after running **chsh** to make sure the information was processed correctly.

Arguments

login-name	A login name of a user.
shell	The absolute path name of a shell. If the file **/etc/shells** exists, the new login shell must be listed in that file. Otherwise, you can specify one of the standard shells listed in the *getusershell*(3C) manual entry. If *shell* is omitted, it defaults to the POSIX shell, **/usr/bin/sh**.

Options

The following option is recognized:

-r	Specify the repository to which the operation is to be applied. Supported repositories include **files**, **nis**, **nisplus**, and **dce**.

Security Restrictions

You must have the **syslog** sensitivity label to execute **chsh**. You must have the **owner** kernel authorization to change another user's login shell.

NETWORKING FEATURES

NFS

File **/etc/passwd** can be implemented as a Network Information Service (NIS) database.

EXAMPLES

To change the login shell for user **voltaire** to the default:

```
chsh voltaire
```

To change the login shell for user **descartes** to the C shell:

```
chsh descartes /usr/bin/csh
```

To change the login shell for user **aristotle** to the Korn shell in the DCE registry:

```
chsh -r dce aristotle /usr/bin/ksh
```

WARNINGS

As many users may try to write the **/etc/passwd** file simultaneously, a passwd locking mechanism was deviced. If this locking fails after subsequent retrying, **chsh** terminates.

AUTHOR

chsh was developed by HP and the University of California, Berkeley.

NOTES

The **chsh** command is a hard link to **passwd** command. When **chsh** is executed actually the **passwd** command gets executed with appropriate arguments to change the user login shell in the *repository* specified in command line. If no *repository* is specified the login shell is changed in **/etc/passwd** file.

FILES

/etc/shells
/etc/ptmp

SEE ALSO

chfn(1), csh(1), ksh(1), pam(1), passwd(1), sh(1), sh_bourne(1), sh_posix(1), getusershell(3C), passwd(4), shells(4).

NAME

ci - check in RCS revisions

SYNOPSIS

ci [*options*] *file*...

DESCRIPTION

ci stores new revisions into RCS files. Each file name ending in **,v** is treated as an RCS file; all others are assumed to be working files. **ci** deposits the contents of each working file into the corresponding RCS file (see *rcsintro*(5)).

If the RCS file does not exist, **ci** creates it and deposits the contents of the working file as the initial revision. The default number is "1.1". The access list is initialized to empty. Instead of the log message, **ci** requests descriptive text (see the **-t** option below).

An RCS file created by **ci** inherits the read and execute permissions from the working file. If the RCS file exists, **ci** preserves its read and execute permissions. **ci** always turns off all write permissions of RCS files.

The caller of the command must have read/write permission for the directories containing the RCS file and the working file, and read permission for the RCS file itself. A number of temporary files are created. A semaphore file is created in the directory containing the RCS file. **ci** always creates a new RCS file and unlinks the old one; therefore links to RCS files are useless.

For **ci** to work, the user's login must be in the access list unless the access list is empty, the user is the owner of the file, or the user is super-user.

Normally, **ci** checks whether the revision to be deposited is different from the preceding one. If it is not different, **ci** either aborts the deposit (if **-q** is given) or asks whether to abort (if **-q** is omitted). A deposit can be forced with the **-f** option.

For each revision deposited, **ci** prompts for a log message. The log message should summarize the change and must be terminated with a line containing a single "." or a control-D. If several files are being checked in, **ci** asks whether or not to reuse the log message from the previous file. If the standard input is not a terminal, **ci** suppresses the prompt and uses the same log message for all files (see **-m** option below.

The number of the deposited revision can be given with any of the options **-r**, **-f**, **-k**, **-l**, **-u**, or **-q** (see **-r** option below).

To add a new revision to an existing branch, the head revision on that branch must be locked by the caller. Otherwise, only a new branch can be created. This restriction is not enforced for the owner of the file, unless locking is set to **strict** (see *rcs*(1)). A lock held by someone else can be broken with the **rcs** command (see *rcs*(1)).

Options

-f[*rev*]	Forces a deposit. The new revision is deposited even if it is not different from the preceding one.
-k[*rev*]	Searches the working file for keyword values to determine its revision number, creation date, author, and state (see *co*(1)), and assigns these values to the deposited revision, rather than computing them locally. A revision number given with a command option overrides the number in the working file. This option is useful for software distribution. A revision that is sent to several sites should be checked in with the **-k** option at these sites to preserve its original number, date, author, and state.
-l[*rev*]	Works like **-r**, except it performs an additional **co -l** for the deposited revision. Thus, the deposited revision is immediately checked out again and locked. This is useful for saving a revision although one wants to continue editing it after the check-in.
-m"*msg***"**	Uses the string *msg* as the log message for all revisions checked in.
-n"*name***"**	Assigns the symbolic name *name* to the checked-in revision. **ci** prints an error message if *name* is already assigned to another number.
-N"*name***"**	Same as **-n**, except that it overrides a previous assignment of *name*.
-q[*rev*]	Quiet mode; diagnostic output is not printed. A revision that is not different from the preceding one is not deposited unless **-f** is given.

-r[*rev*] Assigns the revision number *rev* to the checked-in revision, releases the corresponding lock, and deletes the working file. This is the default.

If *rev* is omitted, **ci** derives the new revision number from the caller's last lock. If the caller has locked the head revision of a branch, the new revision is added to the head of that branch and a new revision number is assigned to the new revision. The new revision number is obtained by incrementing the head revision number. If the caller locked a non-head revision, a new branch is started at the locked revision, and the number of the locked revision is incremented. The default initial branch and level numbers are 1. If the caller holds no lock, but is the owner of the file and locking is not set to *strict*, the revision is added to the head of the trunk.

If *rev* indicates a revision number, it must be higher than the latest one on the branch to which *rev* belongs, or must start a new branch.

If *rev* indicates a branch instead of a revision, the new revision is added to the head of that branch. The level number is obtained by incrementing the head revision number of that branch. If *rev* indicates a non-existing branch, that branch is created with the initial revision numbered *rev* **.1**.

NOTE: On the trunk, revisions can be added to the head, but not inserted.

-s"*state***"** Sets the state of the checked-in revision to the identifier *state*. The default is **Exp**.

-t[*txtfile*] Writes descriptive text into the RCS file (deletes the existing text). If *txtfile* is omitted, **ci** prompts the user for text from standard input that is terminated with a line containing a single **.** or Ctrl-D. Otherwise, the descriptive text is copied from the file *txtfile*. During initialization, descriptive text is requested even if **-t** is not given. The prompt is suppressed if standard input is not a terminal.

-u[*rev*] Similar to **-l**, except that the deposited revision is not locked. This is useful if one wants to process (e.g., compile) the revision immediately after check in.

Access Control Lists (ACLs)

Optional ACL entries should not be added to RCS files, because they might be deleted.

DIAGNOSTICS

For each revision, **ci** prints the RCS file, the working file, and the number of both the deposited and the preceding revision. The exit status always refers to the last file checked in, and is 0 if the operation was successful, 1 if unsuccessful.

EXAMPLES

If the current directory contains a subdirectory **RCS** with an RCS file **io.c,v**, all of the following commands deposit the latest revision from **io.c** into **RCS/io.c,v**:

```
ci io.c
ci RCS/io.c,v
ci io.c,v
ci io.c RCS/io.c,v
ci io.c io.c,v
ci RCS/io.c,v io.c
ci io.c,v io.c
```

Check in version 1.2 of RCS file **foo.c,v**, with the message **Bug fix**:

```
ci -r1.2 -m"Bug Fix" foo.c,v
```

WARNINGS

The names of RCS files are generated by appending **,v** to the end of the working file name. If the resulting RCS file name is too long for the file system on which the RCS file should reside, **ci** terminates with an error message.

The log message cannot exceed 2046 bytes.

A file with approximately 240 revisions may cause a hash table overflow. **ci** cannot add another revision to the file until some of the old revisions have been removed. Use the **rcs -o** (obsolete) command option to remove old revisions.

RCS is designed to be used with TEXT files only. Attempting to use RCS with non-text (binary) files results in data corruption.

AUTHOR

`ci` was developed by Walter F. Tichy.

SEE ALSO

co(1), ident(1), rcs(1), rcsdiff(1), rcsmerge(1), rlog(1), rcsfile(4), acl(5), rcsintro(5).

NAME

cksum - print file checksum and sizes

SYNOPSIS

cksum [*file* ...]

DESCRIPTION

The **cksum** command calculates and prints to standard output a checksum for each named file, the number of octets in the file and the filename.

cksum uses a portable algorithm based on a 32-bit Cyclic Redundancy Check. This algorithm finds a broader spectrum of errors than the 16-bit algorithms used by **sum** (see *sum*(1)). The CRC is the sum of the following expressions, where *x* is each byte of the file.

$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{7} + x^{5} + x^{4} + x^{2} + x^{1} + x^{0}$$

The results of the calculation are truncated to a 32-bit value. The number of bytes in the file is also printed.

Standard input is used if no file names are given.

cksum is typically used to verify data integrity when copying files between systems.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the locale to use for the locale categories when both **LC_ALL** and the corresponding environment variable (beginning with **LC_**) do not specify a locale. If **LANG** is not set or is set to the empty string, a default of "C" (see *lang*(5)) is used.

LC_CTYPE determines the locale for interpretation of sequences of bytes of text data as characters (e.g., single- verses multibyte characters in arguments and input files).

LC_MESSAGES determines the language in which messages are displayed.

If any internationalization variable contains an invalid setting, **cksum** behaves as if all internationalization variables are set to "C". See *environ*(5).

RETURN VALUE

Upon completion, **cksum** returns one of the following values:

0	All files were processed successfully.
>0	One or more files could not be read or another error occurred.

If an inaccessible file is encountered, **cksum** continues processing any remaining files, but the final exit status is affected.

SEE ALSO

sum(1), wc(1), pdf(4).

STANDARDS CONFORMANCE

cksum: XPG4, POSIX.2

NAME

clear - clear terminal screen

SYNOPSIS

clear

DESCRIPTION

clear clears the terminal screen if it is possible to do so. It reads the **TERM** environment variable for the terminal type, then reads the appropriate **terminfo** database to determine how to clear the screen.

FILES

/usr/share/lib/terminfo/?/* terminal database files

AUTHOR

clear was developed by the University of California, Berkeley.

SEE ALSO

terminfo(4).

NAME

cmp - compare two files

SYNOPSIS

cmp [**-l**] [**-s**] *file1 file2* [*skip1* [*skip2*]]

DESCRIPTION

cmp compares two files (if *file1* or *file2* is **-**, the standard input is used). Under default options, **cmp** makes no comment if the files are the same; if they differ, it announces the byte and line number at which the difference occurred. If one file is an initial subsequence of the other, that fact is noted. *skip1* and *skip2* are initial byte offsets into *file1* and *file2*, respectively; and maybe octal or decimal; the form of the number is determined by the environment variable **LC_NUMERIC** (in the C locale, a leading 0 denotes an octal number. See **LANG** on *environ*(5) and *strtol*(3C)).

cmp recognizes the following options:

-l Print the byte number (decimal) and the differing bytes (octal) for each difference (byte numbering begins at 1 rather than 0).

-s Print nothing for differing files; return codes only.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages are displayed. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**. If any internationalization variable contains an invalid setting, **cmp** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported.

DIAGNOSTICS

cmp returns the following exit values:

0 Files are identical.
1 Files are not identical.
2 Inaccessible or missing argument.

cmp prints the following warning if the comparison succeeds till the end of file of file1(file2) is reached.

```
cmp: EOF on file1(file2)
```

SEE ALSO

comm(1), diff(1).

STANDARDS CONFORMANCE

cmp: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

NAME

co - check out RCS revisions

SYNOPSIS

co [*options*] *file* ...

DESCRIPTION

co retrieves revisions from RCS files. Each file name ending in **,v** is taken to be an RCS file. All other files are assumed to be working files. **co** retrieves a revision from each RCS file and stores it in the corresponding working file (see also *rcsintro*(5)).

Revisions of an RCS file can be checked out locked or unlocked. Locking a revision prevents overlapping updates. A revision checked out for reading or processing (e.g., compiling) need not be locked. A revision checked out for editing and later checked in must normally be locked. Locking a revision currently locked by another user fails (a lock can be broken with the **rcs** command, but poses inherent risks when independent changes are being made simultaneously (see *rcs*(1)). **co** with locking requires the caller to be on the access list of the RCS file unless: he is the owner of the file, a user with appropriate privileges, or the access list is empty. **co** without locking is not subject to access list restrictions.

A revision is selected by number, check-in date/time, author, or state. If none of these options are specified, the latest revision on the trunk is retrieved. When the options are applied in combination, the latest revision that satisfies all of them is retrieved. The options for date/time, author, and state retrieve a revision on the selected branch. The selected branch is either derived from the revision number (if given), or is the highest branch on the trunk. A revision number can be attached to the options **-l**, **-p**, **-q**, or **-r**.

The caller of the command must have write permission in the working directory, read permission for the RCS file, and either read permission (for reading) or read/write permission (for locking) in the directory that contains the RCS file.

The working file inherits the read and execute permissions from the RCS file. In addition, the owner write permission is turned on, unless the file is checked out unlocked and locking is set to **strict** (see *rcs*(1)).

If a file with the name of the working file exists already and has write permission, **co** aborts the check out if **-q** is given, or asks whether to abort if **-q** is not given. If the existing working file is not writable, it is deleted before the check out.

A number of temporary files are created. A semaphore file is created in the directory of the RCS file to prevent simultaneous update.

A **co** command applied to an RCS file with no revisions creates a zero-length file. **co** always performs keyword substitution (see below).

Options

-l[*rev*] Locks the checked out revision for the caller. If omitted, the checked out revision is not locked. See option **-r** for handling of the revision number *rev*.

-p[*rev*] Prints the retrieved revision on the standard output rather than storing it in the working file. This option is useful when **co** is part of a pipe.

-q[*rev*] Quiet mode; diagnostics are not printed.

-d*date* Retrieves the latest revision on the selected branch whose check in date/time is less than or equal to *date*. The date and time may be given in free format and are converted to local time. Examples of formats for *date*:

Tue-PDT, 1981, 4pm Jul 21	(free format)
Fri April 16 15:52:25 EST 1982	(output of *ctime*(3C))
4/21/86 10:30am	(format: *mm*/*dd*/*yy hh*:*mm*:*ss*)

Most fields in the date and time can be defaulted. **co** determines the defaults in the order year, month, day, hour, minute, and second (from most- to least-significant). At least one of these fields must be provided. For omitted fields that are of higher significance than the highest provided field, the current values are assumed. For all other omitted fields, the lowest possible values are assumed. For example, the date **20, 10:30** defaults to 10:30:00 of the 20th of the current month and current year. Date/time fields can be delimited by spaces or commas. If spaces are used, the string must be surrounded by double quotes.

For 2-digit year input (*yy*) without the presence of the century field, the following interpretation is taken: [**70**-**99**, **00**-**69** (1970-1999, 2000-2069)].

-r[*rev*] Retrieves the latest revision whose number is less than or equal to *rev*. If *rev* indicates a branch rather than a revision, the latest revision on that branch is retrieved. *rev* is composed of one or more numeric or symbolic fields separated by **.** . The numeric equivalent of a symbolic field is specified with the **ci -n** and **rcs -n** commands (see *ci*(1) and *rcs*(1)).

-s*state* Retrieves the latest revision on the selected branch whose state is set to *state*.

-w[*login*] Retrieves the latest revision on the selected branch that was checked in by the user with login name *login*. If the argument *login* is omitted, the caller's login is assumed.

-j*joinlist* Generates a new revision that is the result of the joining of the revisions on *joinlist*. *joinlist* is a comma-separated list of pairs of the form rev2**:**rev3, where *rev2* and *rev3* are (symbolic or numeric) revision numbers. For the initial pair, *rev1* denotes the revision selected by the options **-l**, ..., **-w**. For all other pairs, *rev1* denotes the revision generated by the previous pair. (Thus, the output of one join becomes the input to the next.)

For each pair, **co** joins revisions *rev1* and *rev3* with respect to *rev2*. This means that all changes that transform *rev2* into *rev1* are applied to a copy of *rev3*. This is particularly useful if *rev1* and *rev3* are the ends of two branches that have *rev2* as a common ancestor. If *rev1* < *rev2* < *rev3* on the same branch, joining generates a new revision that is similar to *rev3*, but with all changes that lead from *rev1* to *rev2* undone. If changes from *rev2* to *rev1* overlap with changes from *rev2* to *rev3*, **co** prints a warning and includes the overlapping sections, delimited as follows:

```
<<<<<<<
rev1
=======
rev3
>>>>>>>
```

For the initial pair, *rev2* can be omitted. The default is the common ancestor. If any of the arguments indicate branches, the latest revisions on those branches are assumed. If the **-l** option is present, the initial *rev1* is locked.

Keyword Substitution

Strings of the form **$***keyword***$** and **$***keyword***:**...**$** embedded in the text are replaced with strings of the form **$***keyword***:** *value $*, where *keyword* and *value* are pairs listed below. Keywords may be embedded in literal strings or comments to identify a revision.

Initially, the user enters strings of the form **$***keyword***$**. On check out, **co** replaces these strings with strings of the form **$***keyword***:** *value $*. If a revision containing strings of the latter form is checked back in, the value fields are replaced during the next checkout. Thus, the keyword values are automatically updated on checkout.

Keywords and their corresponding values:

$Author$	The login name of the user who checked in the revision.
$Date$	The date and time the revision was checked in.
$Header$	A standard header containing the RCS file name, the revision number, the date, the author, and the state.
$Locker$	The login name of the user who locked the revision (empty if not locked).
Log	The log message supplied during checkin, preceded by a header containing the RCS file name, the revision number, the author, and the date. Existing log messages are *not* replaced. Instead, the new log message is inserted after **$Log:**...**$**. This is useful for accumulating a complete change log in a source file.
$Revision$	The revision number assigned to the revision.
$Source$	The full pathname of the RCS file.
$State$	The state assigned to the revision with **rcs -s** or **ci -s**.

Access Control Lists (ACLs)

Optional ACL entries should not be added to RCS files because they might be deleted.

DIAGNOSTICS

The RCS file name, the working file name, and the revision number retrieved are written to the diagnostic output. The exit status always refers to the last file checked out, and is 0 if the operation was successful, 1 if unsuccessful.

EXAMPLES

Assume the current directory contains a subdirectory named **RCS** with an RCS file named **io.c,v**. Each of the following commands retrieves the latest revision from **RCS/io.c,v** and stores it into **io.c**:

```
co  io.c
co  RCS/io.c,v
co  io.c,v
co  io.c  RCS/io.c,v
co  io.c  io.c,v
co  RCS/io.c,v  io.c
co  io.c,v  io.c
```

Check out version 1.1 of RCS file **foo.c,v**:

```
co -r1.1 foo.c,v
```

Check out version 1.1 of RCS file **foo.c,v** to the standard output:

```
co -p1.1 foo.c,v
```

Check out the version of file **foo.c,v** that existed on September 18, 1992:

```
co -d"09/18/92" foo.c,v
```

WARNINGS

The **co** command generates the working file name by removing the **,v** from the end of the RCS file name. If the given RCS file name is too long for the file system on which the RCS file should reside, **co** terminates with an error message.

There is no way to suppress the expansion of keywords, except by writing them differently. In **nroff** and **troff**, this is done by embedding the null-character **\&** into the keyword.

The **-d** option gets confused in some circumstances, and accepts no date before 1970.

The **-j** option does not work for files containing lines consisting of a single **.** .

RCS is designed to be used with *text* files only. Attempting to use RCS with non-text (binary) files results in data corruption.

AUTHOR

co was developed by Walter F. Tichy.

SEE ALSO

ci(1), ident(1), rcs(1), rcsdiff(1), rcsmerge(1), rlog(1), rcsfile(4), acl(5), rcsintro(5).

NAME

col - filter reverse line-feeds and backspaces

SYNOPSIS

col [**-blfxp**]

DESCRIPTION

col reads from the standard input and writes onto the standard output. It performs the line overlays implied by reverse line feeds (ASCII code **ESC-7**), and by forward and reverse half-line feeds (**ESC-9** and **ESC-8**). **col** is particularly useful for filtering multi-column output made with the **nroff .rt** command, and output resulting from use of the **tbl** preprocessor (see *nroff*(1) and *tbl*(1)).

If the **-b** option is given, **col** assumes that the output device in use is not capable of backspacing. In this case, if two or more characters are to appear in the same place, only the last one read is output.

If the **-l** option is given, **col** assumes the output device is a line printer (rather than a character printer) and removes backspaces in favor of multiply overstruck full lines. It generates the minimum number of print operations necessary to generate the required number of overstrikes. (All but the last print operation on a line are separated by carriage returns (\r); the last print operation is terminated by a newline (\n).)

Although **col** accepts half-line motions in its input, it normally does not emit them on output. Instead, text that would appear between lines is moved to the next lower full-line boundary. This treatment can be suppressed by the **-f** (fine) option; in this case, the output from **col** may contain forward half-line feeds (ESC-9), but will still never contain either kind of reverse line motion.

Unless the **-x** option is given, **col** converts white space to tabs on output wherever possible to shorten printing time.

The ASCII control characters SO (\016) and SI (\017) are assumed by **col** to start and end text in an alternate character set. The character set to which each input character belongs is remembered, and on output SI and SO characters are generated as appropriate to ensure that each character is printed in the correct character set.

On input, the only control characters accepted are space, backspace, tab, return, new-line, SI , SO , and VT , (\013), and ESC followed by **7**, **8**, or **9**. The VT character is an alternate form of full reverse line-feed, included for compatibility with some earlier programs of this type. All other non-printing characters are ignored.

Normally, **col** ignores any unrecognized escape sequences found in its input; the **-p** option can be used to cause **col** to output these sequences as regular characters, subject to overprinting from reverse line motions. The use of this option is highly discouraged unless the user is fully aware of the textual position of the escape sequences.

EXTERNAL INFLUENCES

Environment Variables

LANG provides a default value for the internationalization variables that are unset or null. If **LANG** is unset or null, the default value of "C" (see *lang*(5)) is used. If any of the internationalization variables contains an invalid setting, **col** will behave as if all internationalization variables are set to "C". See *environ*(5).

LC_ALL If set to a non-empty string value, overrides the values of all the other internationalization variables.

LC_CTYPE determines the interpretation of text as single and/or multi-byte characters, the classification of characters as printable, and the characters matched by character class expressions in regular expressions.

LC_MESSAGES determines the locale that should be used to affect the format and contents of diagnostic messages written to standard error and informative messages written to standard output.

NLSPATH determines the location of message catalogues for the processing of **LC_MESSAGES.**

International Code Set Support

Single- and multi-byte character code sets are supported.

EXAMPLES

col is used most often with **nroff** and **tbl**. A common usage is:

```
tbl filename | nroff -man | col | more -s
```

(very similar to the usual *man*(1) command). This command allows vertical bars and outer boxes to be printed for tables. The file is run through the **tbl** preprocessor, and the output is then piped through **nroff**, formatting the output using the **-man** macros. The formatted output is then piped through **col**, which sets up the vertical bars and aligns the columns in the file. The file is finally piped through the **more** command, which prints the output to the screen with underlining and highlighting substituted for italic and bold typefaces. The **-s** option deletes excess space from the output so that multiple blank lines are not printed to the screen.

SEE ALSO

nroff(1), tbl(1), ul(1), man(5).

NOTES

The input format accepted by **col** matches the output produced by **nroff** with either the **-T37** or **-Tlp** options. Use **-T37** (and the **-f** option of **col**) if the ultimate disposition of the output of **col** is a device that can interpret half-line motions, and **-Tlp** otherwise.

BUGS

Cannot back up more than 128 lines. Cannot back up across page boundaries.

There is a maximum limit for the number of characters, including backspaces and overstrikes, on a line. The maximum limit is at least 800 characters.

Local vertical motions that would result in backing up over the first line of the document are ignored. As a result, the first line must not have any superscripts.

WARNINGS

This command is likely to be withdrawn from X/Open standards. Applications using this command might not be portable to other vendors' systems.

STANDARDS CONFORMANCE

col: SVID2, SVID3, XPG2, XPG3

NAME

comb - combine SCCS deltas

SYNOPSIS

comb [**-p** *SID*] [**-c***list*] [**-o**] [**-s**] *file* ...

DESCRIPTION

The **comb** command generates a shell procedure (see *sh*(1)) which, when run, reconstructs the given SCCS files. The reconstructed files are usually smaller than the original files. Arguments can be specified in any order, but all options apply to all named SCCS files. If a directory is named, **comb** behaves as though each file in the directory were specified as a named file, except that non-SCCS files (last component of the path name does not begin with **s.**) and unreadable files are silently ignored. If a name of **-** is given, the standard input is read; each line of the standard input is taken to be the name of an SCCS file to be processed; non-SCCS files and unreadable files are silently ignored. The generated shell procedure is written on the standard output.

Options

comb recognizes the following options. Each is explained as if only one named file is to be processed, but the effects of any option apply independently to each named file.

-p*SID* The *S*CCS *ID*entification string (SID) of the oldest delta to be preserved. All older deltas are discarded in the reconstructed file.

-c*list* A *list* of deltas to be preserved (see *get*(1) for the syntax of a *list*). All other deltas are discarded.

-o For each **get -e** generated, this option causes the reconstructed file to be accessed at the release of the delta to be created, otherwise the reconstructed file would be accessed at the most recent ancestor. Use of the **-o** option can decrease the size of the reconstructed SCCS file. It can also alter the shape of the delta tree of the original file.

-s This option causes **comb** to generate a shell procedure which, when run, produces a report giving, for each file: the file name, size (in blocks) after combining, original size (also in blocks), and percentage change computed by:

$$100 \times (\text{original} - \text{combined}) / \text{original}$$

It is recommended that this option be used before any SCCS files are actually combined to determine exactly how much space is saved by the combining process.

If no options are specified, **comb** preserves only leaf deltas and the minimal number of ancestors needed to preserve the tree.

EXTERNAL INFLUENCES

International Code Set Support

Single- and multi-byte character code sets are supported.

DIAGNOSTICS

Use *sccshelp*(1) for explanations.

EXAMPLES

The command:

```
comb -c1.1,1.3,1.6 s.document > save_file
```

creates a shell script named **save_file**, which if executed, creates a new **s.document** using only the deltas **1.1**, **1.3**,**and 1.6** from the old **s.document**. The script overwrites the old **s.document**; thus, it might be wise to copy the original elsewhere. Here is an example of typical technique:

```
cp s.document s.save
comb -c1.1,1.3,1.6 s.document > save_file
sh save_file
```

WARNINGS

comb may rearrange the shape of the tree of deltas. Combining files may or may not save space; in fact, it is possible for the reconstructed file to actually be larger than the original.

FILES

`s.COMB?????`	Temporary file
`comb?????`	Temporary file

SEE ALSO

admin(1), delta(1), get(1), sccshelp(1), prs(1), sh(1), sccsfile(4).

NAME

comm - select or reject lines common to two sorted files

SYNOPSIS

comm [**-**[**123**]] *file1 file2*

DESCRIPTION

comm reads *file1* and *file2*, which should be ordered in increasing collating sequence (see *sort*(1) and Environment Variables below), and produces a three-column output:

Column 1:	Lines that appear only in *file1*,
Column 2:	Lines that appear only in *file2*,
Column 3:	Lines that appear in both files.

If **-** is used for *file1* or *file2*, the standard input is used.

Options 1, 2, or 3 suppress printing of the corresponding column. Thus **comm -12** prints only the lines common to the two files; **comm -23** prints only lines in the first file but not in the second; **comm -123** does nothing useful.

EXTERNAL INFLUENCES

Environment Variables

LC_COLLATE determines the collating sequence **comm** expects from the input files.

LC_MESSAGES determines the language in which messages are displayed.

If **LC_MESSAGES** is not specified in the environment or is set to the empty string, the value of **LANG** determines the language in which messages are displayed. If **LC_COLLATE** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**. If any internationalization variable contains an invalid setting, **comm** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported.

EXAMPLES

The following examples assume that **file1** and **file2** have been ordered in the collating sequence defined by the **LC_COLLATE** or **LANG** environment variable.

Print all lines common to **file1** and **file2** (in other words, print column 3):

```
comm -12 file1 file2
```

Print all lines that appear in **file1** but not in **file2** (in other words, print column 1):

```
comm -23 file1 file2
```

Print all lines that appear in **file2** but not in **file1** (in other words, print column 2):

```
comm -13 file1 file2
```

SEE ALSO

cmp(1), diff(1), sdiff(1), sort(1), uniq(1).

STANDARDS CONFORMANCE

comm: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

NAME

command - execute a simple command

SYNOPSIS

command *command_name* [*argument* ...]

DESCRIPTION

command enables the shell to treat the arguments as a simple command, suppressing the shell function lookup.

If *command_name* is not the name of the function, the effect of **command** is the same as omitting *command*.

OPERANDS

command recognizes the following operands:

command_name	The name of a HP-UX command or a shell built-in command.
argument	One or more strings to be interpreted as arguments to *command_name*.

The **command** command is necessary to allow functions that have the same name as a command to call the command (instead of a recursive call to the function).

Nothing in the description of **command** is intended to imply that the command line is parsed any differently than any other simple command. For example,

command *a* **|** *b* **;** *c*

is not parsed in any special way that causes **|** or **;** to be treated other than a pipe operator or semicolon or that prevents function lookup on *b* or *c*.

EXTERNAL INFLUENCE

Environment Variables

PATH determines the search path used during the command search.

RETURN VALUE

command exits with one of the following values:

- If **command** fails:

126	The utility specified by the **command_name** is found but not executable.
127	An error occurred in the *command* utility or the utility specified by **command_name** is not found.

- If **command** does not fail:

  The exit status of **command** is the same as that of the simple command specified by the arguments:

  command_name[*argument* ...]

EXAMPLES

Create a version of the **cd** command that always prints the name of the new working directory whenever it is used:

```
cd() {
        command "$@" >/dev/null
        pwd
}
```

Circumvent the redefined **cd** command above, and change directories without printing the name of the new working directory:

```
command cd
```

SEE ALSO

getconf(1), sh-posix(1), confstr(2).

STANDARDS CONFORMANCE

command: XPG4, POSIX.2

NAME

compact, uncompact, ccat - compact and uncompact files, and cat them

SYNOPSIS

compact [*name* ...]

uncompact [*name* ...]

ccat [*file* ...]

DESCRIPTION

compact compresses the named files using an adaptive Huffman code. If no file names are given, standard input is compacted and sent to the standard output. **compact** operates as an on-line algorithm. Each time a byte is read, it is encoded immediately according to the current prefix code. This code is an optimal Huffman code for the set of frequencies seen so far. It is unnecessary to attach a decoding tree in front of the compressed file because the encoder and the decoder start in the same state and stay synchronized. Furthermore, **compact** and **uncompact** can operate as filters. In particular,

... **| compact | uncompact |** ...

operates as a (very slow) no-op.

When an argument *file* is given, it is compacted, the resulting file is placed in *file*.**C**, and *file* is unlinked. The first two bytes of the compacted file code the fact that the file is compacted. These bytes are used to prohibit recompaction.

The amount of compression to be expected depends on the type of file being compressed. Typical file size reduction (in percent) through compression are: Text, 38%; Pascal Source, 43%; C Source, 36%; and Binary, 19%.

uncompact restores the original file from a file compressed by **compact.** If no file names are specified, standard input is uncompacted and sent to the standard output.

ccat cats the original file from a file compressed by **compact,** without uncompressing the file.

Access Control Lists (ACLs)

On systems that implement access control lists, when a new file is created with the effective user and group ID of the caller, the original file's ACL is copied to the new file after being altered to reflect any change in ownership (see *acl*(5)).

WARNINGS

On short-filename systems, the last segment of the file name must contain 12 or fewer characters to allow space for the appended **.C**.

DEPENDENCIES

NFS

Access control list entries of networked files are summarized (as returned in **st_mode** by **stat()**), but not copied to the new file (see *stat*(2)).

FILES

***.C**	compacted file created by compact, removed by uncompact

SEE ALSO

compress(1), pack(1), acl(5).

Gallager, Robert G., "Variations on a Theme of Huffman," *I.E.E.E. Transactions on Information Theory*, vol. IT-24, no. 6, November 1978, pp. 668 - 674.

AUTHOR

compact was developed by Colin L. Mc Master.

NAME

compress, uncompress, zcat, compressdir, uncompressdir - compress and expand data

SYNOPSIS

Compress Files

compress [**-d**] [**-f|-z**] [**-z**] [**-v**] [**-c**] [**-V**] [**-b** *maxbits*] [*file* ...]

uncompress [**-f**] [**-v**] [**-c**] [**-V**] [*file* ...]

zcat [**-V**] [*file* ...]

Compress Entire Directory Subtrees

compressdir [*options*] [*directory* ...]

uncompressdir [*options*] [*directory* ...]

DESCRIPTION

The following commands compress and uncompress files and directory subtrees as indicated:

compress Reduce the size of the named *file*s using adaptive Lempel-Ziv coding. If reduction is possible, each *file* is replaced by a new file of the same name with the suffix **.Z** added to indicate that it is a compressed file. Original ownership, modes, access, and modification times are preserved. If no *file* is specified, or if **-** is specified, standard input is compressed to the standard output.

uncompress Restore the compressed *file*s to original form. Resulting files have the original filename, ownership, and permissions, and the **.Z** filename suffix is removed. If no *file* is specified, or if **-** is specified, standard input is uncompressed to the standard output.

zcat Restore the compressed *file*s to original form and send the result to standard output. If no *file* is specified, or if **-** is specified, standard input is uncompressed to the standard output.

compressdir Front-end processor. Recursively descend each specified *directory* subtree and use **compress** to compress each file in *directory*. Existing files are replaced by a compressed file having the same name plus the suffix **.Z**, provided the resulting file is smaller than the original. If no directories are specified, compression is applied to all files starting with the current directory.

options may include any valid **compress** command options (they are passed through to **compress**). To force compression of all files, even when the result is larger than the original file, use the **-f** option.

uncompressdir Opposite of **compressdir**. Restore compressed files to their original form. *options* may include any valid **uncompress** command options (they are passed through to **uncompress**).

The amount of compression obtained depends on the size of the input, the maximum number of bits (*maxbits*) per code, and the distribution of common substrings. Typically, text such as source code or English is reduced by 50-60 percent. Compression is generally much better than that achieved by Huffman coding (as used in **pack**), or adaptive Huffman coding (**compact**), and takes less time to compute.

Options

These commands recognize the following options in the combinations shown above in SYNOPSIS:

-d Decompress *file*. **compress -d** is equivalent to **uncompress**.

-f Force compression of *file*. This is useful for compressing an entire directory, even if some of the files do not actually shrink. If **-f** is not given and **compress** is run in the foreground, the user is prompted as to whether an existing file should be overwritten.

-z This is the same as the **-f** option except that it does not force compression when there is null compression.

-v Print a message describing the percentage of reduction for each file compressed.

-c Force **compress** and **uncompress** to write to the standard output; no files are changed. The nondestructive behavior of **zcat** is identical to that of

uncompress -c.

-V Print the current version and compile options onto the standard error.

-b *maxbits* Specify the maximum number of bits the **compress** algorithm will use. The default is 16 and the range can be any integer between 9 and 16.

compress uses the modified Lempel-Ziv algorithm popularized in *A Technique for High Performance Data Compression* , Terry A. Welch, *IEEE Computer,* vol. 17, no. 6 (June 1984), pages 8-19. Common substrings in the file are first replaced by 9-bit codes 257 and up. When code 512 is reached, the algorithm switches to 10-bit codes and continues to use more bits until the limit specified by the **-b** flag is reached (default 16).

After the *maxbits* limit is attained, **compress** periodically checks the compression ratio. If it is increasing, **compress** continues to use the existing code dictionary. However, if the compression ratio is decreasing, **compress** discards the table of substrings and rebuilds it from scratch. This allows the algorithm to adapt to the next "block" of the file.

Note that the **-b** flag is omitted for **uncompress** since the *maxbits* parameter specified during compression is encoded within the output, along with a magic number to ensure that neither decompression of random data nor recompression of compressed data is attempted.

Access Control Lists

compress retains a file's access control list when compressing and expanding data.

EXTERNAL INFLUENCES

Environment Variables

LC_MESSAGES determines the language in which messages are displayed.

If **LC_MESSAGES** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**.

If any internationalization variable contains an invalid setting, **compress**, **uncompress**, and **zcat** behave as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported.

RETURN VALUE

These commands return the following values upon completion:

0 Completed successfully.
2 Last file is larger after (attempted) compression.
1 An error occurred.

DIAGNOSTICS

Usage: compress [-f|-z] [-dvcV] [-b maxbits] [file ...]
Invalid options were specified on the command line.

Missing maxbits
maxbits must follow **-b**.

file: **not in compressed format**
The file specified to **uncompress** has not been compressed.

file: **compressed with** *xx***bits, can only handle** *yy***bits**
file was compressed by a program that could deal with a higher value of *maxbits* than the compress code on this machine. Recompress the file with a lower value of *maxbits*.

file: **already has .Z suffix -- no change**
The file is assumed to be already compressed. Rename the file and try again.

file: **filename too long to tack on .Z**
The output file name, which is the source file name with a **.Z** extension, is too long for the file system on which the source file resides. Make the source file name shorter and try again.

file already exists; do you wish to overwrite (y or n)?
Respond **y** if you want the output to replace the existing file; otherwise, respond **n**.

uncompress: corrupt input
A **SIGSEGV** violation was detected which usually means that the input file has been corrupted.

Compression: *xx*.*xx*%
Percentage of the input saved by compression. (Relevant only for **-v**.)

-- not a regular file: unchanged
When the input file is not a regular file (a directory for example), it is left unaltered.

-- has *xx***other links: unchanged**
The input file has links which are not symbolic links and has been left unchanged. See *ln*(1) for more information.

-- has symbolic links: unchanged
The input file has symbolic links and has been left unchanged. See *ln*(1) for more information.

-- file unchanged
No savings is achieved by compression. The input remains unaltered.

EXAMPLES

Compress the file named **zenith** and print compression information to the terminal:

```
compress -v zenith
```

The terminal display shows either a line resembling

```
zenith: Compression: 23.55% -- replaced with zenith.Z
```

indicating that the compressed file is 23.55% smaller than the original, or a line resembling

```
zenith: Compression: -12.04% -- file unchanged
```

indicating that an additional 12.04% space must be used to compress the file.

Undo the compression by typing either of the following commands:

```
uncompress zenith.Z
compress -d zenith.Z
```

This restores file **zenith.Z** to its original uncompressed form and name.

uncompress will perform on standard input if no files are specified. For example, to list a compressed tar file:

```
uncompress -c arch.tar.Z | tar -tvf -
```

WARNINGS

Although compressed files are compatible between machines with large memory, **-b**12 should be used for file transfer to architectures with a small process data space (64K bytes or less).

NFS

Access control lists of networked files are summarized (as returned in **st_mode** by **stat()**, but not copied to the new file (see *stat*(2)).

AUTHOR

compress was developed by Joseph M. Orost, Kenneth E. Turkowski, Spencer W. Thomas, and James A. Woods.

FILES

***.Z** Compressed file created by **compress** and removed by **uncompress**.

SEE ALSO

compact(1), pack(1), acl(5).

STANDARDS CONFORMANCE

compress: XPG4

NAME

convert - convert an audio file

SYNOPSIS

/opt/audio/bin/convert [*source_file*] [*target_file*] [**-sfmt** *format*] [**-dfmt** *format*] [**-ddata** *data_type*] [**-srate** *rate*] [**-drate** *rate*] [**-schannels** *number*] [**-dchannels** *number*]

DESCRIPTION

This command converts audio files from one supported file format, data format, sampling rate, and number of channels to another. The unconverted file is retained as a source file.

-sfmt *format* **-dfmt** *format*

are the file formats for the source and destination files. Each *format* can be one of these:

au	Sun file format
snd	NeXT file format
wav	Microsoft RIFF Waveform file format
u	MuLaw format
al	ALaw
l16	linear 16-bit format
lo8	offset (unsigned) linear 8-bit format
l8	linear 8-bit format

If you omit **-sfmt**, **convert** uses the header or filename extension in the source file. You can omit **-dfmt** if you supply a filename extension for the destination file.

-ddata *data_type*

is the data type for the destination files. *data_type* can be one of these:

u	MuLaw
al	ALaw
l16	linear 16-bit
lo8	offset (unsigned) linear 8-bit data
l8	linear 8-bit data

If you omit **-ddata**, **convert** uses an appropriate data type, normally the data type of the source file.

-srate *rate* **-drate** *rate*

are the number of samples per second for the source and destination file. Typical sampling rates range from 8 to 11k (for voice quality) to 44,100 (for CD quality). You can use **k** to indicate thousands. For example, **8k** means 8,000 samples per second.

If you omit **-srate**, **convert** uses a rate defined by the source file header or its filename extension. For a raw file with no extension, 8,000 is used. By playing the file, you can determine if 8,000 samples is too fast or too slow.

If you omit **-drate**, **convert** uses a sampling rate appropriate for the destination file format; if possible, it matches the sampling rate of the source file.

-schannels *number* **-dchannels** *number*

are the number of channels in the source and destination files. Use **1** for mono; **2** for stereo. If **-schannels** is omitted, **convert** uses the information in the header; for raw data files, it uses mono.

If **-dchannels** is omitted, **convert** matches what was used for the source file (through the header or **-schannels** option); for raw data files, it uses mono.

EXAMPLES

Convert a raw data file to a headered file.

```
cd /opt/audio/bin
convert  beep.l16  beep.au
```

Convert a raw data file to a headered file when the source has no extension, was sampled at 11,025 per second, and has stereo data.

```
cd /opt/audio/bin
convert  beep  beep.au -sfmt l16 -srate 11025 -schannels 2
```

To save disk space, convert an audio file with CD quality sound to voice quality sound.

```
cd /opt/audio/bin
convert idea.au  idea2.au -ddata u -drate 8k -dchannels 1
```

AUTHOR

convert was developed by HP.

Sun is a trademark of Sun MicroSystems, Inc.

NeXT is a trademark of NeXT Computers, Inc.

Microsoft is a trademark of Microsoft Corporation.

SEE ALSO

audio(5), asecure(1M), aserver(1M), attributes(1), send_sound(1).

Using the Audio Developer's Kit

NAME

cp - copy files and directory subtrees

SYNOPSIS

cp [**-f**|**-i**] [**-p**] [**-e** *extarg*] *file1 new_file*

cp [**-f**|**-i**] [**-p**] [**-e** *extarg*] *file1* [*file2* ...] *dest_directory*

cp [**-f**|**-i**] [**-p**] [**-R**|**-r**] [**-e** *extarg*] *directory1* [*directory2* ...] *dest_directory*

DESCRIPTION

cp copies:

- *file1* to new or existing *new_file*,
- *file1* to existing *dest_directory*,
- *file1*, *file2*, ... to existing *dest_directory*,
- directory subtree *directory1*, to new or existing *dest_directory*. or
- multiple directory subtrees *directory1*, *directory2*, ... to new or existing *dest_directory*.

cp fails if *file1* and *new_file* are the same (be cautious when using shell metacharacters). When destination is a directory, one or more files are copied into that directory. If two or more files are copied, the destination must be a directory. When copying a single file to a new file, if *new_file* exists, its contents are destroyed.

If the access permissions of the destination *dest_directory* or existing destination file *new_file* forbid writing, **cp** aborts and produces an error message "cannot create *file*".

To copy one or more directory subtrees to another directory, the **-r** option is required. The **-r** option is ignored if used when copying a file to another file or files to a directory.

If *new_file* is a link to an existing file with other links, **cp** overwrites the existing file and retains all links. If copying a file to an existing file, **cp** does not change existing file access permission bits, owner, or group.

When copying files to a directory or to a new file that does not already exist, **cp** creates a new file with the same file permission bits as *file1*, modified by the file creation mask of the user if the **-p** option was not specified, and then bitwise inclusively ORed with S_IRWXU. The owner and group of the new file or files are those of the user. The last modification time of *new_file* (and last access time, if *new_file* did not exist) and the last access time of the source *file1* are set to the time the copy was made.

Options

-i (interactive copy) Cause **cp** to write a prompt to standard error and wait for a response before copying a file that would overwrite an existing file. If the response from the standard input is affirmative, the file is copied if permissions allow the copy. If the **-i** (interactive) and **-f** (forced-copy) options are both specified, the **-i** option is ignored.

-f Force existing destination pathnames to be removed before copying, without prompting for confirmation. This option has the effect of destroying and replacing any existing file whose name and directory location conflicts with the name and location of the new file created by the copy operation.

-p (preserve permissions) Causes **cp** to preserve in the copy as many of the modification time, access time, file mode, user ID, and group ID as allowed by permissions.

-r (recursive subtree copy) Cause **cp** to copy the subtree rooted at each source directory to *dest_directory*. If *dest_directory* exists, it must be a directory, in which case **cp** creates a directory within *dest_directory* with the same name as *file1* and copies the subtree rooted at *file1* to *dest_directory*/*file1*. An error occurs if *dest_directory*/*file1* already exists. If *dest_directory* does not exist, **cp** creates it and copies the subtree rooted at *file1* to *dest_directory*. Note that **cp -r** cannot merge subtrees.

Usually normal files and directories are copied. Character special devices, block special devices, network special files, named pipes, symbolic links, and sockets are copied, if the user has access to the file; otherwise, a warning is printed stating that the file cannot be created, and the file is skipped.

dest_directory should not reside within *directory1*, nor should *directory1* have a cyclic directory structure, since in both cases **cp** attempts to copy an infinite amount of data.

-R (recursive subtree copy) The **-R** option is identical to the **-r** option with the exception that directories copied by the **-R** option are created with read, write, and search permission for the owner. User and group permissions remain unchanged.

With the **-R** and **-r** options, in addition to regular files and directories, **cp** also copies FIFOs, character and block device files and symbolic links. Only superusers can copy device files. All other users get an error. Symbolic links are copied so the target points to the same location that the source did.

Warning: While copying a directory tree that has device special files, use the **-r** option; otherwise, an infinite amount of data is read from the device special file and is duplicated as a special file in the destination directory occupying large file system space.

-e *extarg*
Specifies the handling of any extent attributes of the file[s] to be copied. *extarg* takes one of the following values.

warn	Issues a warning message if extent attributes cannot be copied, but copies the file anyway.
ignore	Does not copy the extent attributes.
force	Fails to copy the file if the extent attribute can not be copied.

Extent attributes can not be copied if the files are being copied to a file system which does not support extent attributes or if that file system has a different block size than the original. If **-e** is not specified, the default value for *extarg* is **warn.**

Access Control Lists (ACLs)

If *new_file* is a new file, or if a new file is created in *dest_directory*, it inherits the access control list of the original *file1*, *file2*, etc., altered to reflect any difference in ownership between the two files (see *acl*(5)).

EXTERNAL INFLUENCES

Environment Variables

LC_CTYPE determines the interpretation of text as single and/or multi-byte characters.

LANG and **LC_CTYPE** determine the local language equivalent of y (for yes/no queries).

LANG determines the language in which messages are displayed.

If **LC_CTYPE** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG.** If any internationalization variable contains an invalid setting, **cp** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported.

EXAMPLES

The following command moves the directory *sourcedir* and its contents to a new location (*targetdir*) in the file system. Since **cp** creates the new directory, the destination directory *targetdir* should not already exist.

```
cp -r sourcedir targetdir && rm -rf sourcedir
```

The **-r** option copies the subtree (files and subdirectories) in directory *sourcedir* to directory *targetdir*. The double ampersand (**&&**) causes a conditional action. If the operation on the left side of the **&&** is successful, the right side is executed (and removes the old directory). If the operation on the left of the **&&** is not successful, the old directory is not removed.

This example is equivalent to:

```
mv sourcedir targetdir
```

To copy all files and directory subtrees in the current directory to an existing *targetdir*, use:

```
cp -r * targetdir
```

To copy all files and directory subtrees in *sourcedir* to *targetdir*, use:

```
cp -r sourcedir/* targetdir
```

Note that directory pathnames can precede both *sourcedir* and *targetdir*.

To create a zero-length file, use any of the following:

```
cat /dev/null >file
cp /dev/null file
touch file
```

DEPENDENCIES

NFS

Access control lists of networked files are summarized (as returned in **st_mode** by **stat()**), but not copied to the new file. When using **mv** or **ln** on such files, a **+** is not printed after the mode value when asking for permission to overwrite a file.

AUTHOR

cp was developed by AT&T, the University of California, Berkeley, and HP.

SEE ALSO

cpio(1), ln(1), mv(1), rm(1), link(1M), lstat(2), readlink(2), stat(2), symlink(2), symlink(4), acl(5).

STANDARDS CONFORMANCE

cp: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

NAME

cpio - copy file archives in and out; duplicate directory trees

SYNOPSIS

cpio -o [**-e** *extarg*] [**achvxABC**]

cpio -i[**bcdfmrstuvxBPRSU6**] [*pattern*...]

cpio -p [**-e** *extarg*] [**adlmruvxU**] *directory*

DESCRIPTION

The **cpio** command saves and restores archives of files on magnetic tape, other devices, or a regular file, and copies files from one directory to another while replicating the directory tree structure.

cpio -o (copy out, export) Read standard input to obtain a list of path names, and copy those files to standard output together with path name and status information. The output is padded to a 512-byte boundary.

cpio -i (copy in, import) Extract files from standard input, which is assumed to be the result of a previous **cpio -o**.

If *pattern*..., is specified, only the files with names that match a *pattern* according to the rules of Pattern Matching Notation (see *regexp*(5)) are selected. A leading **!** on a pattern indicates that only those names that do *not* match the remainder of the pattern should be selected. Multiple patterns can be specified. The patterns are additive. If no *pattern* is specified, the default is ***** (select all files). See the **f** option, as well.

Extracted files are conditionally created and copied into the current directory tree, as determined by the options described below. The permissions of the files match the permissions of the original files when the archive was created by **cpio -o** unless the **U** option is used. File owner and group are that of the current user unless the user has appropriate privileges, in which case **cpio** retains the owner and group of the files of the previous **cpio -o**.

cpio -p (passthrough) Read standard input to obtain a list of path names of files which are then conditionally created and copied into the destination *directory* tree as determined by the options described below. *directory* must exist. Destination path names are interpreted relative to *directory*.

Options

cpio recognizes the following options, which can be appended as appropriate to **-i**, **-o**, and **-p**. Whitespace and hyphens are not permitted between these options and **-i**, **-o**, or **-p**.

a Reset access times of input files after they are copied.

b Swap both bytes and half-words. Use only with **-i**. See the **P** option for details; see also the **s** and **S** options.

c Write or read header information in ASCII character form for portability.

d Create directories as needed.

-e *extarg*
Specifies the handling of any extent attributes of the file(s) to be archived or copied. *extarg* takes one of the following values.

warn	Archive or copy the file and issue a warning message if extent attributes cannot be preserved.
ignore	Do not issue a warning message even if extent attributes cannot be preserved.
force	Any file(s) with extent attributes will not be archived and a warning message will be issued.

When using the **-o** option, extent attributes are not preserved in the archive. Furthermore, the **-p** option will not preserve extent attributes if the files are being copied to a file system that does not support extent attributes. If **-e** is not specified, the default value for *extarg* is **warn.**

f Copy in all files except those selected by *pattern*....

h Follow symbolic links as though they were normal files or directories. Normally, **cpio** archives the link.

l Whenever possible, link files rather than copying them. This option does not destroy existing files. Use only with **-p**.

m Retain previous file modification time. This option does not affect directories that are being copied.

r Rename files interactively. If the user types a null line, the file is skipped.

s Swap all bytes of the file. Use only with **-i**. See the **P** option for details; see also the **s** and **S** options.

t Print only a table of contents of the input. No files are created, read, or copied.

u Copy unconditionally (normally, an older file does not replace a newer file with the same name).

v Print a list of file names as they are processed. When used with the **t** option, the table of contents has the format:

numeric-mode owner-name blocks date-time filename

where *numeric-mode* is the file privileges in numeric format, *owner-name* is the name of the file owner, *blocks* is the size of the file in 512-byte blocks, *date-time* is the date and time the file was last modified, and *filename* is the path name of the file as recorded in the archive.

x Save or restore device special files. Since **mknod()** is used to recreate these files on a restore, **-ix** and **-px** can be used only by users with appropriate privileges (see *mknod*(2)). This option is intended for intrasystem (backup) use only. Restoring device files from previous versions of the OS, or from different systems can be very dangerous. **cpio** may prevent the restoration of certain device files from the archive.

A Suppress warning messages regarding optional access control list entries. **cpio** does not back up optional access control list entries in a file's access control list (see *acl*(5)). Normally, a warning message is printed for each file that has optional access control list entries.

B Block input/output at 5120 bytes to the record (does not apply to **cpio -p**). This option is meaningful only with data directed to or from devices that support variable-length records such as magnetic tape.

C Have **cpio** checkpoint itself at the start of each volume. If **cpio** is writing to a streaming tape drive with immediate-report mode enabled and a write error occurs, it normally aborts and exits with return code **2**. With this option specified, **cpio** instead automatically restarts itself from the checkpoint and rewrites the current volume. Alternatively, if **cpio** is not writing to such a device and a write error occurs, **cpio** normally continues with the next volume. With this option specified, however, the user can choose to either ignore the error or rewrite the current volume.

P Read a file written on a PDP-11 or VAX system (with byte-swapping) that did not use the **c** option. Use only with **-i**. Files copied in this mode are not changed. Non-ASCII files are likely to need further processing to be readable. This processing often requires knowledge of file contents, and thus cannot always be done by this program. The **b**, **s**, and **S** options can be used when swapping all the bytes on the tape (rather than just in the headers) is appropriate. In general, text is best processed with **P** and binary data with one of the other options.

(PDP-11 and VAX are registered trademarks of Digital Equipment Corporation.)

R Resynchronize automatically when **cpio** goes "out of phase", (see DIAGNOSTICS).

S Swap all half-words in the file. Use only with **-i**. See the **P** option for details; see also the **b** and **s** options.

U Use the process's file-mode creation mask (see *umask*(2)) to modify the mode of files created, in the same manner as *creat*(2).

6 Process a UNIX Sixth-Edition-format file. Use only with **-i**.

Note that **cpio** archives created using a raw device file must be read using a raw device file.

When the end of the tape is reached, **cpio** prompts the user for a new special file and continues.

If you want to pass one or more metacharacters to **cpio** without the shell expanding them, be sure to precede each of them with a backslash (\).

Device files written with the **-ox** option (such as **/dev/tty03**) do not transport to other implementations of HP-UX.

EXTERNAL INFLUENCES

Environment Variables

LC_COLLATE determines the collating sequence used in evaluating pattern matching notation for file name generation.

LC_CTYPE determines the interpretation of text as single and/or multi-byte characters, and the characters matched by character class expressions in pattern matching notation.

LC_TIME determines the format and content of date and time strings output when listing the contents of an archive with the **v** option.

LANG determines the language in which messages are displayed.

If **LC_COLLATE**, **LC_CTYPE**, or **LC_TIME** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**. If any internationalization variable contains an invalid setting, **cpio** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported.

RETURN VALUE

cpio returns the following exit codes:

0	Successful completion. Review standard error for files that could not be transferred.
1	Error during resynchronization. Some files may not have been recovered.
2	Out-of-phase error. A file header is corrupt or in the wrong format.

DIAGNOSTICS

Out of phase--get help
Perhaps the "c" option should[n't] be used

cpio -i could not read the header of an archived file. The header is corrupt or it was written in a different format. Without the **R** option, **cpio** returns an exit code of **2**.

If no file name has been displayed yet, the problem may be the format. Try specifying a different header format option: null for standard format; **c** for ASCII; **b**, **s**, **P**, or **S**, for one of the byte-swapping formats; or **6** for UNIX Sixth Edition.

Otherwise, a header may be corrupt. Use the **R** option to have **cpio** attempt to resynchronize the file automatically. Resynchronizing means that **cpio** tries to find the next good header in the archive file and continues processing from there. If **cpio** tries to resynchronize from being out of phase, it returns an exit code of **1**.

Other diagnostic messages are self-explanatory.

EXAMPLES

Copy the contents of a directory into a tape archive:

```
ls | cpio -o > /dev/rmt/c0t0d0BEST
```

Duplicate a directory hierarchy:

```
cd olddir
find . -depth -print | cpio -pd newdir
```

The trivial case

```
find . -depth -print | cpio -oB >/dev/rmt/c0t0d0BEST
```

can be handled more efficiently by:

```
find . -cpio /dev/rmt/c0t0d0BEST
```

WARNINGS

Because of industry standards and interoperability goals, **cpio** does not support the archival of files larger than 2GB or files that have user/group IDs greater than 60K. Files with user/group IDs greater than 60K are archived and restored under the user/group ID of the current process.

Do not redirect the output of **cpio** to a named **cpio** archive file residing in the same directory as the original files belonging to that **cpio** archive. This can cause loss of data.

cpio strips any leading **./** characters in the list of filenames piped to it.

Path names are restricted to **PATH_MAX** characters (see **<limits.h>** and *limits*(5)). If there are too many unique linked files, the program runs out of memory to keep track of them. Thereafter, linking information is lost. Only users with appropriate privileges can copy special files.

cpio tapes written on HP machines with the **-ox**[**c**] options can sometimes mislead (non-HP) versions of **cpio** that do not support the **x** option. If a non-HP (or non-AT&T) version of **cpio** happens to be modified so that the (HP) **cpio** recognizes it as a device special file, a spurious device file might be created.

If **/dev/tty** is not accessible, **cpio** issues a complaint and exits.

The **-pd** option does not create the directory typed on the command line.

The **-idr** option does not make empty directories.

The **-plu** option does not link files to existing files.

POSIX defines a file named **TRAILER!!!** as an end-of-archive marker. Consequently, if a file of that name is contained in a group of files being written by **cpio -o**, the file is interpreted as end-of-archive, and no remaining files are copied. The recommended practice is to avoid naming files anything that resembles an end-of-archive file name.

To create a POSIX-conforming **cpio** archive, the **c** option must be used. To read a POSIX-conforming **cpio** archive, the **c** option must be used and the **b**, **s**, **S**, and **6** options should not be used. If the user does not have appropriate privileges, the **U** option must also be used to get POSIX-conforming behavior when reading an archive. Users with appropriate privileges should not use this option to get POSIX-conforming behavior.

Using Cartridge Tape Drives

For an explanation of the constraints on cartridge tapes, see *ct*(7).

Using **cpio** to write directly to a cartridge tape unit can severely damage the tape drive in a short amount of time, and is therefore strongly discouraged. The recommended method of writing to the cartridge tape unit is to use the **tcio** command (see *tcio*(1)) in conjunction with **cpio** (note that the **B** option must not be used by **cpio** when **tcio** is used). **tcio** buffers data into larger pieces suitable for cartridge tapes. The **B** option must be used when writing directly (that is, without using **tcio**) to a CS/80 cartridge tape unit.

DEPENDENCIES

If the path given to **cpio** contains a symbolic link as the last element, this link is traversed and pathname resolution continues. **cpio** uses the symbolic link's target, rather than that of the link.

SEE ALSO

ar(1), find(1), tar(1), tcio(1), cpio(4), acl(5), environ(5), lang(5), regexp(5).

STANDARDS CONFORMANCE

cpio: SVID2, SVID3, XPG2, XPG3

NAME

cpp - the C language preprocessor

SYNOPSIS

/usr/ccs/lbin/cpp [*option* ...] [*ifile* [*ofile*]]

DESCRIPTION

cpp is the C language preprocessor which is invoked as the first pass of any C compilation using the **cc** command (see *cc*(1)). Its purpose is to process **#include** and conditional compilation instructions and macros. Thus the output of **cpp** is designed to be in a form acceptable as input to the next pass of the C compiler. As the C language evolves, **cpp** and the rest of the C compilation package will be modified to follow these changes. Therefore, the use of **cpp** in other than this framework is not suggested. The preferred way to invoke **cpp** is through the **cc** command, since the functionality of **cpp** may someday be moved elsewhere. See *m4*(1) for a general macro processor.

cpp optionally accepts two file names as arguments. *ifile* and *ofile* are respectively the input and output for the preprocessor. They default to standard input and standard output if not specified.

Options

The following options are recognized by **cpp**:

-A Remove all predefined symbols that begin with a letter and **_HPUX_SOURCE**. The user is expected to define **_POSIX_SOURCE** or **_XOPEN_SOURCE** when using this option.

-C By default, **cpp** strips C-style comments. If the **-C** option is specified, all comments (except those found on **cpp** directive lines) are passed along.

-D*name*
-D*name***=***def* Define *name* as if by a **#define** directive. If no **=***def* is given, *name* is defined as **1**. The **-D** option has lower precedence than the **-U** option. Thus, if the same name is used in both a **-U** option and a **-D** option, the name is undefined regardless of the order of the options.

-H*nnn* Change the internal macro definition table to be *nnn* bytes in size. The default buffer size is at least 8 188 bytes. This option serves to eliminate “Macro param too large”, “Macro invocation too large”, “Macro param too large after substitution”, “Quoted macro param too large”, “Macro buffer too small”, “Input line too long”, and “Catenated input line too long” errors.

-h[*inclfile*] Generates included files and sents the results to the file *inclfile*. If the argument *inclfile* is omitted, the result is sent to the standard error.

-I*dir* Change the algorithm for searching for **#include** files whose names do not begin with **/** to look in *dir* before looking in the directories on the standard list. Thus, **#include** files whose names are enclosed in double quotes (**""**) are searched for first in the directory of the file containing the **#include** line, then in directories named in **-I** options in left-to-right order, and last in directories on a standard list. For **#include** files whose names are enclosed in angle brackets (**<>**), the directory of the file containing the **#include** line is not searched. However, directory *dir* is still searched.

-M[*makefile*] Generates makefile dependencies and sends the results to the file *makefile*. If the argument *makefile* is omitted, the result is sent to the standard error.

-P Preprocess the input without producing the line-control information used by the next pass of the C compiler.

-T HP-UX no longer restricts preprocessor symbols to eight characters. The **-T** option forces **cpp** to use only the first eight characters for distinguishing different preprocessor names. This behavior is the same as preprocessors on some other systems with respect to the length of names, and is included for backward compatibility.

-U*name* Remove any initial definition of *name*, where *name* is a reserved symbol that is predefined by the particular preprocessor. The current list of these symbols includes:

Operating system:	**unix**	**__unix**	
Hardware:	**hp9000s200**	**hp9000s300**	**__hp9000s300**
	hp9000s500	**hp9000s800**	**__hp9000s800**
	hp9000ipc	**hppa**	**__hppa**

	_PA_RISC1_0	**_PA_RISC1_1**	**_SIO**	**_WSIO**
UNIX systems variant:	**hpux** **PWB**	**__hpux** **_PWB**	**_HPUX_SOURCE**	
lint(1):	**lint**	**__lint**		

In addition, all symbols that begin with an underscore and either an upper-case letter or another underscore are reserved. Other symbols may be defined by the **CCOPTS** variable or other command-line options to the C compiler at compile time (see *cc*(1)). All HP-UX systems have the symbols **PWB**, **hpux**, **unix**, **_PWB, __hpux**, and **__unix** defined. Each system defines at least one hardware variant, as appropriate. The lint symbols are defined when *lint*(1) is running. See DEPENDENCIES.

Two special names are understood by **cpp**. **__LINE__** is defined as the current line number (as a decimal integer) as counted by **cpp**. **__FILE__** is defined as the current file name (as a C string) as known by **cpp**. They can be used anywhere (including in macros) just as any other defined names.

Directives

All **cpp** directives start with lines begun by **#**. Any number of blanks and tabs are allowed between the **#** and the directive. The directives are:

#define *name token-string*

Replace subsequent instances of *name* with *token-string*. *token-string* can be null.

#define *name***(***arg***, ... ,** *arg***)** *token-string*

Replace subsequent instances of *name* followed by a **(**, a list of comma-separated set of arguments, and a **)** by *token-string*, where each occurrence of an *arg* in the *token-string* is replaced by the corresponding set of tokens in the comma-separated list. When a macro with arguments is expanded, the arguments are placed into the expanded *token-string* unchanged. After the entire *token-string* has been expanded, **cpp** restarts its scan for names to expand at the beginning of the newly created *token-string*.

Notice that there can be no space between *name* and the **(**.

#endif [*text***]**

Ends a section of lines begun by a test directive (**#if**, **#ifdef**, or **#ifndef**). Each test directive must have a matching **#endif**. Any *text* occurring on the same line as the **#endif** is ignored and thus may be used to mark matching **#if**–**#endif** pairs. This makes it easier, when reading the source, to match **#if**, **#ifdef**, and **#ifndef** directives with their associated **#endif** directive.

#elif *constant-expression*

Equivalent to:

#else
#if *constant-expression*

#else

Reverses the notion of the test directive that matches this directive. Thus, if lines previous to this directive are ignored, the following lines appear in the output, and vice versa.

#if *constant-expression*

The lines following appear in the output if and only if the *constant-expression* evaluates to nonzero. All binary nonassignment C operators, the **?:** operator, the unary **-**, **!**, and **~** operators are all legal in *constant-expression*. The precedence of the operators is the same as defined by the C language.

There is also a unary operator, **defined**, which can be used in *constant-expression* in these two forms: **defined(***name***)** or **defined** *name*. This allows the use of **#ifdef** and **#ifndef** in an **#if** directive.

Only these operators, integer constants, and names that are known by **cpp** should be used in *constant-expression*. In particular, the **sizeof** operator is not available.

#ifdef *name*

The lines following appear in the output if and only if *name* has been the subject of a previous **#define** without being the subject of an intervening **#undef**.

#ifndef *name*

The lines following do not appear in the output if and only if *name* has been the subject of a previous **#define** without being the subject of an intervening **#undef**.

#include "*filename***"**
#include <*filename***>**

Include at this point the contents of *filename* (which are then run through **cpp**). See the **-I** option above for more detail.

#line *integer-constant* **"***filename***"**

Causes **cpp** to generate line-control information for the next pass of the C compiler. *integer-constant* is the line number of the next line and *filename* is the file where it comes from. If *filename* and the quotation marks are omitted, the current file name is unchanged.

#undef *name*

Cause the definition of *name* (if any) to be forgotten from now on.

The test directives and the possible **#else** directives can be nested. **cpp** supports names up to 255 characters in length.

EXTERNAL INFLUENCES

Environment Variables

LC_CTYPE determines the interpretation of comments and string literals as single- or multibyte characters.

LANG determines the language in which messages are displayed.

If **LC_CTYPE** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, it defaults to "C" (see *lang*(5)). If any internationalization variable contains an invalid setting, **cpp** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multibyte character code sets are supported.

DIAGNOSTICS

Error messages produced by **cpp** are intended to be self-explanatory. The line number and filename where the error occurred are printed along with the diagnostic.

WARNINGS

When newline characters were found in argument lists for macros to be expanded, previous versions of **cpp** put out the newlines as they were found and expanded. The current version of **cpp** replaces these newlines with blanks to alleviate problems that the previous versions had when this occurred.

DEPENDENCIES

Series 700

The symbols **hp9000s700** and **__hp9000s700** are not reserved symbols recognized by the **-U** option. They are supplied to **cpp** either automatically by the compiler, or by the use of a compiler option. For example, on a Series 700 system, the command:

```
cc -v t.c
```

produces:

```
/usr/ccs/lbin/cpp     t.c     /var/tmp/ctmAAAa29220     -D__hp9000s700
-D__hp9000s800 ...
```

(Also see the **-D** option of the **cc** command.)

FILES

/usr/include Standard directory for **#include** files

SEE ALSO

cc(1), m4(1).

NOTES

The macro substitution scheme has been changed. Previous versions of **cpp** saved macros in a macro definition table whose table size is 128 000 bytes by default. The current version of **cpp** replaces this macro definition table with several small buffers. The default size of the small buffers is 8 188 bytes.

STANDARDS CONFORMANCE

cpp: SVID2, SVID3, XPG2

NAME

crontab - user job file scheduler

SYNOPSIS

crontab [*file*]

crontab -e [username]

crontab -l [username]

crontab -r [username]

DESCRIPTION

The **crontab** command manages a crontab file for the user. You can use a crontab file to schedule jobs that are executed automatically by **cron** (see *cron*(1M)) on a regular basis. The command has four forms:

crontab [*file*] Create or replace your crontab file by copying the specified *file*, or standard input if *file* is omitted or - is specified as *file* , into the crontab directory, **/var/spool/cron/crontabs**. The name of your crontab file in the crontab directory is the same as your effective user name.

crontab -e [username]
Edit a copy of the user's crontab file, or create an empty file to edit if the crontab file does not exist. When editing is complete, the file will be copied into the crontab directory as the user's crontab file.

crontab -l [username]
Lists the user's crontab file.

crontab -r [username]
Remove the user's crontab file from the crontab directory.

Only a privileged user can use username following the -e, -l, or -r options, to edit, list, or remove the crontab file of the specified user.

The entries in a crontab file are lines of six fields each. The fields are separated by spaces or tabs. The lines have the following format:

minute hour monthday month weekday command

The first five are integer patterns that specify when the sixth field, *command*, should be executed. They can have the following ranges of values:

minute	The minute of the hour, **0**–**59**
hour	The hour of the day, **0**–**23**
monthday	The day of the month, **1**–**31**
month	The month of the year, **1**–**12**
weekday	The day of the week, **0**–**6**, **0**=Sunday

Each pattern can be either an asterisk (*****), meaning all legal values, or a list of elements separated by commas. An element is either a number in the ranges shown above, or two numbers in the range separated by a hyphen (meaning an inclusive range). Note that the specification of days can be made in two fields: *monthday* and *weekday*. If both are specified in an entry, they are cumulative. For example,

```
0   0   1,15   *   1    command
```

runs *command* at midnight on the first and fifteenth of each month, as well as every Monday. To specify days in only one field, set the other field to asterisk (*****). For example,

```
0   0   *   *   1    command
```

runs *command* only on Mondays.

The sixth field, *command* (the balance of a line including blanks in a crontab file), is a string that is executed by the shell at the specified times. A percent character (**%**) in this field (unless escaped by a backslash (\)) is translated to a newline character, dividing the field into "lines". Only the first "line" (up to a **%** or end-of-line) of the command field is executed by the shell. Any other "lines" are made available to the command as standard input.

Blank lines and those whose first non-blank character is # will be ignored.

cron invokes the command from the user's **HOME** directory with the POSIX shell, (**/usr/bin/sh**). It runs in the **c** queue (see *queuedefs*(4)).

cron supplies a default environment for every shell, defining:

```
HOME=user's-home-directory
LOGNAME=user's-login-id
PATH=/usr/bin:/usr/sbin:.
SHELL=/usr/bin/sh
```

Users who desire to have their **.profile** executed must explicitly do so in the crontab entry or in a script called by the entry.

You can execute **crontab** if your name appears in the file **/usr/lib/cron/cron.allow**. If that file does not exist, you can use **crontab** if your name does not appear in the file **/usr/lib/cron/cron.deny**. If only **cron.deny** exists and is empty, all users can use **crontab**. If neither file exists, only the **root** user can use **crontab**. The **allow**/**deny** files consist of one user name per line.

EXTERNAL INFLUENCES

Environment Variables

LC_CTYPE determines the interpretation of text within file as single- and/or multi-byte characters.

LC_MESSAGES determines the language in which messages are displayed.

If **LC_CTYPE** or **LC_MESSAGES** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**.

If any internationalization variable contains an invalid setting, **crontab** behaves as if all internationalization variables are set to "C". See *environ*(5). **EDITOR** determines the editor to be invoked when **-e** option is specified. The default editor is vi.

International Code Set Support

Single-byte and multi-byte character code sets are supported.

WARNINGS

Be sure to redirect the standard output and standard error from commands. If this is not done, any generated standard output or standard error is mailed to the user.

FILES

/var/adm/cron	Main cron directory
/var/adm/cron/cron.allow	List of allowed users
/var/adm/cron/cron.deny	List of denied users
/var/adm/cron/log	Accounting information
/var/spool/cron/crontabs	Directory containing the crontab files

SEE ALSO

sh(1), cron(1M), queuedefs(4).

STANDARDS CONFORMANCE

crontab: SVID2, SVID3, XPG2, XPG3, XPG4

NAME

crypt - encode/decode files

SYNOPSIS

crypt [*password*]

DESCRIPTION

crypt reads from the standard input and writes on the standard output. *password* is a key that selects a particular transformation. If no *password* is given, **crypt** demands a key from the terminal and turns off printing while the key is being typed in. **crypt** encrypts and decrypts with the same key:

crypt *key* **<***clear* **>***cypher*
crypt *key* **<***cypher* **|pr**

The latter command decrypts the file and prints the clear version.

Files encrypted by **crypt** are compatible with those treated by the **ed** editor in encryption mode (see *ed*(1)).

Security of encrypted files depends on three factors: the fundamental method must be hard to solve; direct search of the key space must be infeasible; "sneak paths" by which keys or clear text can become visible must be minimized.

crypt implements a one-rotor machine designed along the lines of the German Enigma, but with a 256-element rotor. Methods of attack on such machines are known, but not widely; moreover the amount of work required is likely to be large.

The transformation of a key into the internal settings of the machine is deliberately designed to be expensive; i.e., to take a substantial fraction of a second to compute. However, if keys are restricted to, for example, three lowercase letters, then encrypted files can be read by expending only a substantial fraction of five minutes of machine time.

Since the key is an argument to the **crypt** command, it is potentially visible to users executing the **ps** or a derivative (see *ps*(1)). The choice of keys and key security are the most vulnerable aspect of **crypt**.

EXAMPLES

The following example demonstrates the use of **crypt** to edit a file that the user wants to keep strictly confidential:

```
$ crypt <plans >plans.x
key: violet
$ rm plans
   ...
$ vi -x plans.x
key: violet
   ...
:wq
$
   ...
$ crypt <plans.x | pr
key: violet
```

Note that the **-x** option is the encryption mode of **vi**, and prompts the user for the same key with which the file was encrypted.

WARNINGS

If output is piped to **nroff** and the encryption key is *not* given on the command line, **crypt** can leave terminal modes in a strange state (see *nroff*(1) and *stty*(1)).

If two or more files encrypted with the same key are concatenated and an attempt is made to decrypt the result, only the the first of the original files is decrypted correctly.

FILES

/dev/tty	for typed key

SEE ALSO

ed(1), makekey(1), stty(1).

NAME

csh - a shell (command interpreter) with C-like syntax

SYNOPSIS

csh [**-cefinstvxTVX**] [*command_file*] [*argument_list* ...]

DESCRIPTION

csh is a command language interpreter that incorporates a command history buffer, C-like syntax, and job control facilities.

Command Options

Command options are interpreted as follows:

-c	Read commands from the (single) following argument which must be present. Any remaining arguments are placed in **argv**.
-e	C shell exits if any invoked command terminates abnormally or yields a non-zero exit status.
-f	Suppress execution of the **.cshrc** file in your home directory, thus speeding up shell start-up time.
-i	Force **csh** to respond interactively when called from a device other than a computer terminal (such as another computer). **csh** normally responds non-interactively. If **csh** is called from a computer terminal, it always responds interactively, regardless of which options are selected.
-n	Parse but do *not* execute commands. This is useful for checking syntax in shell scripts. All substitutions are performed (history, command, alias, etc.).
-s	Take command input from the standard input.
-t	Read and execute a single line of input.
-v	Set the **verbose** shell variable, causing command input to be echoed to the standard output device after history substitutions are made.
-x	Set the **echo** shell variable, causing all commands to be echoed to the standard error immediately before execution.
-T	Disable the tenex features which use the ESC key for command/file name completion and CTRL-**D** for listing available files (see the *CSH UTILITIES* section below)
-V	Set the **verbose** variable before **.cshrc** is executed so that all **.cshrc** commands are also echoed to the standard output.
-X	Set the **echo** variable before **.cshrc** is executed so that all **.cshrc** commands are also echoed to the standard output.

After processing the command options, if arguments remain in the argument list, and the **-c**, **-i**, **-s**, or **-t** options were not specified, the first remaining argument is taken as the name of a file of commands to be executed.

COMMANDS

A simple command is a sequence of words, the first of which specifies the command to be executed. A sequence of simple commands separated by vertical bar (**|**) characters forms a pipeline. The output of each command in a pipeline becomes the input for the next command in the pipeline. Sequences of pipelines can be separated by semicolons (**;**) which causes them to be executed sequentially. A sequence of pipelines can be executed in background mode by adding an ampersand character (**&**) after the last entry.

Any pipeline can be placed in parentheses to form a simple command which, in turn, can be a component of another pipeline. Pipelines can also be separated by **||** or **&&** indicating, as in the C language, that the second pipeline is to be executed only if the first fails or succeeds, respectively.

Jobs

csh associates a **job** with each pipeline and keeps a table of current jobs (printed by the **jobs** command) and assigns them small integer numbers. When a job is started asynchronously using **&**, the shell prints a line resembling:

```
[1] 1234
```

indicating that the job which was started asynchronously was job number 1 and had one (top-level) process, whose process id was 1234.

If you are running a job and want to do something else, you can type the currently defined *suspend* character (see *termio*(7)) which sends a stop signal to the current job. **csh** then normally indicates that the job has been ‘Stopped’, and prints another prompt. You can then manipulate the state of this job, putting it in the background with the **bg** command, run some other commands, and then eventually bring the job back into the foreground with the foreground command **fg**. A *suspend* takes effect immediately and is like an interrupt in that pending output and unread input are discarded when it is typed. There is a delayed *suspend* character which does not generate a stop signal until a program attempts to *read*(2) it. This can usefully be typed ahead when you have prepared some commands for a job which you want to stop after it has read them.

A job being run in the background stops if it tries to read from the terminal. Background jobs are normally allowed to produce output, but this can be disabled by giving the command **stty tostop** (see *stty*(1)). If you set this tty option, background jobs stop when they try to produce output, just as they do when they try to read input. Keyboard signals and line-hangup signals from the terminal interface are not sent to background jobs on such systems. This means that background jobs are immune to the effects of logging out or typing the interrupt, quit, suspend, and delayed suspend characters (see *termio*(7)).

There are several ways to refer to jobs in the shell. The character **%** introduces a job name. If you wish to refer to job number 1, you can name it as **%1**. Just naming a job brings it to the foreground; thus **%1** is a synonym for **fg %1 ,** bringing job 1 back into the foreground. Similarly, typing **%1 &** resumes job 1 in the background. Jobs can also be named by prefixes of the string typed in to start them if these prefixes are unambiguous; thus **%ex** normally restarts a suspended *ex*(1) job, if there is only one suspended job whose name begins with the string **ex**. It is also possible to say **%?string** which specifies a job whose text contains *string*, if there is only one such job.

csh maintains a notion of the current and previous jobs. In output pertaining to jobs, the current job is marked with a **+** and the previous job with a **-**. The abbreviation **%+** refers to the current job and **%-** refers to the previous job. For close analogy with the syntax of the **history** mechanism (described below), **%%** is also a synonym for the current job.

csh learns immediately whenever a process changes state. It normally informs you whenever a job becomes blocked so that no further progress is possible, but only just before printing a prompt. This is done so that it does not otherwise disturb your work. If, however, you set the shell variable **notify**, **csh** notifies you immediately of changes in status of background jobs. There is also a **csh** built-in command called **notify** which marks a single process so that any status change is immediately reported. By default, **notify** marks the current process. Simply type **notify** after starting a background job to mark it.

If you try to leave the shell while jobs are stopped, **csh** sends the warning message: **You have stopped jobs.** Use the **jobs** command to see what they are. If you do this or immediately try to exit again, **csh** does not warn you a second time, and the suspended jobs are terminated (see *exit*(2)).

Built-In Commands

Built-in commands are executed within the shell without spawning a new process. If a built-in command occurs as any component of a pipeline except the last, it is executed in a subshell. The built-in commands are:

alias
alias *name*
alias *name wordlist*

The first form prints all aliases. The second form prints the alias for *name*. The third form assigns the specified *wordlist* as the alias of *name*. Command and file name substitution are performed on *wordlist*. *name* cannot be **alias** or **unalias**.

bg [**%***job* ...]

Put the current (*job* not specified) or specified jobs into the background, continuing them if they were stopped.

break Causes execution to resume after the **end** of the nearest enclosing **foreach** or **while**. The remaining commands on the current line are executed. Multi-level breaks are thus possible by writing them all on one line.

breaksw
Causes a break from a **switch**, resuming after the **endsw**.

case *label***:**
A label in a **switch** statement as discussed below.

cd
cd *directory_name*
chdir
chdir *directory_name*
Change the shell's current working directory to *directory_name*. If not specified, *directory_name* defaults to your home directory.
If *directory_name* is not found as a subdirectory of the current working directory (and does not begin with **/**, **./**, or **../**), each component of the variable *cdpath* is checked to see if it has a subdirectory *directory_name*. Finally, if all else fails, **csh** treats *directory_name* as a shell variable. If its value begins with **/**, this is tried to see if it is a directory.

continue
Continue execution of the nearest enclosing **while** or **foreach**. The rest of the commands on the current line are executed.

default:
Labels the default case in a **switch** statement. The default should come after all other **case** labels.

dirs Prints the directory stack; the top of the stack is at the left; the first directory in the stack is the current directory.

echo *wordlist*
echo -n *wordlist*
The specified words are written to the shell's standard output, separated by spaces, and terminated with a new-line unless the **-n** option is specified.

else
end
endif
endsw See the descriptions of the **foreach**, **if**, **switch**, and **while** statements below.

eval *arguments ...*
(Same behavior as *sh*(1).) *arguments* are read as input to the shell and the resulting command(s) executed. This is usually used to execute commands generated as the result of command or variable substitution, since parsing occurs before these substitutions.

exec *command*
The specified command is executed in place of the current shell.

exit
exit (*expression***)**
csh exits either with the value of the **status** variable (first form) or with the value of the specified *expression* (second form).

fg [**%***job ...*]
Brings the current (*job* not specified) or specified jobs into the foreground, continuing them if they were stopped.

foreach *name* **(***wordlist***)**
...
end The variable *name* is successively set to each member of *wordlist* and the sequence of commands between this command and the matching **end** are executed. (Both **foreach** and **end** must appear alone on separate lines.)

The built-in command **continue** can be used to continue the loop prematurely; the built-in command **break** to terminate it prematurely. When this command is read from the terminal, the loop is read once, prompting with **?** before any statements in the loop are executed. If you make a mistake while typing in a loop at the terminal, use the erase or line-kill character as appropriate to recover.

glob *wordlist*
Like **echo** but no \ escapes are recognized and words are delimited by null characters in the output. Useful in programs that use the shell to perform file name expansion on a list of words.

goto *word*

The specified *word* is file name and command expanded to yield a string of the form **label**. The shell rewinds its input as much as possible and searches for a line of the form **label:** possibly preceded by blanks or tabs. Execution continues after the specified line.

hashstat

Print a statistics line indicating how effective the internal hash table has been at locating commands (and avoiding **exec**s). An **exec** is attempted for each component of the *path* where the hash function indicates a possible hit, and in each component that does not begin with a **/**.

history [**-h**][**-r**][*n*]

Displays the history event list. If *n* is given, only the *n* most recent events are printed. The **-r** option reverses the order of printout to be most recent first rather than oldest first. The **-h** option prints the history list without leading numbers for producing files suitable for the **source** command.

if (*expression***)** *command*

If *expression* evaluates true, the single *command* with arguments is executed. Variable substitution on *command* happens early, at the same time it does for the rest of the **if** command. *command* must be a simple command; not a pipeline, a command list, a parenthesized command list, or an aliased command. Input/output redirection occurs even if *expression* is false, meaning that *command* is *not* executed (this is a bug).

if (*expression1***) then**
. . .
else if (*expression2***) then**
. . .
else
. . .
endif If *expression1* is true, all commands down to the first **else** are executed; otherwise if *expression2* is true, all commands from the first **else** down to the second **else** are executed, etc. Any number of **else**-**if** pairs are possible, but only one **endif** is needed. The **else** part is likewise optional. (The words **else** and **endif** must appear at the beginning of input lines. The **if** must appear alone on its input line or after an **else**.)

jobs [**-l**]

Lists active jobs. The **-l** option lists process IDs in addition to the usual information.

kill % *job*
kill - *sig* **%** *job* ...
kill *pid*
kill - *sig pid*...
kill -l

Sends either the TERM (terminate) signal or the specified signal to the specified jobs or processes. Signals are either given by number or by names (as given in **/usr/include/signal.h**, stripped of the **SIG** prefix (see *signal*(2)). The signal names are listed by **kill -l**. There is no default, so **kill** used alone does not send a signal to the current job. If the signal being sent is TERM (terminate) or HUP (hangup), the job or process is sent a CONT (continue) signal as well.

limit[**-h**][*resource*][*maximum_use*]

Limits the usage by the current process and each process it creates not to (individually) exceed *maximum_use* on the specified *resource*. If *maximum_use* is not specified, then the current limit is displayed; if *resource* is not specified, then all limitations are given.

If the **-h** flag is specified, the hard limits are used instead of the current limits. The hard limits impose a ceiling on the values of the current limits. Only the superuser can raise the hard limits, but a user can lower or raise the current limits within the legal range.

Controllable resources currently include:

addresspace	Maximum address space in bytes for a process
coredumpsize	Size of the largest core dump that is created
cputime	Maximum number of CPU seconds to be used by each process

datasize	Maximum growth of the data region allowed beyond the end of the program text
descriptors	Maximum number of open files for each process
filesize	Largest single file that can be created
memoryuse	Maximum size to which a process's resident set size can grow
stacksize	Maximum size of the automatically extended stack region

The *maximum_use* argument can be specified as a floating-point or integer number followed by a scale factor: *k* or *kilobytes* (1024 bytes), *m* or *megabytes*, or *b* or *blocks* (the units used by the *ulimit* system call). For both *resource* names and scale factors, unambiguous prefixes of the names can be used. *filesize* can be lowered by an instance of *csh*, but can only be raised by an instance whose effective user ID is *root*. For more information, refer to the documentation for the *ulimit* system call.

login Terminates a login shell, replacing it with an instance of **/usr/bin/login**. This is one way to log off, included for compatibility with *sh*(1).

logout
Terminates a login shell. Especially useful if *ignoreeof* is set. A similar function, **bye**, which works for sessions that are not login shells, is provided for historical reasons. Its use is not recommended because it is not part of the standard BSD **csh** and may not be supported in future releases.

newgrp
Changes the group identification of the caller; for details see *newgrp*(1). A new shell is executed by **newgrp** so that the current shell environment is lost.

nice
nice +*number*
nice *command*
nice +*number command*
The first form sets the *nice* (run command priority) for this shell to 4 (the default). The second form sets the priority to the given *number*. The final two forms run *command* at priority 4 and *number* respectively. The user with appropriate privileges can raise the priority by specifying negative niceness using **nice -***number* ... *command* is always executed in a sub-shell, and restrictions placed on commands in simple **if** statements apply.

nohup [*command*]
Without an argument, **nohup** can be used in shell scripts to cause hangups to be ignored for the remainder of the script. With an argument, causes the specified *command* to be run with hangups ignored. All processes executed in the background with **&** are effectively **nohup**ed as described under Jobs in the *COMMANDS* section.

notify [*job* ...]
Causes the shell to notify the user asynchronously when the status of the current (*job* not specified) or specified jobs changes; normally notification is presented before a prompt. This is automatic if the shell variable *notify* is set.

onintr [**-**] [*label*]
Controls the action of the shell on interrupts. With no arguments, *onintr* restores the default action of the shell on interrupts, which action is to terminate shell scripts or return to the terminal command input level. If **-** is specified, all interrupts are ignored. If a *label* is given, the shell executes a **goto** *label* when an interrupt is received or a child process terminates because it was interrupted.

If the shell is running in the background and interrupts are being ignored, *onintr* has no effect; interrupts continue to be ignored by the shell and all invoked commands.

popd [**+***n*]
Pops the directory stack, returning to the new top directory. With an argument, discards the *n* th entry in the stack. The elements of the directory stack are numbered from 0 starting at the top. A synonym for **popd**, called **rd**, is provided for historical reasons. Its use is not recommended because it is not part of the standard BSD **csh** and may not be supported in future releases.

pushd [*name*] [**+***n*]
With no arguments, *pushd* exchanges the top two elements of the directory stack. Given a *name* argument, *pushd* changes to the new directory (using *cd*) and pushes the old current working directory (as in *csw*) onto the directory stack. With a numeric argument, *pushd* rotates the *n* th argument of the directory stack around to be the top element and changes to that directory. The members of the directory stack are numbered from the top starting at 0. A synonym for *pushd*, called *gd*, is provided for historical reasons. Its use is not recommended since it is not part of the standard BSD **csh** and may not be supported in future releases.

rehash
Causes the internal hash table of the contents of the directories in the *path* variable to be recomputed. This is needed if new commands are added to directories in the *path* while you are logged in. This should only be necessary if you add commands to one of your own directories or if a systems programmer changes the contents of one of the system directories.

repeat *count command*
The specified *command* (which is subject to the same restrictions as the *command* in the one-line **if** statement above) is executed *count* times. I/O redirections occur exactly once, even if *count* is 0.

set
set *name*
set *name***=***word*
set *name***[** *index* **]=***word*
set *name***=(** *wordlist* **)**
The first form of **set** shows the value of all shell variables. Variables whose value is other than a single word print as a parenthesized word list. The second form sets *name* to the null string. The third form sets *name* to the single *word*. The fourth form sets the *index*th component of *name* to *word*; this component must already exist. The final form sets *name* to the list of words in *wordlist*. In all cases the value is command and file-name expanded.

These arguments can be repeated to set multiple values in a single *set* command. Note, however, that variable expansion happens for all arguments before any setting occurs.

setenv *name value*
Sets the value of environment variable *name* to be *value*, a single string. The most commonly used environment variables, **USER**, **TERM**, and **PATH**, are automatically imported to and exported from the **csh** variables *user*, *term*, and *path*; there is no need to use **setenv** for these.

shift [*variable*]
If no argument is given, the members of **argv** are shifted to the left, discarding **argv[1]**. An error occurs if **argv** is not set or has less than two strings assigned to it. When *variable* is specified, *shift* performs the same function on the specified *variable*.

source [**-h**] *name*
csh reads commands from *name*. **source** commands can be nested, but if nested too deeply the shell may run out of file descriptors. An error in a **source** at any level terminates all nested **source** commands. Normally, input during **source** commands is not placed on the history list. The **-h** option can be used to place commands in the history list without being executing them.

stop [**%***job* ...]
Stops the current (no argument) or specified jobs executing in the background.

suspend
Causes **csh** to stop as if it had been sent a *suspend* signal. Since **csh** normally ignores *suspend* signals, this is the only way to suspend the shell. This command gives an error message if attempted from a login shell.

switch (*string***)**
case *str1* **:**
  . . .
breaksw
  . . .

C

default:
. . .
breaksw
endsw Each **case** label (*str1*) is successively matched against the specified *string* which is first command and file name expanded. The form of the **case** labels is the Pattern Matching Notation with the exception that non-matching lists in bracket expressions are not supported (see *regexp*(5)). If none of the labels match before a **default** label is found, the execution begins after the **default** label. Each **case** label and the **default** label must appear at the beginning of a line. The **breaksw** command causes execution to continue after the *endsw*. Otherwise, control may fall through **case** labels and **default** labels as in C. If no label matches and there is no default, execution continues after the **endsw**.

time [*command*]
When *command* is not specified, a summary of time used by this shell and its children is printed. If specified, the simple *command* is timed and a time summary as described under the **time** variable is printed. If necessary, an extra shell is created to print the time statistic when the command completes.

umask [*value*]
The current file creation mask is displayed (*value* not specified) or set to the specified *value*. The mask is given in octal. Common values for the mask are **002**, which gives all permissions to the owner and group and read and execute permissions to all others, or **022**, which gives all permissions to the owner, and only read and execute permission to the group and all others.

unalias *pattern*
All aliases whose names match the specified *pattern* are discarded. Thus, all aliases are removed by **unalias ***. No error occurs if *pattern* does not match an existing alias.

unhash
Use of the internal hash table to speed location of executed programs is disabled.

unset *pattern*
All variables whose names match the specified *pattern* are removed. Thus, all variables are removed by **unset ***; this has noticeably undesirable side-effects. No error occurs if *pattern* matches nothing.

unsetenv *pattern*
Removes all variables whose names match the specified *pattern* from the environment. See also the **setenv** command above and *printenv*(1).

wait Waits for all background jobs to terminate. If the shell is interactive, an interrupt can disrupt the wait, at which time the shell prints names and job numbers of all jobs known to be outstanding.

while (*expression***)**
. . .
end While the specified *expression* evaluates non-zero, the commands between the **while** and the matching **end** are evaluated. **break** and **continue** can be used to terminate or continue the loop prematurely. (The **while** and **end** must appear alone on their input lines.) If the input is a terminal (i.e., not a script), prompting occurs the first time through the loop as for the **foreach** statement.

%*job* Brings the specified job into the foreground.

%*job* **&** Continues the specified job in the background.

@
@ *name***=***expression*
@ *name***[***index***]=***expression*
The first form prints the values of all the shell variables. The second form sets the specified *name* to the value of *expression*. If the expression contains **<**, **>**, **&**, or **|**, at least this part of the expression must be placed within parentheses. The third form assigns the value of *expression* to the *index*th argument of *name*. Both *name* and its *index*th component must already exist.

The operators ***=**, **+=**, etc., are available as in C. White space can optionally separate the *name* from the assignment operator. However, spaces are mandatory in separating components of *expression* which would otherwise be single words.

Special postfix **++** and **--** operators increment and decrement *name*, respectively (e.g., **@ i++**).

Non-Built-In Command Execution

When a command to be executed is not a built-in command, **csh** attempts to execute the command via *exec*(2). Each word in the variable *path* names a directory in which the shell attempts to find the command (if the command does not begin with **/**). If neither **-c** nor **-t** is given, the shell hashes the names in these directories into an internal table so that an *exec* is attempted only in those directories where the command might possibly reside. This greatly speeds command location when a large number of directories are present in the search path. If this mechanism has been turned off (via **unhash**), or if **-c** or **-t** was given, or if any directory component of *path* does not begin with a **/**, the shell concatenates the directory name and the given command name to form a path name of a file which it then attempts to execute.

Commands placed inside parentheses are always executed in a subshell. Thus

```
(cd ; pwd)
```

prints the *home* directory then returns to the current directory upon completion, whereas:

```
cd ; pwd
```

remains in the *home* directory upon completion.

When commands are placed inside parentheses, it is usually to prevent *chdir* from affecting the current shell.

If the file has execute permissions but is not an executable binary file, it is assumed to be a script file, which is a file of data for an interpreter that is executed as a separate process.

csh first attempts to load and execute the script file (see *exec*(2)). If the first two characters of the script file are **#!**, *exec*(2) expects an interpreter path name to follow and attempts to execute the specified interpreter as a separate process to read the entire script file.

If no **#! interpreter** is named, and there is an *alias* for the shell, the words of the *alias* are inserted at the beginning of the argument list to form the shell command. The first word of the *alias* should be the full path name of the command to be used. Note that this is a special, late-occurring case of *alias* substitution, which inserts words into the argument list without modification.

If no **#! interpreter** is named and there is no shell *alias*, but the first character of the file is **#**, the interpreter named by the **$shell** variable is executed (note that this normally would be **/usr/bin/csh**, unless the user has reset **$shell**). If **$shell** is not set, **/usr/bin/csh** is executed.

If no **!# interpreter** is named, and there is no shell alias, and the first character of the file is not **#**, **/usr/bin/sh** is executed to interpret the script file.

History Substitutions

History substitutions enable you to repeat commands, use words from previous commands as portions of new commands, repeat arguments of a previous command in the current command, and fix spelling or typing mistakes in an earlier command.

History substitutions begin with an exclamation point (**!**). Substitutions can begin anywhere in the input stream, but *cannot* be nested. The exclamation point can be preceded by a backslash to cancel its special meaning. For convenience, an exclamation point is passed to the parser unchanged when it is followed by a blank, tab, newline, equal sign, or left parenthesis. Any input line that contains history substitution is echoed on the terminal before it is executed for verification.

Commands input from the terminal that consist of one or more words are saved on the history list. The history substitutions reintroduce sequences of words from these saved commands into the input stream. The number of previous commands saved is controlled by the **history** variable. The previous command is always saved, regardless of its value. Commands are numbered sequentially from 1.

You can refer to previous events by event number (such as **!10** for event 10), relative event location (such as **!-2** for the second previous event), full or partial command name (such as **!d** for the last event using a command with initial character **d**), and string expression (such as **!?mic?** referring to an event containing the characters **mic**).

These forms, without further modification, simply reintroduce the words of the specified events, each separated by a single blank. As a special case, **!!** is a re-do; it refers to the previous command.

To select words from a command, use a colon (**:**) and a designator for the desired words after the event specification. The words of an input line are numbered from zero. The basic word designators are:

0	First word (i.e., the command name itself).
n	*n*th word.
^	First argument. (This is equivalent to **1**.)
$	Last word.
a-*b*	Range of words from *a* through *b*. Special cases are **-***y*, an abbreviation for "word 0 through word *y*"; and *x***-**, which means "word *x* up to, but not including, word **$**".
*****	Range from the second word through the last word.
%	Used with a search sequence to substitute the immediately preceding matching word.

The colon separating the command specification from the word designator can be omitted if the argument selector begins with a **^**, **$**, *****, **-**, or **%**.

After word designator can be followed by a sequence of modifiers, each preceded by a colon. The following modifiers are defined:

h	Use only the first component of a path name by removing all following components.
r	Use the root file name by removing any trailing suffix (.xxx).
e	Use the file name's trailing suffix (**.***xxx*) by removing the root name.
s /*l***/***r*	substitute the value of *r* for the value *l* in the indicated command.
t	Use only the final file name of a path name by removing all leading path name components.
&	Repeat the previous substitution.
p	Print the new command but do not execute it.
q	Quote the substituted words, preventing further substitutions.
x	Like **q**, but break into words at blanks, tabs and newlines.
g	Use a **g**lobal command as a prefix to another modifier to cause the specified change to be made globally. All words in the command are changed, one change per word, and each string enclosed in single quotes (**'**) or double quotes (**"**) is treated as a single word.

Unless preceded by a **g**, the modification is applied only to the first modifiable word. An error results if a substitution is attempted and cannot be completed (i.e., if you ask for a substitution of **!11** on a history buffer containing only 10 commands).

The left hand side of substitutions are strings; not regular expressions in the sense of HP-UX editors. Any character can be used as the delimiter in place of a slash (/). Use a backslash to quote a delimiter character if it is used in the *l* or *r* string. The character **&** in the right-hand side is replaced by the text from the left. A \ also quotes **&**. A null *l* string uses the previous string either from an *l* or from a contextual scan string *s* in **!?***s***?**. The trailing delimiter in the substitution can be omitted if a new-line character follows immediately, as may the trailing **?** in a contextual scan.

A history reference can be given without an event specification (as in **!$**). In this case, the reference is to the previous command unless a previous history reference occurred on the same line, in which case this form repeats the previous reference. Thus

```
!?foo?^ !$
```

gives the first and last arguments from the command matching **?foo?**.

A special abbreviation of a history reference occurs when the first non-blank character of an input line is a circumflex (^). This is equivalent to **!:s^**, providing a convenient shorthand for substitutions on the text of the previous line. Thus **^lb^lib** fixes the spelling of **lib** in the previous command.

Finally, a history substitution can be enclosed within curly braces **{ }** if necessary to insulate it from the characters which follow. Thus, after

```
ls -ld ~paul
```

one could execute **!{l}a** to do

```
ls -ld ~paula
```

while **!la** would look for a command starting with **la**.

Quoting with Single and Double Quotes

The quotation of strings by single quotes (') and double quotes (**"**) can be used to prevent all or some of the remaining substitutions. Strings enclosed in single quotes are protected from any further interpretation. Strings enclosed in double quotes are still variable- and command-expanded as described below.

In both cases the resulting text becomes (all or part of) a single word. Only in one special case (see *Command Substitution* below) does a double-quoted string yield parts of more than one word; single-quoted strings never do.

Alias Substitution

csh maintains a list of aliases that can be established, displayed, and modified by the **alias** and **unalias** commands. After a command line is scanned, it is parsed into distinct commands and the first word of each command, left-to-right, is checked to see if it has an alias. If it does, the text which is the alias for that command is reread with the history mechanism available as if that command was the previous input line. The resulting words replace the command and argument list. If no reference is made to the history list, the argument list is left unchanged.

Thus, if the alias for **ls** is **ls -l**, the command **ls /usr** maps to **ls -l /usr**, leaving the argument list undisturbed. Similarly, if the alias for **lookup** was **grep !^ /etc/passwd**, **lookup bill** maps to **grep bill /etc/passwd** .

If an alias is found, the word transformation of the input text is performed and the aliasing process begins again on the re-formed input line. Looping is prevented if the first word of the new text is the same as the old by flagging it to prevent further aliasing. Other loops are detected and cause an error.

Note that the mechanism allows aliases to introduce parser metasyntax. Thus:

```
alias print 'pr \!* | lp'
```

makes a command that uses *pr*(1) to print its arguments on the line printer.

Expressions

Some of the built-in commands take expressions in which the operators are similar to those of C, with the same precedence. These expressions appear in the **@**, **exit**, **if**, and **while** commands. The following operators are available (shown in order of increasing precedence):

```
|| && | ^ & == != =~ !~ <= >= < > << >> + - * / % ! ~ ( )
```

The following list shows the grouping of these operators. The precedence decreases from top to bottom in the list:

```
* / %
+ -
<< >>
<= >= < >
== != =~ !~
```

The operators **==**, **!=**, **=~**, and **!~** compare their arguments as strings; all others operate on numbers. The operators **=~** and **!~** are similar to **!=** and **==**, except that the right-hand side is a *pattern* (containing *****s, **?**s, and instances of **[**...**]**) against which the left hand operand is matched. This reduces the need for use of the **switch** statement in shell scripts when all that is really needed is pattern matching.

Strings beginning with **0** are considered octal numbers. Null or missing arguments are considered **0**. The result of all expressions are strings that represent decimal numbers. It is important to note that no two components of an expression can appear in the same word. These components should be surrounded by spaces except when adjacent to components of expressions that are syntactically significant to the parser: **-**, **&**, **|**, **<**, **>**, **(**, and **)**.

Also available in expressions as primitive operands are command executions enclosed in curly braces (**{ }**) and file enquiries of the form **-***l filename*, where *l* is one of:

r	read access
w	write access
x	execute access
e	existence
o	ownership
z	zero size
f	plain file
d	directory

The specified *filename* is command- and file-name expanded then tested to see if it has the specified relationship to the real user. If the file does not exist or is inaccessible, all inquiries return false (0). Command executions succeed, returning true, if the command exits with status 0; otherwise they fail, returning false. If more detailed status information is required, the command should be executed outside of an expression and the **status** variable examined.

Control of the Flow

csh contains a number of commands that can be used to regulate the flow of control in command files (shell scripts) and (in limited but useful ways) from terminal input. These commands all operate by forcing the shell to reread or skip parts of its input and, due to the implementation, restrict the placement of some of the commands.

The **foreach**, **switch**, and **while** statements, as well as the **if-then-else** form of the **if** statement require that the major keywords appear in a single simple command on an input line as shown below.

If the shell's input is not seekable, the shell buffers input whenever a loop is being read and performs seeks in this internal buffer to accomplish the rereading implied by the loop. (To the extent that this allows, backward **goto**s succeed on non-seekable inputs.)

Signal Handling

csh normally ignores *quit* signals. Jobs running in background mode are immune to signals generated from the keyboard, including hangups. Other signals have the values which the shell inherited from its parent. **csh**'s handling of interrupts and terminate signals in shell scripts can be controlled by *onintr*. Login shells catch the *terminate* signal; otherwise this signal is passed on to children from the state in the shell's parent. In no case are interrupts allowed when a login shell is reading the file **.logout**.

Command Line Parsing

csh splits input lines into words at blanks and tabs. The following exceptions (parser metacharacters) are considered separate words:

&	ampersand;
|	vertical bar;
;	semicolon;
<	less-than sign;
>	greater-than sign;
(	left parenthesis;
)	right parenthesis;
&&	double ampersand;
||	double vertical bar;
<<	double less-than sign;
>>	double greater-than sign;
#	comment delimiter

The backslash (****) removes the special meaning of these parser metacharacters. A parser metacharacter preceded by a backslash is interpreted as its ASCII value. A newline character (ASCII 10) preceded by a backslash is equivalent to a blank.

Strings enclosed in single or double quotes form parts of a word. Metacharacters in these strings, including blanks and tabs, do not form separate words. Within pairs of backslashes or quotes, a newline preceded by a backslash gives a true newline character.

When **csh**'s input is not a terminal, the **#** character introduces a comment terminated by a newline.

CSH VARIABLES

csh maintains a set of variables. Each variable has a value equal to zero or more strings (words). Variables have names consisting of up to 80 letters and digits starting with a letter. The underscore character is considered a letter. The value of a variable may be displayed and changed by using the **set** and

unset commands. Some of the variables are Boolean, that is, the shell does not care what their value is, only whether they are set or not.

Some operations treat variables numerically. The at sign (**@**) command permits numeric calculations to be performed and the result assigned to a variable. The null string is considered to be zero, and any subsequent words of multi-word values are ignored.

After the input line is aliased and parsed, and before each command is executed, variable expansion is performed keyed by the dollar sign (**$**)**character.** Variable expansion can be prevented by preceding the dollar sign with a backslash character (****) except within double quotes (**"**) where substitution **always** occurs. Variables are never expanded if enclosed in single quotes. Strings quoted by single quotes are interpreted later (see *Command Substitution*) so variable substitution does not occur there until later, if at all. A dollar sign is passed unchanged if followed by a blank, tab, or end-of-line.

Input/output redirections are recognized before variable expansion, and are variable expanded separately. Otherwise, the command name and entire argument list are expanded together.

Unless enclosed in double quotes or given the **:q** modifier, the results of variable substitution may eventually be command and file name substituted. Within double quotes, a variable whose value consists of multiple words expands to a portion of a single word, with the words of the variable's value separated by blanks. When the **:q** modifier is applied to a substitution, the variable expands to multiple words with each word separated by a blank and quoted to prevent later command or file name substitution.

The following metasequences are provided for introducing variable values into the shell input. Except as noted, it is an error to reference a variable that is not set.

$*variable_name*
${*variable_name***}**
When interpreted, this sequence is replaced by the words of the value of the variable *variable_name*, each separated by a blank. Braces insulate *variable_name* from subsequent characters that would otherwise be interpreted to be part of the variable name itself.
If *variable_name* is not a **csh** variable, but is set in the environment, that value is used. Non-**csh** variables cannot be modified as shown below.

$*variable_name*[*selector*]
${*variable_name*[*selector*]**}**
This modification selects only some of the words from the value of *variable_name*. The selector is subjected to variable substitution, and can consist of a single number or two numbers separated by a dash. The first word of a variable's value is numbered **1**. If the first number of a range is omitted it defaults to **1**. If the last member of a range is omitted it defaults to the total number of words in the variable (**$#***variable_name*). An asterisk metacharacter used as a selector selects all words.

$#*variable_name*
${#*variable_name***}**
This form gives the number of words in the variable, and is useful for forms using a [*selector*] option.

$0 This form substitutes the name of the file from which command input is being read. An error occurs if the file name is not known.

$*number*
${*number***}**
This form is equivalent to an indexed selection from the variable **argv** (**$argv[** *number***]**).

$* This is equivalent to selecting all of *argv* (**$argv[*]**).

The modifiers **:h**, **:t**, **:r**, **:q**, and **:x** can be applied to the substitutions above, as can CR :gh , CR :gt , and CR :gr . If curly braces (**{ }**) appear in the command form, the modifiers must appear within the braces. *The current implementation allows only one* **:** *modifier on each* **$d** *expansion*.

The following substitutions cannot be modified with **:** modifiers:

$?*variable_name*
${?*variable_name***}**
Substitutes the string **1** if *variable_name* is set, **0** if it is not.

$?0 Substitutes **1** if the current input file name is known, **0** if it is not.

$$ Substitutes the (decimal) process number of the (parent) shell.

$< Substitutes a line from the standard input, with no further interpretation thereafter. It can be used to read from the keyboard in a shell script.

Pre-Defined and Environment Variables

The following variables have special meaning to the shell. Of these **autologout**, **argv**, **cwd**, **home**, **path**, **prompt**, **shell**, and **status** are always set by the shell. Except for **cwd** and **status**, this setting occurs only at initialization (initial execution of **csh**). These variables are not modified unless modified explicitly by the user.

csh copies the HP-UX environment variable USER into the shell variable **user**, the environment variable **TERM** into **term**, the environment variable **HOME** into **home**, and **PATH** into **path**. **csh** copies these values back into the environment whenever the **csh** variables are reset.

In a windowed environment, if **csh** detects that the window has changed size, **csh** sets the environment variables **LINES** and **COLUMNS** to match the new window size.

argv This variable is set to the arguments of the **csh** command statement. It is from this variable that positional parameters are substituted; i.e., **$1** is replaced by **$argv[1]**, etc.

cdpath This variable gives a list of alternate directories searched to find subdirectories in *chdir* commands.

cwd This variable contains the absolute path name of the current working directory. Whenever changing directories (using *cd*), this variable is updated.

echo This variable is set by the **-x** command line option. If set, all built-in commands and their arguments are echoed to the standard output device just before being executed. Built-in commands are echoed before command and file name substitution, since these substitutions are then done selectively. For non-built-in commands, all expansions occur before echoing.

history This variable is used to create the command history buffer and to set its size. If this variable is not set, no command history is maintained and history substitutions cannot be made. Very large values of **history** can cause shell memory overflow. Values of 10 or 20 are normal. All commands, executable or not, are saved in the command history buffer.

home This variable contains the absolute path name to your home directory. The variable **home** is initialized from the HP-UX environment. File name expansion of tilde (**~**) refers to this variable.

ignoreeof If set, **csh** ignores end-of-file characters from input devices that are terminals. **csh** exits normally when it encounters the end-of-file condition (CTRL-**D** typed as the first character on a command line). Setting *ignoreeof* prevents the current shell from being killed by an accidental (CTRL-**D**. However, to prevent an infinite loop of EOF input, **csh** terminates if it receives 26 consecutive EOFs.

mail This variable contains a list of the files where **csh** checks for your mail. **csh** periodically (default is 10 minutes) checks this variable before producing a prompt upon command completion. If the variable contains a file name that has been modified since the last check (resulting from mail being put in the file), **csh** prints **You have new mail**.

If the first word of the value of **mail** is numeric, that number specifies a different mail checking interval in seconds.

If multiple mail files are specified, the shell says **New mail in** *file_name*, where *file_name* is the file containing the mail.

noclobber This variable places restrictions on output redirection to ensure that files are not accidentally destroyed, and that commands using append redirection (**>>**) refer to existing files.

noglob If set, file name expansion is inhibited. This is most useful in shell scripts that are not dealing with file names, or after a list of file names has been obtained and further expansions are not desirable.

nonomatch If set, it is no longer an error for a file name expansion to not match any existing files. If there is no match, the primitive pattern is returned. It is still an error for the primitive pattern to be malformed. For example, **'echo ['** still gives an error.

notify If set, **csh** notifies you immediately (through your standard output device) of background job completions. The default is **unset** (indicate job completions just before printing a prompt).

path Each word of the path variable specifies a directory in which commands are to be sought for execution. A null word specifies your current working directory. If there is no *path* variable, only full path names can be executed. When *path* is not set and when users do not specify full path names, **csh** searches for the command through the directories **.** (current directory) and **/usr/bin**. A **csh** which is given neither the **-c** nor the **-t** option normally hashes the contents of the directories in the **path** variable after reading **.cshrc**, and each time the **path** variable is reset. If new commands are added to these directories while the shell is active, it is necessary to execute **rehash** for **csh** to access these new commands.

prompt This variable lets you select your own prompt character string. The prompt is printed before each command is read from an interactive terminal input. If a **!** appears in the string, it is replaced by the current command history buffer event number unless a preceding **** is given. The default prompt is the percent sign (**%**) for users and the **#** character for the super-user.

savehist The number of lines from the history list that are saved in **~/.history** when the user logs out. Large values for **savehist** slow down the **csh** during startup.

shell This variable contains the name of the file in which the **csh** program resides. This variable is used in forking shells to interpret files that have their execute bits set but which are not executable by the system. (See the description of *Non-Built-In Command Execution*).

status This variable contains the status value returned by the last command. If the command terminated abnormally, 0200 is added to the status variable's value. Built-in commands which terminated abnormally return exit status **1**, and all other built-in commands set status to **0**.

time This variable contains a numeric value that controls the automatic timing of commands. If set, **csh** prints, for any command taking more than the specified number of cpu seconds, a line of information to the standard output device giving user, system, and real execution times plus a utilization percentage. The utilization percentage is the ratio of user plus system times to real time. This message is printed after the command finishes execution.

verbose This variable is set by the **-v** command line option. If set, the words of each command are printed on the standard output device after history substitutions have been made.

Command and File name Substitution

The remaining substitutions, command and file name substitution, are applied selectively to the arguments of built-in commands. This means that portions of expressions that are not evaluated are not subjected to these expansions. For commands which are not internal to the shell, the command name is substituted separately from the argument list. This occurs very late, after input-output redirection is performed, and in a child of the main shell.

Command Substitution

Command substitution is indicated by a command enclosed in grave accents (\` ...\`). The output from such a command is normally broken into separate words at blanks, tabs and newlines, with null words being discarded; this text then replacing the original string. Within double quotes, only newlines force new words; blanks and tabs are preserved.

In any case, the single final newline does not force a new word. Note that it is thus possible for a command substitution to yield only part of a word, even if the command outputs a complete line.

File name Substitution

Each command *word* is processed as a pattern for file name substitution, also known as **globbing**, and replaced with a sorted list of file names which match the pattern. The form of the patterns is the Pattern Matching Notation defined by *regexp*(5) with the following exceptions:

- Non-matching lists in bracket expressions are not supported.
- In a list of words specifying file name substitution it is an error for no pattern to match an existing file name, but it is not required for each pattern to match.
- The metanotation **a{b,c,d}e** is a shorthand for "abe ace ade". Left to right order is preserved, with results of matches being sorted separately at a low level to preserve this order. This construct may be nested. Thus:

  ```
  ~source/s1/{oldls,ls}.c
  ```

  expands to

  ```
  /home/source/s1/oldls.c /home/source/s1/ls.c
  ```

  whether or not these files exist, without any chance of error if the home directory for **source** is **/home/source**. Similarly,

  ```
  ../{memo,*box}
  ```

  might expand to

  ```
  ../memo ../box ../mbox
  ```

  (Note that **memo** was not sorted with the results of matching ***box**.) As a special case, **{**, **}**, and **{ }** are passed undisturbed.

Input/Output

The standard input and standard output of a command can be redirected with the following syntax:

< *name* — Open file *name* (which is first variable, command and file name expanded) as the standard input.

<< *word* — Read the shell input up to a line which is identical to *word*. *word* is not subjected to variable, file name or command substitution, and each input line is compared to *word* before any substitutions are done on this input line. Unless a quoting \, **'**, or ` appears in *word*, variable and command substitution is performed on the intervening lines, allowing **\\** to quote **$**, **\\** and `. Commands which are substituted have all blanks, tabs, and newlines preserved, except for the final newline which is dropped. The resultant text is placed in an anonymous temporary file which is given to the command as standard input.

> *name*
>! *name*
>& *name*
>&! *name* — The file *name* is used as standard output. If the file does not exist, it is created; if the file exists, it is truncated, and its previous contents are lost.

If the variable **noclobber** is set, the file must not exist or be a character special file (e.g., a terminal or **/dev/null**) or an error results. This helps prevent accidental destruction of files. In this case the exclamation point (**!**) forms can be used to suppress this check.

The forms involving the ampersand character (**&**) route the standard error into the specified file as well as the standard output. *name* is expanded in the same way as < input file names are.

>> *name*
>>& *name*
>>! *name*
>>&! *name* — Uses file *name* as standard output the same as **>**, but appends output to the end of the file. If the variable **noclobber** is set, it is an error for the file not to exist unless one of the **!** forms is given. Otherwise, it is similar to **>**.

A command receives the environment in which the shell was invoked as modified by the input-output parameters and the presence of the command in a pipeline. Thus, unlike some previous shells, commands executed from a shell script have no access to the text of the commands by default; rather they receive the original standard input of the shell. The **<<** mechanism should be used to present inline data. This permits shell scripts to function as components of pipelines and allows the shell to block-read its input.

Diagnostic output can be directed through a pipe with the standard output. Simply use the form **|&** rather than **|** by itself.

CSH UTILITIES

File Name Completion

In typing file names as arguments to commands, it is no longer necessary to type a complete name, only a unique abbreviation is necessary. When you want the system to try to match your abbreviation, press the ESC key. The system then completes the file name for you, echoing the full name on your terminal. If the abbreviation does not match an available file name, the terminal's bell is sounded. The file name may be partially completed if the prefix matches several longer file names. In this case, the name is extended up to the ambiguous deviation, and the bell is sounded.

File name completion works equally well when other directories are addressed. In addition, the tilde (~) convention for home directories is understood in this context.

Viewing a File or Directory List

At any point in typing a command, you can request "what files are available" or "what files match my current specification". Thus, when you have typed:

```
% cd ~speech/data/bench/fritz/
```

you may wish to know what files or subdirectories exist (in **~speech/data/bench/fritz**), without aborting the command you are typing. Typing CTRL-**D** at this point lists the files available. Files are listed in multicolumn format, sorted by column. Directories and executable files are identified by a trailing **/** and *****, respectively. Once printed, the command is re-echoed for you to complete. Additionally, you may want to know which files match a prefix, the current file specification so far. If you had typed:

```
% cd ~speech/data/bench/fr
```

followed by a CTRL-**D**, all files and subdirectories whose prefix was **fr** in the directory **~speech/data/bench** would be printed. Notice that the example before was simply a degenerate case of this with a null trailing file name. (The null string is a prefix of all strings.) Notice also that a trailing slash is required to pass to a new sub-directory for both file name completion and listing. Note that the degenerate case

```
% ~^D
```

prints a full list of login names on the current system.

Command Name Recognition

Command name recognition and completion works in the same manner as file name recognition and completion above. The current value of the environment variable **PATH** is used in searching for the command. For example

```
% newa [Escape]
```

might expand to

```
% newaliases
```

Also,

```
% new [Control]-[D]
```

lists all commands (along **PATH**) that begin with **new**. As an option, if the shell variable **listpathnum** is set, a number indicating the index in **PATH** is printed next to each command on a [Control]-[D] listing.

Autologout

A new shell variable has been added called **autologout**. If the terminal remains idle (no character input) at the shell's top level for a number of minutes greater than the value assigned to **autologout**, you are automatically logged off. The **autologout** feature is temporarily disabled while a command is executing. The initial value of **autologout** is 60. If unset or set to 0, **autologout** is entirely disabled.

Command Line Control

A **^R** re-prints the current command line; **^W** erases the last word entered on the current command line.

Sanity

C shell restores your terminal to a sane mode if it appears to return from some command in raw, cbreak, or noecho mode.

Saving Your History Buffer

csh has the ability to save your history list between login sessions. If the shell variable **savehist** is set to a number, that number of command events from your history list is saved. For example, placing the line

```
set history=10 savehist=10
```

in your **.cshrc** file maintains a history buffer of length 10 and saves the entire list when you logout. When you log back in, the entire buffer is restored. The commands are saved in the file **.history** in your login directory.

EXTERNAL INFLUENCES

Environment Variables

LC_COLLATE determines the collating sequence used in evaluating pattern matching notation for file name substitution.

LC_CTYPE determines the interpretation of text as single and/or multi-byte characters, the classification of characters as letters, and the characters matched by character class expressions in pattern matching notation.

LANG determines the language in which messages are displayed.

If LC_COLLATE or LC_CTYPE is not specified in the environment or is set to the empty string, the value of LANG is used as a default for each unspecified or empty variable. If LANG is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of LANG. If any internationalization variable contains an invalid setting, **csh** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported.

WARNINGS

The **.cshrc** file should be structured such that it cannot generate any output on standard output or standard error, including occasions when it is invoked without an affiliated terminal. *rcp*(1) causes **.cshrc** to be sourced, and any output generated by this file, even to standard error causes problems. Commands such as *stty*(1) should be placed in **.login**, not in **.cshrc**, so that their output cannot affect *rcp*(1).

csh has certain limitations. Words or environment variables can be no longer than 10240 characters. The system limits argument lists to 10240 characters. The number of arguments to a command which involves file name expansion is limited to one-sixth the number of characters allowed in an argument list. Command substitutions may substitute no more characters than are allowed in an argument list.

To detect looping, the shell restricts the number of **alias** substitutions on a single line to 20.

When a command is restarted from a stop, **csh** prints the directory it started in if it is different from the current directory; this can be misleading (i.e., wrong) because the job may have changed directories internally.

Shell built-in functions are not stoppable/restartable. Command sequences of the form **a ; b ; c** are also not handled gracefully when stopping is attempted. If you interrupt **b**, the shell then immediately executes **c**. This is especially noticeable if this expansion results from an **alias**. It suffices to place the sequence of commands in parentheses to force it into a subshell; i.e., **(a ; b ; c)**.

Because of the signal handling required by csh, interrupts are disabled just before a command is executed, and restored as the command begins execution. There may be a few seconds delay between when a command is given and when interrupts are recognized.

Control over tty output after processes are started is primitive; perhaps this will inspire someone to work on a good virtual terminal interface. In a virtual terminal interface much more interesting things could be done with output control.

Alias substitution is most often used to clumsily simulate shell procedures; shell procedures should be provided rather than aliases.

Commands within loops, prompted for by **?**, are not placed in the *history* list. Control structure should be parsed rather than being recognized as built-in commands. This would allow control commands to be placed anywhere, to be combined with **|**, and to be used with **&** and **;** metasyntax.

It should be possible to use the **:** modifiers on the output of command substitutions. All and more than one **:** modifier should be allowed on **$** substitutions.

Terminal type is examined only the first time you attempt recognition.

To list all commands on the system along **PATH**, enter [Space]-[Ctrl]-[D].

The csh metasequence **!~** does not work.

In an international environment, character ordering is determined by the setting of LC_COLLATE, rather than by the binary ordering of character values in the machine collating sequence. This brings with it certain attendant dangers, particularly when using range expressions in file name generation patterns. For example, the command,

```
rm [a-z]*
```

might be expected to match all file names beginning with a lowercase alphabetic character. However, if dictionary ordering is specified by LC_COLLATE, it would also match file names beginning with an uppercase character (as well as those beginning with accented letters). Conversely, it would fail to match letters collated after **z** in languages such as Norwegian.

The correct (and safe) way to match specific character classes in an international environment is to use a pattern of the form:

```
rm [[:lower:]]*
```

This uses **LC_CTYPE** to determine character classes and works predictably for all supported languages and codesets. For shell scripts produced on non-internationalized systems (or without consideration for the above dangers), it is recommended that they be executed in a non-NLS environment. This requires that **LANG**, **LC_COLLATE**, etc., be set to "C" or not set at all.

csh implements command substitution by creating a pipe between itself and the command. If the root file system is full, the substituted command cannot write to the pipe. As a result, the shell receives no input from the command, and the result of the substitution is null. In particular, using command substitution for variable assignment under such circumstances results in the variable being silently assigned a NULL value.

Relative path changes (such as **cd ..**), when in a symbolically linked directory, cause **csh**'s knowledge of the working directory to be along the symbolic path instead of the physical path.

Prior to HP-UX Release 9.0, **csh**, when getting its input from a file, would exit immediately if unable to execute a command (such as if it was unable to find the command). Beginning at Release 9.0, **csh** continues on and attempts to execute the remaining commands in the file. However, if the old behavior is desired for compatibility purposes, set the environment variable **EXITONERR** to 1.

AUTHOR

csh was developed by the University of California, Berkeley and HP.

FILES

~/.cshrc	A csh script sourced (executed) at the beginning of execution by each shell. See WARNINGS
~/.login	A csh script sourced (executed) by login shell, after **.cshrc** at login.
~/.logout	A csh script sourced (executed) by login shell, at logout.
/etc/passwd	Source of home directories for **~name**.
/usr/bin/sh	Standard shell, for shell scripts not starting with a **#**.
/etc/csh.login	A csh script sourced (executed) before **~/.cshrc** and **~/.login** when starting a csh login (analogous to **/etc/profile** in the Bourne shell).
/tmp/sh*	Temporary file for **<<**.

SEE ALSO

sh(1), access(2), exec(2), fork(2), pipe(2), umask(2), wait(2), tty(7), a.out(4), environ(5), lang(5), regexp(5).

C Shell tutorial in *Shells Users Guide*.

NAME

csplit - context split

SYNOPSIS

csplit [**-s**] [**-k**] [**-f** *prefix*] [**-n** *number*] *file arg1* [*... argn*]

DESCRIPTION

csplit reads *file*, separates it into *n*+1 sections as defined by the arguments *arg1* ... *argn*, and places the results in separate files. The maximum number of arguments (*arg1* through *argn*) allowed is 99 unless the **-n** *number* option is used to allow for more output file names. If the **-f** *prefix* option is specified, the resulting filenames are *prefix***00** through *prefixNN* where *NN* is the two-digit value of *n* using a leading zero if *n* is less than 10. If the **-f** *prefix* option is not specified, the default filenames **xx00** through **xx***NN* are used. *file* is divided as follows:

Default Filename	Prefixed Filename	Contents
xx00	*prefix***00**	From start of *file* up to (but not including) the line referenced by *arg1*.
xx01	*prefix***01**	From the line referenced by *arg1* up to the line referenced by *arg2*.
		.
		.
		.
xx*NN*	*prefixNN*	From the line referenced by *argn* to end of *file*.

If the *file* argument is **-**, standard input is used.

csplit supports the Basic Regular Expression syntax (see *regexp*(5)).

Options

csplit recognizes the following options:

-s Suppress printing of all character counts (**csplit** normally prints the character counts for each file created).

-k Leave previously created files intact (**csplit** normally removes created files if an error occurs).

-f *prefix* Name created files *prefix***00** through *prefixNN* (default is **xx00** through **xx***NN*.

-n *number* The output file name suffix will use *number* digits instead of the default **2.** This allows creation of more than 100 output files.

Arguments (*arg1* through *argn*) to **csplit** can be any combination of the following:

/*regexp***/** Create a file containing the section from the current line up to (but not including) the line matching the regular expression *regexp*. The new current line becomes the line matching *regexp*.

/*regexp***/+***n*
/*regexp***/-***n* Create a file containing the section from the current line up to (but not including) the *n*th before (**-***n*) or after (**+***n*) the line matching the regular expression *regexp*. (e.g., **/Page/-5**). The new current line becomes the line matching *regexp*±*n* lines.

%*regexp***%** equivalent to **/***regexp***/,** except that no file is created for the section.

line_number Create a file from the current line up to (but not including) *line_number*. The new current line becomes *line_number*.

{*num***}** Repeat argument. This argument can follow any of the above argument forms. If it follows a *regexp* argument, that argument is applied *num* more times. If it follows *line_number*, the file is split every *line_number* lines for *num* times from that point until end-of-file is reached or *num* expires.

{*****}** Repeats previous operand as many times as necessary to finish input.

Enclose in appropriate quotes all *regexp* arguments containing blanks or other characters meaningful to the shell. Regular expressions must not contain embedded new-lines. **csplit** does not alter or remove the original file; it is the user's responsibility to remove it when appropriate.

EXTERNAL INFLUENCES

Environment Variables

LC_COLLATE determines the collating sequence used in evaluating regular expressions.

LC_CTYPE determines the characters matched by character class expressions in regular expressions.

LC_MESSAGES determines the language in which messages are displayed.

If **LC_COLLATE** or **LC_CTYPE** or **LC_MESSAGES** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG.** If any internationalization variable contains an invalid setting, **csplit** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported.

DIAGNOSTICS

Messages are self explanatory except for:

```
arg - out of range
```

which means that the given argument did not reference a line between the current position and the end of the file. This warning also occurs if the file is exhausted before the repeat count is.

EXAMPLES

Create four files, **cobol00** through **cobol03**. After editing the “split” files, recombine them back into the original file, destroying its previous contents.

```
csplit -f cobol file '/procedure division/' /par5./ /par16./
```

Perform editing operations

```
cat cobol0[0-3] > file
```

Split a file at every 100 lines, up to 10,000 lines (100 files). The **-k** option causes the created files to be retained if there are fewer than 10,000 lines (an error message is still printed).

```
csplit -k file 100 '{99}'
```

Assuming that **prog.c** follows the normal C coding convention of terminating routines with a **}** at the beginning of the line, create a file containing each separate C routine (up to 21) in **prog.c**.

```
csplit -k prog.c '%main(%' '/^}/+1' '{20}'
```

SEE ALSO

sh(1), split(1), environ(5), lang(5), regexp(5).

STANDARDS CONFORMANCE

csplit: SVID2, SVID3, XPG2, XPG3, XPG4

NAME

ct - spawn getty to a remote terminal (call terminal)

SYNOPSIS

ct [**-w** *n*] [**-x** *n*] [**-h**] [**-v**] [**-s** *speed*] *telno*...

DESCRIPTION

ct dials *telno*, the telephone number of a modem that is attached to a terminal, and spawns a *getty*(1M) process to that terminal.

ct tries each line listed in file **/etc/uucp/Devices** until it finds an available line with appropriate attributes or runs out of entries. If no lines are free, **ct** asks whether it should wait for a line, and if so, how many minutes it should wait before giving up. **ct** searches again for an available line at one-minute intervals until the specified limit is exceeded. Note that normally, **ct** disconnects the current tty line, so that the line can answer the incoming call. This is because **ct** assumes that the current tty line is connected to the terminal to spawn the **getty** process.

The *telno* argument specifies the telephone number, which can be composed of characters **0** through **9**, **-**, **=**, *****, and **#**. Use equal signs to signify secondary dial tones and minus signs for delays at appropriate places. The maximum length of *telno* is 31 characters. If more than one telephone number is specified, **ct** tries each in succession until one answers; this is useful for specifying alternate dialing paths.

When **ct** disconnects the current line, **getty** should not be spawned on this line if **ct** is going to make use of the same line to reconnect. To do this, set the entry for this line in the **inittab** file to **uugetty** instead of **getty** (see *inittab*(4)).

Options

ct recognizes the following options and command-line arguments:

-w*n*	Instruct **ct** to wait for a line a maximum of *n* number of minutes, if lines are busy. If this option is specified, **ct** does not query the user about whether to wait for a line.
-x*n*	Produce detailed output from program execution on the standard error output. This option is used for debugging. The debugging level *n* is a single digit; the most useful value is **-x9**.
-h	Prevent **ct** from disconnecting ("hanging up") the current tty line. This option is necessary if the user is using a different tty line than the one used by **ct** to spawn the **getty**.
-v	Verbose mode. The **-v** option is used with the **-h** option and causes **ct** to send a running narrative to the standard error output stream.
-s*speed*	Set the data rate where *speed* is expressed in baud. The default rate is **1200**.

After the user on the destination terminal logs out, **ct** prompts, **Reconnect?** If the response begins with the letter **n** the line is dropped. Otherwise, **getty** is restarted and the **login:** prompt is printed.

Of course, the destination terminal must be attached to a modem that can automatically answer incoming calls.

FILES

/var/adm/ctlog
/etc/uucp/Devices

SEE ALSO

cu(1), login(1), uucp(1), getty(1M), uugetty(1M).

NAME

ctags - create a tags file

SYNOPSIS

ctags [**-xvFBatwu**] *files* ...

DESCRIPTION

ctags makes a tags file for *ex*(1) (or *vi*(1)) from the specified C, Pascal and FORTRAN sources. A **tags** file gives the locations of specified objects (for C, functions, macros with argments, and typedefs; Pascal, procedures, programs and functions; FORTRAN, subroutines, programs and functions) in a group of files. Each line of the tags file contains the object name, the file in which it is defined, and an address specification for the object definition. Output is sorted in ascending collation order (see Environment Variables below). All objects except C *typedefs* are searched with a pattern, *typedefs* with a line number. Specifiers are given in separate fields on the line, separated by spaces or tabs. Using the *tags* file, **ex** can quickly find these objects' definitions.

-x Cause **ctags** to print a simple function index. This is done by assembling a list of function names, file names on which each function is defined, the line numbers where each function name occurs, and the text of each line. The list is then printed on the standard output. No *tags* file is created or changed.

-v Produce a page index on the standard output. This listing contains the function name, file name, and page number within that file (assuming 56-line pages to match *pr*(1)).

Files whose name ends in **.c** or **.h** are assumed to be C source files and are searched for C routine and macro definitions. Others are first examined to see if they contain any Pascal or FORTRAN routine definitions; if not, they are processed again looking for C definitions.

Other options are:

-F Use forward searching patterns (**/**...**/**) (default).

-B Use backward searching patterns (**?**...**?**).

-a Add the information from the files to the *tags* file. Unlike re-building the *tags* file from the original files, this can cause the same symbol to be entered twice in the *tags* file. This option should be used with caution and then only in very special circumstances.

-t Create tags for typedefs.

-w Suppress warning diagnostics.

-u Update the specified files in *tags*; that is, all references to those files are deleted, and the new values are added to the file as in **-a** above. (Beware: this option is implemented in a way which is rather slow; it is usually faster to simply rebuild the *tags* file.)

The tag **main** is treated specially in C programs. The tag formed is created by adding **M** to the beginning of name of the file, with any trailing **.c** removed, and leading pathname components also removed. This makes use of **ctags** practical in directories with more than one program.

EXTERNAL INFLUENCES

Environment Variables

LC_COLLATE determines the order in which the output is sorted.

LC_CTYPE determines the interpretation of the single- and/or multi-byte characters within comments and string literals.

If **LC_COLLATE** or **LC_CTYPE** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**. If any internationalization variable contains an invalid setting, **ctags** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported with the exception that multi-byte character file names are not supported.

DIAGNOSTICS

Too many entries to sort.
An attempt to get additional heap space failed; the sort could not be performed.

Unexpected end of function in file *file*, **line** *line*.
The tags file may be incorrect.

A **}** character was found unexpectedly in the first column. This can lead to incorrect entries in the *tags* file. **ctags** expects the source code to conform to the format as described on *cb*(1).

Duplicate entry in file *file*, **line** *line*: **name. Second entry ignored.**
The same name was detected twice in the same file. A *tags* entry was made only for the first name found.

Duplicate entry in files *file1* **and** *file2*: *name* **(Warning only).**
The same name was detected in two different files. A *tags* entry was made only for the first name found.

EXAMPLES

Create a tags file named **tags** in the current directory for all C source (***.c**) files and all header (***.h**) files in the current directory:

```
ctags *.[ch]
```

Once the tags file exists in the current directory, it can be used with commands that support tag files (such as **vi** (see *vi*(1)).

Use the tags file with **vi** to edit a particular function **myfunc()** located in one of the source files:

```
vi -t myfunc
```

While editing a C source file using **vi**, use the **ex**-mode tag command to edit function **myfunc()**:

```
:tag myfunc
```

Use **vi** to find **main()** in file **myprog.c**:

```
vi -t Mmyprog
```

While using **vi**, find **main()** in file **myprog.c** (does not have to be the file currently being edited):

```
:tag Mmyprog
```

WARNINGS

Recognition of **functions**, **subroutines**, and **procedures** for FORTRAN and Pascal is done in a very simple way. No attempt is made to deal with block structure; if there are two Pascal procedures in different blocks with the same name, a warning message is generated.

The method of deciding whether to look for C or Pascal and FORTRAN functions is an approximation, and can be fooled by unusual programs.

ctags does not know about **#ifdefs** and Pascal types.

It relies on the input being well formed to detect *typedefs*.

Use of **-tx** shows only the last line of typedefs.

ex is naive about *tags* files with several identical tags; it simply chooses the first entry its (non-linear) search finds with that tag. Such files can be created with either the **-u** or **-a** options or by editing a *tags* file.

If more than one (function) definition appears on a single line, only the first definition is indexed.

AUTHOR

ctags was developed by the University of California, Berkeley.

FILES

tags	output tags file
OTAGS	temporary file used by **-u**

SEE ALSO

cb(1), ex(1), vi(1).

STANDARDS CONFORMANCE

ctags: XPG4

C

NAME

cu - call another (UNIX) system; terminal emulator

SYNOPSIS

cu [**-s** *speed*] [**-l** *line*] [**-h**] [**-q**] [**-t**] [**-d** *level*] [**-e**|**-o**] [**-m**] [**-n**] [*telno*|*systemname*|**dir**]

XPG4 Syntax:

cu [**-s** *speed*] [**-l** *line*] [**-h**] [**-q**] [**-t**] [**-d**] [**-e**|**-o**] [**-m**] [**-n**] [*telno*|*systemname*|**dir**]

DESCRIPTION

cu calls up another system, which is usually a UNIX operating system, but can be a terminal or a non-UNIX operating system. **cu** manages all interaction between systems, including possible transfers of ASCII files.

Options

cu recognizes the following options and command-line arguments:

-s*speed*	Specify the transmission speed (110, 150, 300, 600, 1200, 2400, 3600, 4800, 7200, 9600, 19200). The default value is 300.
-l*line*	Specify a device name to use as the communication line. This can be used to override searching for the first available line having the right speed. When the **-l** option is used without the **-s** option, the speed of a line is obtained from file **/etc/uucp/Devices**. When the **-l** and **-s** options are used simultaneously, **cu** searches **/etc/uucp/Devices** to determine whether the requested speed for the requested line is available. If so, the connection is made at the requested speed; otherwise, an error message is printed and the call is not made. The specified device is usually a directly connected asynchronous line (such as **/dev/ttyapb**). In this case, a telephone number is not required, but the string **dir** can be used to specify that a dialer is not required. If the specified device is associated with an auto-dialer, a telephone number must be provided.
-h	Emulate local echo, supporting calls to other computer systems that expect terminals to be set to half-duplex mode.
-q	Use ENQ/ACK handshake (remote system sends ENQ, **cu** sends ACK.)
-t	Used when dialing an ASCII terminal that has been set to auto-answer. Appropriate mapping of carriage-return to carriage-return-line-feed pairs is set.
-d*level*	Print diagnostic traces. *level* is a number from 0-9, where higher *level*s produce more detail in the diagnostic messages.
-d	(XPG4 only.) Print diagnostic traces. The level is always 9.
-e (**-o**)	Generate even (odd) parity for data sent to the remote.
-m	Specify a direct line that has modem controls. Modem controls are ignored by **cu**.
-n	Cause the telephone number that **cu** dials to be requested interactively from the user rather than taking it from the command line.
telno	When using an automatic dialer, *telno* is the telephone number, with equal signs for secondary dial tone or minus signs for delays appropriately placed in the *telno* string.
systemname	A UUCP system name can be used instead of a telephone number (see *uucp*(1)); in this case, **cu** obtains an appropriate direct line or telephone number from **/etc/uucp/Systems** (including appropriate baud rate). **cu** dials each telephone number or direct line for *systemname* in the **Systems** file until a connection is made or all the entries are tried.
dir	Using **dir** ensures that **cu** uses the line specified by the **-l** option.

After making the connection, **cu** runs as two processes:

- *transmit* process reads data from the standard input and, except for lines beginning with **~**, passes it to the remote system;
- *receive* process accepts data from the remote system and, except for lines beginning with **~**, passes it to the standard output.

Normally, an automatic DC3/DC1 protocol is used to control input from the remote to ensure that the buffer is not overrun. "Prompt handshaking" can be used to control transfer of ASCII files to systems that have no type-ahead capability but require data to be sent only after a prompt is given. This is described in detail below. Lines beginning with **~** have special meanings.

Transmit Process Commands

The *transmit* process interprets the following commands:

~., **~..** Terminate the conversation. On hard-wired lines, **~.** sends several EOF characters to log out the session, whereas **~..** suppresses the EOF sequence. In general the remote hard-wired machine is unaware of the disconnect if **~..** is used. On dial-up connections, **~.** and **~..** do not differ.

~! Escape to an interactive shell on the local system.

~!*cmd* ... Run *cmd* on the local system (via **sh -c**).

~& Similar to **~!** but kill the receive process, restarting it upon return from the shell. This is useful for invoking sub-processes that read from the communication line where the receive process would otherwise compete for input.

~&*cmd* ... Run *cmd* on the local system (via **sh -c**) and kill the receive process, restarting it later.

~| *cmd* Pipe incoming data from the remote system through the standard input to *cmd* on the local system. To terminate, reset with either a **~&** or **~|** command.

~| Resets the receive process following a **~|cmd** command.

~$*cmd* ... Run *cmd* locally and send its output to the remote system.

~%cd Change the directory on the local system. *Note*: **~!cd** causes the command to be run by a sub-shell, causing a return to the current directory upon completion.

~%take *remote_source_file* [*local_destination_file*]
Copy file *remote_source_file* from the remote system to file *local_destination_file* on the local system. If *local_destination_file* is not specified, the *remote_source_file* argument is used in both places.

~%put *local_source_file* [*remote_destination_file*]
Copy file *local_source_file* on local system to file *remote_destination_file* on remote system. If *remote_destination_file* is not specified, the *local_source_file* argument is used in both places.

~~ ... Send the line **~** ... to the remote system. If you use **cu** on the remote system to access a third remote system, send **~~.** to cause the second remote **cu** to exit.

~%break Transmit a BREAK to the remote system.

~%nostop Toggle between DC3/DC1 input control protocol and no input control. This is useful if the remote system does not respond properly to the DC3 and DC1 characters.

~%<*file* Send the contents of the local file to the remote system using *prompt handshaking*. The specified file is read one line at a time, and each line is sent to the remote system when the *prompt sequence* is received. If no prompt is received by the time the *prompt timeout* occurs, the line is sent anyway. If the timeout is set to 0 seconds, or if the first character in the prompt sequence is a null character (**^@**), the handshake always appears to be satisfied immediately, regardless of whether or not the remote system generates a prompt. This capability is intended mainly to facilitate transfer of ASCII files from HP-UX to an HP 3000 system running MPE. This is usually accomplished by running the MPE **FCOPY** utility and giving the command **from=;to=***destfile***;new** and then running the **cu** input diversion to send the file to **FCOPY** which saves it in *destfile*. This facility might be useful with other systems also, such as an HP 1000 running RTE.

~%setpt *n* Specify the number of seconds to wait for a prompt before giving up. The default is 2 seconds. Specifying a timeout of 0 seconds disables handshaking; that is, handshake appears to complete immediately.

~%setps *xy* Set the handshake prompt to the characters *xy*. The default is DC1. The prompt can be any one or two characters. To specify a control character for *x* or *y*, use the Ctrl-X form where a circumflex (ASCII 94) precedes the character, as in **^X**. A null character can be specified with **^@**. (A null first character in the prompt implies a "null" prompt, which always appears to be satisfied.) A circumflex is specified by **^ ^**.

~%>[**>**]*file* Divert output from the remote system to the specified file until another **~%>** command is given. When an output diversion is active, typing **~%>** terminates it, whereas **~%>** *anotherfile* terminates it and begins a new one. The output diversion remains active through a **~&** subshell, but unpredictable results can occur if input/output diversions are intermixed with **~%take** or **~%put.** The **~%>>** command appends to the named file. Note that these commands, which are interpreted by the *transmit* process, are unrelated to the **~>** commands described below, which are interpreted by the receive process.

~*susp* Suspend the **cu** session. *susp* is the suspend character set in the terminal when **cu** was invoked (usually **^Z** — see *stty*(1)). As in all other lines starting with tilde, a **~***susp* line must be terminated by pressing **Return**.

Receive Process

The *receive* process normally copies data from the remote system to its standard output. A line from the remote that begins with **~>** initiates an output diversion to a file. The complete sequence is:

~>[**>**]**:** *file*
zero or more lines to be written to *file*
~>

Data from the remote is diverted (or appended, if **>>** is used) to *file*. The trailing **~>** terminates the diversion.

The use of **~%put** requires *stty*(1) and *cat*(1) on the remote side. It also requires that the current erase and kill characters on the remote system be identical to the current ones on the local system. Backslashes are inserted at appropriate places.

The use of **~%take** requires that the remote system support the **echo** and **cat** commands (see *echo*(1) and *cat*(1). Also, **stty tabs** mode should be set on the remote system if tabs are being copied without expansion. When connecting to a machine that uses the eighth bit as a parity bit, **stty istrip** mode should be set on the local system.

When **cu** is used on system X to connect to system Y and subsequently used on system Y to connect to system Z, commands on system Y can be executed if **~~** is used. For example, using the keyboard on system X, **uname** can be executed on Z, X, and Y as follows where lines 1, 3, and 5 are keyboard commands, and lines 2, 4, and 6 are system responses:

```
uname
Z
~!uname
X
~~!uname
Y
```

In general, **~** causes the command to be executed on the original machine; **~~** causes the command to be executed on the next machine in the chain.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages are displayed.

If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**. If any internationalization variable contains an invalid setting, **cu** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported.

DIAGNOSTICS

Exit code is zero for normal exit; non-zero (various values) otherwise.

EXAMPLES

To dial a system whose number is 9 201 555 1212 using 1200 baud:

```
cu -s1200 9=2015551212
```

If the speed is not specified, 300 is the default value.

To log in on a system connected by a direct line:

cu -l/dev/tty*XpX* **dir**

To dial a system with the specific line and a specific speed:

cu -s1200 -l/dev/tty*XpX* **dir**

To dial a system using a specific line:

cu -l/dev/cul*XpX* **2015551212**

To use a system name (**yyyzzz**):

```
cu yyyzzz
```

To connect directly to a modem:

```
cu -l/dev/culXX -m dir cu -l/dev/cu1XX -m dir
```

WARNINGS

cu buffers input internally.

AUTHOR

cu was developed by AT&T and HP.

FILES

```
/etc/uucp/Systems
/etc/uucp/Devices
/etc/uucp/Dialers
/var/spool/locks/LCK..(tty-device)
/dev/null
```

SEE ALSO

cat(1), ct(1), echo(1), stty(1), uname(1), uucp(1), uuname(1).

STANDARDS CONFORMANCE

cu: SVID2, SVID3, XPG2, XPG3, XPG4

NAME

cue - HP Character-Terminal User Environment (CUE)

SYNOPSIS

/usr/bin/cue

DESCRIPTION

CUE provides an easy-to-use, attractive, customizable environment that allows users on Series 800 HP-UX systems to easily identify themselves to the system and begin a work session. See DEPENDENCIES for supported terminal types.

A menubar is available for changing the native language of the session, changing the type of session to start upon a successful login, or getting on-line help. To obtain context-sensitive help at any time, press the function key labeled HELP (*f*1).

A pulldown menu and function keys (*f*1-*f*8) are displayed, allowing the user to modify various options or to get help. Before the login is initiated, the user has the option of interactively changing the native language of the session and the type of session to start upon a successful login.

The default native language is C, but the language is easily modifiable by entering the Language Menu which is accessible by selecting the Configuration item in the menu bar. The native language can also be specified as a parameter to **cuegetty** (see *cuegetty*(1M)).

The default session type is the POSIX shell, **sh**, but the session type can be easily changed to **tsm**, **keysh**, or **csh** by entering the Session Type Menu which is accessible by selecting the Configuration item in the menu bar.

The following standard **login** features are available:

- password aging
- logging invalid login attempts in **/var/adm/btmp**
- list of valid *ttys* for super-user login

CUE displays a visual screen that prompts for the *username* and corresponding *password*. If your username does not have a password, press the <carriage return> key to skip this field. Terminal echo is turned off (where possible) during typing of the password so that it will not appear on any written record of the session. After three unsuccessful login attempts, a *hangup* signal is issued.

If password aging has been invoked by the super-user on your behalf, your password may have expired. In this case, you will be diverted into **passwd** to change it, after which you can attempt to login again. See *passwd*(1).

If login is not successfully completed within a certain period of time (e.g., five minutes), the terminal may be silently disconnected.

After a successful login, the accounting files are updated, initializing the user and group ids, group access list, and working directory. If the session type chosen is *tsm*, the **SHELL** to start in each *tsm* session is determined from corresponding user entries in the **/etc/passwd** file. **cue** then forks the appropriate shell by using the last component of the shell pathname preceded by a **-** (for example, **-sh** or **-ksh**). When the session type is invoked with its name preceded by a minus in this manner, the shell performs its own initialization, including execution of profile, login, or other initialization scripts.

For example, if the user login shell is *sh*(1) or *ksh*(1) the shell executes the profile files **/etc/profile** and **$HOME/.profile** if they exist (and possibly others as well). Depending on the contents of the profile files, messages regarding mail in your mail file or any messages you may have received since your last login may be displayed. At this point, *cuesession* is started to perform accounting procedures, display messages, and start your session.

If **/var/adm/btmp** is present, all unsuccessful login attempts are logged to this file. This feature is disabled if the file is not present. A summary of bad login attempts can be viewed by users with appropriate privileges by using **lastb**, see *last*(1M).

If **/etc/securetty** is present, login security is in effect, meaning that only users with appropriate privileges are allowed to login successfully on the ttys listed in this file. Restricted ttys are listed by device name, one per line. Valid tty names are dependent on installation. Some examples could be **console**, **tty01**, **ttya1**, etc. Note that this feature does not inhibit a normal user from using **su**.

Starting Cue

There are several methods that can be used to start **cue**.

- An entry for **cuegetty** can be placed in the **/etc/inittab** file. See *cuegetty*(1M)). This is the preferred method as the user does not need to do anything further to start **cue**.
- Start **cue** from the command line by typing: **cue**.
- Start **cue** by making it the last entry in the user's **.login** configuration file.

Multiple **cue** logins may run simultaneously on separate terminals attached to the same local host. **cuegetty** can be configured in the **/etc/inittab** file for all users.

Remote users to the CUE system must access CUE by entering the **cue** command at the command-line prompt or as the last item in the user's **.login** configuration file.

EXTERNAL INFLUENCES

Environment Variables

cue invokes the user's session with the following default environment:

CUESESSION is set to the session type selected. Valid values are:

/usr/bin/sh	POSIX Shell (DEFAULT)
/usr/bin/tsm	manages up to 10 sessions at once
/usr/bin/keysh	Easy Context-Sensitive Softkey Shell
/usr/bin/ksh	Korn Shell
/usr/bin/csh	C Shell

HOME	is set to the home directory of the user
LANG	is set to the native language selected (C is the default)
LOGNAME	is set to the user name
MAIL	is set to **/var/mail/$LOGNAME**
NLSPATH	is set to the path applications search for NLS message catalogs, usually **/usr/lib/nls/%L/%N.cat**
PATH	is set to the path to be searched for commands **:/usr/bin**
SHELL	is set to the user's default shell (from **/etc/passwd**)

Several methods are available to modify or add to this list depending on the desired scope of the resulting environment variable.

Basic environment variables can be set for all CUE users on a system by setting the values in **/etc/profile** and **/etc/csh.login**. Personal environment variables can be set on a per-user basis in the script file **$HOME/.profile** for *sh* and *ksh* users or **.cshrc** for *csh* users.

Note that alias and function definitions need to be included in the file specified by **ENV** for *ksh*, as this file will be sourced for each invocation of the shell. For *csh* users, the **.cshrc** file should be structured such that it cannot generate any output on standard output or standard error, including occasions when it is invoked without an affiliated terminal. The **rcp** command sources the **.cshrc** file and any output generated by this file, even to standard error, causes problems. Commands such as **stty** should be placed in the **.login** file, not in **.cshrc**, so that their output cannot affect **rcp**.

For users with appropriate privileges, **PATH** is augmented to include **/etc**.

VT320 Terminal Support

Because the VT320 terminal has predefined local functions for keys labeled as F1, F2, F3 and F4, users should use following mapping when they desire to use function keys:

HP or Wyse60	VT320 or HP 700/60 in VT320 mode
F1	PF2 **!**
F2	PF1 **!**
F3	space bar
F4	PF3 **!**
F5	F10, [EXIT], F5 *****
F6	none
F7	F18, first unlabeled key to right of Pause/Break*
F8	F19, second unlabeled key to right of Pause/Break*

***** When using PC-AT keyboard with HP 700/60 in VT320 mode

! See "Configuration: HP 700/60 in DEC mode, or DEC terminals with PC-AT type keyboard"

Further, since DEC terminals do not support softkey menu, no such menu is displayed on these terminals.

Many applications tend to use TAB for forward navigation (moving from one field to another) and shift-TAB is used for backward navigation. Users having DEC terminals or using terminals in DEC emulation modes such as VT100 or VT320 may note that these terminals/emulators may give out same character for TAB and shift-TAB. As such, it is impossible for an application to distinguish between TAB and shift-TAB, and both of them treated as if a TAB key was pressed. It might present slight overhead to users in case they want to go backwards. Now instead, they should complete rest of the inputs and get back to the desired field later.

VT100 Terminal Support

VT100 does not allow the (f1-f8) function keys to be configured. Therefore, the following keyboard mappings will apply to VT100 terminals:

HP or Wyse60	VT100 or HP 700/60 in VT100 mode
F1	PF2 **!**
F2	PF1 **!**
F3	space bar
F4	[PF3], [space bar] or [PF3], [=] **!**
F5	return
F6	none
F7	none
F8	none

! See "Configuration: HP 700/60 in DEC mode, or DEC terminals with PC-AT type keyboard"

Further, since DEC terminals do not support softkey menu, no such menu is displayed on these terminals.

Many applications tend to use TAB for forward navigation (moving from one field to another) and shift-TAB is used for backward navigation. Users having DEC terminals or using terminals in DEC emulation modes such as VT100 or VT320 may note that these terminals/emulators may give out same character for TAB and shift-TAB. As such, it is impossible for an application to distinguish between TAB and shift-TAB, and both of them treated as if a TAB key was pressed. It might present slight overhead to users in case they want to go backwards. Now instead, they should complete rest of the inputs and get back to the desired field later.

Configuration: HP 700/60 terminal in DEC mode, or DEC terminal with PC-AT type keyboard

Customers using the following configuration may want to be aware of the following keyboard difference.

It may be possible for a user with the "HP 700/60 terminal in DEC mode, or DEC terminal with PC-AT type keyboard" configuration to be told to press function key F1 through F4 to achieve some desired result. For HP 700/60 terminal in DEC mode or DEC terminals, these functions keys may be mapped onto PF1-PF4 keys. (see "Keyboard Mappings"). However, the PC-AT type keyboard does not provide PF1, PF2, PF3, or PF4 keys, as does the DEC/ANSI keyboard.

Keyboard Mappings

"Num Lock"	maps to "PF1"
"/"	maps to "PF2"
"*"	maps to "PF3"
"-"	maps to "PF4"

The "Num Lock", "/", "*", and "-" keys are located on the keyboard, in a row, above the number pad on the right side of the keyboard. Please note that although this keyboard is called a PC-AT type keyboard, it is supplied by HP. A PC-AT type keyboard can be recognized by location of ESC key at the left-top of the keyboard.

Wyse60 Terminal Support

On Wyse60, use DEL (located next to Backspace) key to backspace. On an HP 700/60 with a PC-AT type keyboard in Wyse60 mode, the DEL key is located in the bottom row on the number pad.

Wyse60 terminals provide a single line to display softkey labels unlike HP terminals which provide two lines. Sometimes this may result in truncated softkey labels. For example, "Help on Context" label for F1 may appear as "Help on C". Some standard labels for screen-oriented applications such as SAM and swinstall are as follows:

On wyse60 may appear as ..	means
Help On C	Help On Context
Select/D	Select/Deselect
Menubar	Menubar on/off

Internationalization

All screens, labels, and messages are localizable. The message catalog **cue.cat** contains the localized representations of the default labels and messages. **cue** will read the appropriate message catalog indicated by the **LANG** environment variable and display the localized strings. By selecting a native language in the Language Menu, the language of the CUE screens and the future work session can be specified. If the the message catalog exists for **cue** in the language selected, **cue** will be redisplayed in that language. If not, the CUE screens will continue in the current language and the work session that is started after a successful login will be started in the language selected. In either case, the **LANG** environment variable will be set appropriately for the resulting work session.

If **cue** will be started on the command line or as the last item in the **.login** file, the CUE screens will be brought up using the language specified by the **LANG** environment variable. If CUE screens do not exist, then the default native language, C, will be used.

To use **cue** with Asian languages, the **AWTERM** environment variable must be set to **hpterm-asian**. This will allow the text in Asian fonts to be displayed properly by **cue**.

If **cue** will be started by **cuegetty**, it is possible to start up the CUE Login Screens in a language other than the default, C, by invoking **cuegetty** with the **-L** *nls_language* option. Of course, CUE screens and the **cue.cat** file must exist for the *nls_language* specified.

DEPENDENCIES

CUE is available only on Series 800 systems, and is compatible only with the following terminals:

HP 700/92 HP 700/94 HP 2392 HP 2394 VT 320 VT 100 WYSE 60

WARNINGS

cue is an HP proprietary command, which will be *obsoleted* in a future release, and is not portable to other vendor's platforms.

FILES

/var/adm/btmp	history of bad login attempts
/etc/logingroup	group file - defines group access lists
/etc/motd	message-of-the-day
/etc/passwd	password file - defines users, passwords, and primary groups
/etc/profile	system profile (initialization for all users)
/etc/securetty	list of valid ttys for root login
/etc/utmp	users currently logged in
/var/adm/wtmp	history of logins, logouts, and date changes
/var/mail/*your-name*	mailbox for user *your-name*
/usr/bin/cue	*cue* executable
/usr/sbin/cuegetty	*cuegetty* executable
/usr/newconfig/etc/cue.inittab	template for */etc/inittab*
/etc/cue.dm	screen descriptions
/usr/lbin/cuesession	starts selected session type
/usr/lib/nls/$LANG/cue.cat	NLS message catalog
/usr/share/man/man1.Z/cue.1	man page for *cue*(1)
/usr/share/man/man1m.Z/cuegetty.1m	man page for *cuegetty*(1M)

SEE ALSO

csh(1), cuegetty(1M), env(1), keysh(1), ksh(1), login(1), passwd(1), sh(1), tsm(1), btmp(4), environ(5), hpnls(5), lang(5).

NAME

cut - cut out (extract) selected fields of each line of a file

SYNOPSIS

cut -c *list* [*file* ...]

cut -b *list* [**-n**] [*file* ...]

cut -f *list* [**-d** *char*] [**-s**] [*file* ...]

DESCRIPTION

cut cuts out (extracts) columns from a table or fields from each line in a file; in data base parlance, it implements the projection of a relation. Fields as specified by *list* can be fixed length (defined in terms of character or byte position in a line when using the **-c** or **-b** option), or the length can vary from line to line and be marked with a field delimiter character such as the tab character (when using the **-f** option). **cut** can be used as a filter; if no files are given, the standard input is used.

When processing single-byte character sets, the **-c** and **-b** options are equivalent and produce identical results. When processing multi-byte character sets, when the **-b** and **-n** options are used together, their combined behavior is very similar, but not identical to the **-c** option.

Options

Options are interpreted as follows:

list A comma-separated list of integer *byte* (-b option), *character* (-c option), or *field* (-f option) numbers, in increasing order, with optional **-** to indicate ranges. For example:

1,4,7	Positions 1, 4, and 7.
1-3,8	Positions 1 through 3 and 8.
-5,10	Positions 1 through 5 and 10.
3-	Position 3 through last position.

-b *list* Cut based on a list of bytes. Each selected byte is output unless the **-n** option is also specified.

-c *list* Cut based on character positions specified by *list* (**-c 1-72** extracts the first 72 characters of each line).

-f *list* Where *list* is a list of fields assumed to be separated in the file by a delimiter character (see **-d**); for example, **-f 1,7** copies the first and seventh field only. Lines with no field delimiters will be passed through intact (useful for table subheadings), unless **-s** is specified.

-d *char* The character following **-d** is the field delimiter **(**-f option only). Default is *tab*. Space or other characters with special meaning to the shell must be quoted. Adjacent field delimiters delimit null fields. *char* may be an international code set character.

-n Do not split characters. If the high end of a range within a list is not the last byte of a character, that character is not included in the output. However, if the low end of a range within a list is not the first byte of a character, the entire character *is* included in the output."

-s Suppresses lines with no delimiter characters when using **-f** option. Unless **-s** is specified, lines with no delimiters appear in the output without alteration.

Hints

Use **grep** to extract text from a file based on text pattern recognition (using regular expressions). Use **paste** to merge files line-by-line in columnar format. To rearrange columns in a table in a different sequence, use **cut** and **paste**. See *grep*(1) and *paste*(1) for more information.

EXTERNAL INFLUENCES

Environment Variables

LC_CTYPE determines the interpretation of text as single and/or multi-byte characters.

If **LC_CTYPE** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**. If any internationalization variable contains an invalid setting, **cut** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

cut supports both single- and multi-byte character code sets. International code set characters may be specified in the *char* given to the **-d** option. **cut** recognizes the international code set characters according to the locale specified in the **LC_CTYPE** environment variable.

EXAMPLES

Password file mapping of user ID to user names:

```
cut -d : -f 1,5 /etc/passwd
```

Set environment variable **name** to current login name:

```
name=‘who am i | cut -f 1 -d " "‘
```

Convert file **source** containing lines of arbitrary length into two files where **file1** contains the first 500 bytes (unless the 500th byte is within a multi-byte character), and **file2** contains the remainder of each line:

```
cut -b 1-500 -n source > file1
cut -b 500- -n source > file2
```

DIAGNOSTICS

line too long
Line length must not exceed **LINE_MAX** characters or fields, including the new-line character (see *limits*(5)).

bad list for b/c/f option
Missing **-b**, **-c**, or **-f** option or incorrectly specified *list*. No error occurs if a line has fewer fields than the *list* calls for.

no fields *list* is empty.

WARNINGS

cut does not expand tabs. Pipe text through *expand*(1) if tab expansion is required.

Backspace characters are treated the same as any other character. To eliminate backspace characters before processing by **cut**, use the **fold** or **col** command (see *fold*(1) and *col*(1)).

AUTHOR

cut was developed by OSF and HP.

SEE ALSO

grep(1), paste(1).

STANDARDS CONFORMANCE

cut: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

NAME

date - display or set the date and time

SYNOPSIS

date [**-u**]

date [**-u**] **+***format*

date [**-u**] [*mmddhhmm*[[*cc*]*yy*]]

date [**-a** [**-**]*sss*[**.***fff*]]

DESCRIPTION

The **date** command displays or sets the current HP-UX system clock date and time. Since the HP-UX system operates in Coordinated Universal Time (UTC), **date** automatically converts to and from local standard or daylight/summer time, based on your **TZ** environment variable. See *Environment Variables* in EXTERNAL INFLUENCES below.

Options

date recognizes the following option:

-u Input and output values in Coordinated Universal Time (UTC), functionally equivalent to Greenwich Mean Time (GMT), instead of in local time.

-a [**-**]*sss*[**.***fff*]
Slowly adjust the time by *sss***.***fff* seconds (*fff* represents fractions of a second). This adjustment can be positive or negative. The system's clock will be sped up or slowed down until it has drifted by the number of seconds specified.

Formats

The **date** command has two forms for displaying the date and time and one form for setting them.

date [**-u**]

Display the current date and time. The output is the same as for the **%c** formatting directive for all languages except the **C** default language. See *Formatting Directives* and EXAMPLES below.

date [**-u**] **+***format*

Display the current date and time according to formatting directives specified in *format*, which is a string of zero or more formatting directives and ordinary characters. If it contains blanks, enclose it in apostrophes or quotation marks.

See *Formatting Directives* below.

All ordinary characters are copied unchanged into the output string.

The output string is always terminated with a newline character.

If **+** is specified and *format* is omitted, only a newline is output.

date [**-u**] [*mmddhhmm*[[*cc*]*yy*]]

Set the HP-UX system clock to the date and time specified. You require the superuser privilege.

If you include the **-u** option, the specified date and time is assumed to be in Coordinated Universal Time (UTC).

The numeric argument is interpreted left to right in two-digit pairs as follows:

mm	Month number [**01**-**12**].
dd	Day number in the month [**01**-**31**].
hh	Hour number (24-hour system) [**00**-**23**].
mm	Minute number [**00**-**59**].
cc	Century minus one [**19**-**20**].
yy	Last two digits of the year number [**70**-**99**, **00**-**37** (1970-1999, 2000-2037)]. If omitted, the current year is used.

If you attempt to set the date backwards, **date** generates the warning,

do you really want to run time backwards?[yes/no]

Type **yes** or the equivalent for your locale to set the clock backwards; anything else to cancel the command.

When **date** is used to set the date, a pair of date change records is written to the file **/var/adm/wtmp**.

(XPG4 only.) No warning is generated if date is set backwards.

Formatting Directives

The following formatting directives, shown without the optional field width and precision specification, are replaced by the indicated characters. If a directive is not one of the following, the result is undefined.

The output for digits, characters, and words depends on the language/locale settings. See *Environment Variables* in EXTERNAL INFLUENCES below.

The examples assume that the **date** command was executed on Wednesday, January 12, 1994 at 7:45:58 p.m. Pacific Standard Time, using the **C** default language.

%a	Abbreviated weekday name. For example, **Wed**.
%A	Full weekday name. For example, **Wednesday**.
%b	Abbreviated month name. For example, **Jan**.
%B	Full month name. For example, **January**.
%c	Current date and time representation. For example, **Wed Jan 12 19:45:58 1994**.
%C	Century (the year divided by 100 and truncated to an integer) as a two-digit decimal number [**00**-**99**]. For example, **19**.
%d	Day of the month as a two-digit decimal number [**01**-**31**]. For example, **12**.
%e	Day of the month as a two-character decimal number with leading space fill [**" 1"**-**"31"**]. For example, **12**.
%E	Combined Emperor/Era name and year.
%H	Hour (24-hour clock) as a two-digit decimal number [**00**-**23**]. For example, **19**.
%I	Hour (12-hour clock) as a two-digit decimal number [**01**-**12**]. For example, **07**.
%j	Day of the year as a three-digit decimal number [**001**-**366**]. For example, **012**.
%m	Month as a decimal two-digit number [**01**-**12**]. For example, **01**.
%M	Minute as a decimal two-digit number [**00**-**59**]. For example, **45**.
%n	Newline character.
%N	Emperor/Era name.
%o	Emperor/Era year.
%p	Equivalent of either AM or PM. For example, **PM**.
%R	Time as %H:%M
%S	Second as a two-digit decimal number (allows for possible leap seconds) [**00**-**61**]. For example, **58**.
%t	Tab character.
%u	Weekday as a one-digit decimal number [**1**-**7** (Monday-Sunday)]. For example, **3**.
%U	Week number of the year (Sunday as the first day of the week) as a two-digit decimal number [**00**-**53**]. All days that precede the first Sunday in the year are considered to be in week **00**. For example, **02**.
%V	Week number of the year (Monday as the first day of the week) as a two-digit decimal number [**01**-**53**]. If the week containing January 1 has four or more days in the new year (January 1 is Thursday or sooner), it is designated as week **01**; otherwise, (January 1 is Friday or later), it is designated as week **53** of the previous year, and the next week is week **01**. For example, **02**.

%w Weekday as a one-digit decimal number [**0**-**6** (Sunday-Saturday)]. For example, **3**.

%W Week number of the year (Monday as the first day of the week) as a two-digit decimal number [**00**-**53**]. All days that precede the first Monday in the year are considered to be in week **00**. For example, **02**.

%x Current date representation. For example, **01/12/94**.

%X Current time representation. For example, **19:45:58**.

%y Year without century as a two-digit decimal number [**00**-**99**]. For example, **93**.

%Y Year with century as a four-digit decimal number [**1970**-**2037**]. For example, **1994**.

%Z Time zone name (or no characters if time zone cannot be determined). For example, **PST**.

%% The **%** character.

Obsolescent Directives

The following directives are provided for backward compatibility. It is recommended that the preceding directives be used instead.

%D Date in usual U.S. format. For example, **01/12/94**. Use **%x** or **%m/%d/%y** instead.

%F Full month name. For example, **January**. Use **%B** instead.

%h Abbreviated month name. For example, **Jan**. Use **%b** instead.

%r Time in 12-hour U.S. format. For example, **07:45:58 PM**. Use **"%I:%M:%S %p"** instead.

%T Time in 24-hour U.S. format. For example, **19:45:58**. Use **%X** or **%H:%M:%S** instead.

%z Time zone name (or no characters if time zone cannot be determined). For example, **PST**. Use **%Z** instead.

Modified Formatting Directives

Some Formatting Directives can be modified by the **E** and **O** modifier characters to indicate a different format or specification for the language specified in the **LC_TIME** environment variable.

If the corresponding keyword (**era**, **era_year**, **era_d_fmt**, and **alt_digit**) is not specified or not supported, the unmodified field descriptor value is used. The command

LC_ALL=*language* **locale -ck era era_year era_d_fmt alt_digit**

displays the keywords and their values in the specified *language* (see *locale*(1)).

%Ec Alternate appropriate date and time representation.

%EC The name of the base year in alternate representation.

%Ex Alternate date representation.

%Ey Offset from **%EC** (year only) in the alternate representation.

%EY Full alternate year representation.

%Od Day of month using the alternate numeric symbols.

%Oe Day of month using the alternate numeric symbols with leading space-character fill if applicable.

%OH Hour (24-hour clock) using the alternate numeric symbols.

%OI Hour (12-hour clock) using the alternate numeric symbols.

%Om Month using the alternate numeric symbols.

%OM Minutes using the alternate numeric symbols.

%OS Seconds using the alternate numeric symbols.

%OU Week number of the year (Sunday is the first day of the week) using the alternate numeric symbols.

%Ow Weekday as number using the alternate numeric symbols (Sunday=**0**).

%OW Weekday number of the year (Monday is the first day of the week) using the alternate numeric symbols.

%Oy Year (offset from **%C**) in alternate representation.

Field Width and Precision

An optional field width and precision specification can immediately follow the initial **%** of a formatting directive in the following order:

[**-**|**0**]*width* The decimal digit string *width* specifies a *minimum* field width in which the result of the conversion is right- or left-justified. The default is right-justified with space padding on the left. If the string starts with "**-**", the result is left-justified with space padding on the right. If the string starts with "**0**", the result is right-justified and padded with zeros on the left.

.*prec* The decimal digit string *prec* specifies the *minimum* number of digits to appear for the **d**, **H**, **I**, **j**, **m**, **M**, **o**, **S**, **U**, **w**, **W**, **y**, and **Y** numeric directives. If a directive supplies fewer digits than specified by the precision, it will be expanded with leading zeros.

prec specifies the *maximum* number of characters to be used from the **a**, **A**, **b**, **B**, **c**, **D**, **E**, **F**, **h**, **n**, **N**, **p**, **r**, **t**, **T**, **x**, **X**, **z**, **Z**, and **%** text directives. If a directive supplies more characters than specified by the precision, excess characters are truncated on the right.

If no field width or precision is specified for a **d**, **H**, **I**, **m**, **M**, **S**, **U**, **W**, or **y** directive, the default is **.2**; for the **j** directive, the default is **.3**; for **Y**, the default is **.4**; for **w**, the default is **.1**.

EXTERNAL INFLUENCES

Environment Variables

LC_CTYPE determines the interpretation of the bytes within the *format* string as single- and/or multi-byte characters.

LC_NUMERIC determines the characters used to form numbers for those directives that produce numbers in the output. The characters used are those defined by **alt_digit** (see *locale*(1) and **ALT_DIGIT** in *langinfo*(5)).

LC_TIME determines the content (for example, the weekday names produced by the **%a** directive) and format (for example, the current time representation produced by the **%X** directive) of date and time strings output by the **date** command.

LC_MESSAGES determines the language in which messages (other than the date and time strings) are displayed.

If **LC_CTYPE**, **LC_NUMERIC**, **LC_TIME**, or **LC_MESSAGES** is not specified or is null, it defaults to the value of **LANG**.

If **LANG** is not specified or is null, it defaults to **C** (see *lang*(5)).

If any internationalization variable contains an invalid setting, all internationalization variables default to **C** (see *environ*(5)).

TZ determines the conversion between the system time in UTC and the time in the user's local time zone. See *environ*(5) and *tztab*(4). **TZ** also determines the content (that is, the time-zone name produced by the **%z** and **%Z** directives) of date and time strings output by the **date** command.

If **TZ** is not set or is set to the empty string, its default value is **EST5EDT**.

International Code Set Support

Single- and multi-byte character code sets are supported.

DIAGNOSTICS

The following messages may be displayed.

bad conversion

The date/time specification is syntactically incorrect. Check it against the usage and for the correct range of each of the digit-pairs.

bad format character - *c*

The character *c* is not a valid format directive, field width specifier, or precision specifier.

do you really want to run time backwards?[yes/no]

The date/time you specified is earlier than the current clock value. Type **yes** (or the equivalent for your locale) to set the clock backwards; anything else to cancel the command.

no permission

You need the superuser privilege to change the date.

EXAMPLES

Date in Different Languages

Display the date. In this example, the **TZ** environment variable contains **PST8PDT**, and the language environment variables are set as noted.

```
date    → Fri Aug 20 15:03:37 PDT 1993  ← C (default)
date -u → Fri Aug 20 22:03:37 UTC 1993  ← C (default)
date    → Fri, Aug 20, 1993 03:03:37 PM ← en_US.roman8 (U.S. English)
date    → Fri. 20 Aug, 1993 03:03:37 PM ← en_GB.roman8 (U.K. English)
date    → 20/08/1993 15.47.47           ← pt_PT.roman8 (Portuguese)
```

Set Date

Set the date to Oct 8, 12:45 a.m.

```
date 10080045
```

Display Formatted Date

Display the current date and time using a format. Note the use of quotation marks due to the blanks in the format.

```
date "+DATE: %m/%d/%y%nTIME: %H:%M:%S"
```

The output resembles the following:

```
DATE: 10/08/87
TIME: 12:45:05
```

Display Formatted Date Using Local Language Conversion

With the date as set in the "Set Date" example above and **LC_TIME** set to **de_De.roman8** (German):

```
date +'%-4.4h %2.1d %H:%M'
```

generates output similar to:

```
Okt   8 12:45
```

where the month field is four characters long, flush-left, and space-padded on the right if the month name is shorter than four characters. The day field is two characters long, with leading zeros suppressed.

WARNINGS

The former HP-UX format directive **A** has been changed to **W** for ANSI compatibility.

Changing the date while the system is running in multiuser mode should be avoided to prevent disrupting user-scheduled and time sensitive programs and processes. Also, changing the date can cause *make*(1) and the SCCS and *cron*(1M) subsystems to behave in an unexpected manner. The **cron** daemon should be killed prior to setting the date backwards, then restarted. SCCS files should be checked with the **val** command (see *val*(1)) if deltas have been made while the clock was wrongly set.

The following formatting directives may be deleted from future releases: **%E**, **%F**, **%o**, **%z**.

Currently, the maximum date supported is December 31, 2037 23:59:00 UTC.

AUTHOR

date was developed by AT&T and HP.

FILES

/var/adm/wtmp

SEE ALSO

locale(1), stime(2), ctime(3C), strftime(3C), tztab(4), environ(5), lang(5), langinfo(5).

STANDARDS CONFORMANCE

`date`: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

NAME

dc - desk calculator

SYNOPSIS

dc [*file*]

DESCRIPTION

dc is an arbitrary precision arithmetic package. Ordinarily it operates on decimal integers, but one may specify an input base, output base, and a number of fractional digits to be maintained. (See *bc*(1), a preprocessor for **dc** that provides infix notation and a C-like syntax that implements functions. **bc** also provides reasonable control structures for programs.) The overall structure of **dc** is a stacking (reverse Polish) calculator. If an argument is given, input is taken from that file until its end, then from the standard input. An end of file on standard input or the **q** command stop dc. The following constructions are recognized:

number — The value of the number is pushed on the stack. A number is an unbroken string of the digits **0**-**9** or **A**-**F**. It can be preceded by an underscore (_) to input a negative number. Numbers can contain decimal points.

+ - / * % ^
: The top two values on the stack are added (**+**), subtracted (**-**), multiplied (*****), divided (**/**), remaindered (**%**), or exponentiated (**^**). The two entries are popped off the stack; the result is pushed on the stack in their place. Any fractional part of an exponent is ignored and a warning generated. The remainder is calculated according to the current scale factor; it is not the integer modulus function. **7 % 3** yields .1 (one tenth) if scale is 1 because **7 / 3** is 2.3 with .1 as the remainder.

s*x* — The top of the stack is popped and stored into a register named *x*, where *x* can be any character. If the **s** is capitalized, *x* is treated as a stack and the value is pushed on it.

l*x* — The value in register *x* is pushed on the stack. Register *x* is not altered. All registers start with zero value. If the **l** is capitalized, register *x* is treated as a stack and its top value is popped onto the main stack.

d — The top value on the stack is duplicated.

p — The top value on the stack is printed. The top value remains unchanged. **P** interprets the top of the stack as an ASCII string, removes it, and prints it.

f — All values on the stack are printed.

q — exits the program. If executing a string, the recursion level is popped by two. If **q** is capitalized, the top value on the stack is popped and the string execution level is popped by that value.

x — treats the top element of the stack as a character string and executes it as a string of **dc** commands.

X — replaces the number on the top of the stack with its scale factor.

[... **]** — puts the bracketed ASCII string onto the top of the stack. Strings can be nested by using nested pairs of brackets.

<*x* **>***x* **=***x*
!<*x* **!>***x* **!=***x*
: The top two elements of the stack are popped and compared. Register *x* is evaluated if they obey the stated relation.

v — Replaces the top element on the stack by its square root. Any existing fractional part of the argument is taken into account, but otherwise the scale factor is ignored.

! — Interprets the rest of the line as an HP-UX system command (unless the next character is **<**, **>**, or **=**, in which case appropriate relational operator above is used).

c — All values on the stack are popped.

i — The top value on the stack is popped and used as the number radix for further input.

I — pushes the input base on the top of the stack.

o	The top value on the stack is popped and used as the number radix for further output. See below for notes on output base.
O	pushes the output base on the top of the stack.
k	the top of the stack is popped, and that value is used as a non-negative scale factor: the appropriate number of places are printed on output, and maintained during multiplication, division, and exponentiation. The interaction of scale factor, input base, and output base will be reasonable if all are changed together.
K	pushes the scale factor on the top of the stack.
z	The stack level is pushed onto the stack.
Z	replaces the number on the top of the stack with its length.
?	A line of input is taken from the input source (usually the terminal) and executed.
; and **:**	Used by **bc** for array operations.
Y	Generates debugging output for **dc** itself.

The input base may be any number, but only the digits 0-9 and A-F are available for input, thus limiting the usefulness of bases outside the range 1-16. All 16 possible digits may be used in any base; they always take their conventional values.

The output base may be any number. Bases in the range of 2-16 generate the "usual" results, with the letters A-F representing the values from 10 through 16. Bases 0 and 1 generate a string of **1**s whose length is the value of the number. Base −1 generates a similar string consisting of **d**s. Other bases have each "digit" represented as a (multi-digit) decimal number giving the ordinal of that digit. Each "digit" is signed for negative bases. "Digits" are separated by spaces. Given the definition of output base, the command **Op** always yields "10" (in a representation appropriate to the base); **O1-p** yields useful information about the output base.

DIAGNOSTICS

x **is unimplemented**	Where *x* is an octal number.
stack empty	There are insufficient elements on the stack to do what was asked.
Out of space	The free list is exhausted (too many digits).
Out of headers	Too many numbers are being kept around.
Out of pushdown	Too many items are on the stack.
Nesting Depth	There are too many levels of nested execution.

EXAMPLES

This example prints the first ten values of n! (n factorial):

```
[la1+dsa*pla10>y]sy
0sa1
lyx
```

SEE ALSO

bc(1).

DC: An Interactive Desk Calculator tutorial in *Number Processing Users Guide*.

NAME

dd - convert, reblock, translate, and copy a (tape) file

SYNOPSIS

dd [*option*=*value*] ...

DESCRIPTION

dd copies the specified input file to the specified output file with possible conversions. The standard input and output are used by default. Input and output block size can be specified to take advantage of raw physical I/O. Upon completion, **dd** reports the number of whole and partial input and output records.

Options

dd recognizes the following *option*=*value* pairs:

if=*file*	Input file name; default is standard input.
of=*file*	Output file name; default is standard output. The output file is created using the same owner and group used by **creat()**.
ibs=*n*	Input block size is *n* bytes; default is 512.
obs=*n*	Output block size is *n* bytes; default is 512.
bs=*n*	Set both input and output block size to the same size, superseding **ibs** and **obs**. This option is particularly efficient if no conversion (**conv** option) is specified, because no in-core copy is necessary.
cbs=*n*	Conversion buffer size is *n* bytes.
skip=*n*	Skip *n* input blocks before starting copy.
iseek=*n*	Skip *n* input blocks before starting copy. (This is an alias for the **skip** option.)
seek=*n*	Skip *n* blocks from beginning of output file before copying.
oseek=*n*	Skip *n* blocks from beginning of output file before copying. (This is an alias for the **seek** option.)
count=*n*	Copy only *n* input blocks.
files=*n*	Copy and concatenate *n* input files. This option should be used only when the input file is a magnetic tape device.

conv=*value* [**,***value* ...]

Where *value*s are comma-separated symbols from the following list.

ascii	Convert EBCDIC to ASCII.
ebcdic	Convert ASCII to EBCDIC.
ibm	Convert ASCII to EBCDIC using an alternate conversion table.
	The **ascii**, **ebcdic**, and **ibm** values are mutually exclusive.
block	Convert each newline-terminated or end-of-file-terminated input record to a record with a fixed length specified by **cbs**. Any newline character is removed, and space characters are used to fill the block to size **cbs**. Lines that are longer than **cbs** are truncated; the number of truncated lines (records) is reported (see *DIAGNOSTICS* below).
	The **block** and **unblock** values are mutually exclusive.
unblock	Convert fixed-length input records to variable-length records. For each input record, **cbs** bytes are read, trailing space characters are deleted, and a newline character is appended.
lcase	Map upper-case input characters to the corresponding lower-case characters.
	The **lcase** and **ucase** values are mutually exclusive.
ucase	Map lower-case input characters to the corresponding upper-case characters.

swab	Swap every pair of input bytes.
noerror	Do not stop processing on an input error. If the **sync** conversion symbol is also specified, missing input is replaced with null bytes and processed normally; otherwise, the input block is omitted from the output.
notrunc	Do not truncate existing output file. Blocks in the output file not overwritten by this invocation of **dd** are preserved.
sync	Pad every input block to size **ibs**. If **block** or **unblock** is also specified, pad with space characters; otherwise, pad with null bytes.

Where sizes are required, *n* indicates a numerical value in bytes. Numbers can be specified using the forms:

n	for *n* bytes
*n***k**	for *n* Kbytes ($n \times 1024$),
*n***b**	for *n* blocks ($n \times 512$), or
*n***w**	for *n* words ($n \times 2$).

To indicate a product, use **x** to separate number pairs.

The **cbs** option is used when **block**, **unblock**, **ascii** or **ebcdic** conversion is specified. In case of **ascii**, *cbs* characters are placed into the conversion buffer, converted to ASCII, trailing blanks are trimmed, and a newline is added before sending the line to the output. In case of **ebcdic**, ASCII characters are read into the conversion buffer, converted to EBCDIC, and blanks are added to make up an output block of size *cbs*.

EXTERNAL INFLUENCES

International Code Set Support

Single- and multi-byte character code sets are supported.

Environment Variables

The following environment variables affect execution of **dd**:

LANG determines the locale when **LC_ALL** and a corresponding variable (beginning with **LC_**) do not specify a locale.

LC_ALL determines the locale used to override any values set by **LANG** or any environment variables beginning with **LC_**.

The **LC_CTYPE** variable determines the locale for the interpretation of sequences of bytes of text data as characters (single-byte/multi-byte characters, upper-case/lower-case characters).

The **LC_MESSAGES** variable determines the language in which messages are written.

RETURN VALUE

Exit values are:

0	Successful completion.
>0	Error condition occurred.

DIAGNOSTICS

Upon completion, **dd** reports the number of input and output records:

*f***+***p* **records in**	Number of full and partial blocks read.
*f***+***p* **records out**	Number of full and partial blocks written.

When **conv=block** is specified and there is at least one truncated block, the number of truncated records is also reported:

n **truncated records**

EXAMPLES

Read an EBCDIC tape blocked ten 80-byte EBCDIC card images per block into an ASCII file named **x**:

```
dd if=/dev/rmt/c0t0d0BEST of=x ibs=800 cbs=80 conv=ascii,lcase
```

Note the use of the raw magnetic tape device file. **dd** is especially suited to I/O on raw physical devices because it allows reading and writing in arbitrary block sizes.

WARNINGS

You may experience trouble writing directly to or reading directly from a cartridge tape. For best results, use *tcio*(1) as an input or output filter. For example, for output to a cartridge tape, use

```
... | dd ... | tcio -ovVS 256 /dev/rct/c4t1d0
```

or, for input from a cartridge tape, use

```
tcio -ivS 256 /dev/rct/c4t1d0 | dd ... | ...
```

Some devices, such as 1/2-inch magnetic tapes, are incapable of seeking. Such devices may be positioned prior to running **dd** by using *mt*(1) or some other appropriate command. The **skip**, **seek**, **iseek** and **oseek** options do work for such devices. However, skipping blocks using these options is slow on devices that cannot seek, since the blocks must actually be read to get to the desired position on the tape.

ASCII and EBCDIC conversion tables are taken from the 256-character ACM standard, Nov, 1968. The **ibm** conversion, while less widely accepted as a standard, corresponds better to certain IBM print train conventions. There is no universal solution.

Newline characters are inserted only on conversion to ASCII; padding is done only on conversion to EBCDIC. These should be separate options.

If **if** or **of** refers to a raw disk, **bs** should always be a multiple of the sector size of the disk. By default, **bs** is 512 bytes. If the sector size of the disk is different from 512 bytes, **bs** should be specified using a multiple of sector size. The character special (raw) device file should always be used for devices.

It is entirely up to the user to insure there is enough room in the destination file, file system and/or device to contain the output since **dd** cannot pre-determine the required space after conversion.

SEE ALSO

cp(1), mt(1), tr(1), disk(7), mt(7).

STANDARDS CONFORMANCE

dd: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

NAME

delta - make a delta (change) to an SCCS file

SYNOPSIS

delta [**-r** *SID*] [**-s**] [**-n**] [**-g** *list*] [**-m** *mrlist*] [**-y** *comment*] [**-p**] *files*

DESCRIPTION

The **delta** command is used to permanently introduce into the named SCCS file changes that were made to the file retrieved by **get** (called the *g-file*, or generated file). See *get*(1).

delta makes a delta to each named SCCS file. If a directory is named, **delta** behaves as though each file in the directory was specified as a named file, except that non-SCCS files (last component of the path name does not begin with **.s**) and unreadable files are silently ignored. If a name of **-** is given, the standard input is read (see WARNINGS). Each line of the standard input is taken to be the name of an SCCS file to be processed.

delta may issue prompts on the standard output, depending upon certain options specified and flags (see *admin*(1)) that may be present in the SCCS file (see the **-m** and **-y** options below).

Options

Option arguments apply independently to each named file.

-r*SID* Uniquely identifies which delta is to be made to the SCCS file. Use of this option is necessary only if two or more outstanding **get**s for editing (**get -e**) on the same SCCS file were done by the same person (login name). The *SID* value specified with the **-r** option can be either the *SID* specified on the **get** command line or the *SID* to be made as reported by the **get** command (see *get*(1)). A diagnostic results if the specified *SID* is ambiguous, or, if necessary and omitted on the command line.

-s Suppresses issuing, on the standard output, of the created delta's *SID* as well as the number of lines inserted, deleted and unchanged in the SCCS file.

-n Specifies retention of the edited *g-file* (normally removed at completion of delta processing).

-g*list* Specifies a *list* (see *get*(1) for the definition of *list*) of deltas which are to be *ignored* when the file is accessed at the change level (*SID*) created by this delta.

-m[*mrlist*] If the SCCS file has the **v** flag set (see *admin*(1)), a Modification Request (MR) number *must* be supplied as the reason for creating the new delta.

If **-m** is not used and the standard input is a terminal, the prompt **MRs?** is issued on the standard output before the standard input is read. If the standard input is not a terminal, no prompt is issued. The **MRs?** prompt always precedes the **comments?** prompt (see the **-y** option).

MRs in a list are separated by blanks and/or tab characters. An unescaped new-line character terminates the MR list.

Note that if the **v** flag has a value (see *admin*(1)), it is assumed to be the name of a program (or shell procedure) that is to validate the correctness of the MR numbers. If a non-zero exit status is returned from the MR number-validation program, **delta** assumes that the MR numbers were not all valid and terminates.

-y[*comment*] Arbitrary text used to describe the reason for making the delta. A null string is considered a valid *comment*.

If **-y** is not specified and the standard input is a terminal, the prompt **comments?** is issued on the standard output before the standard input is read. If the standard input is not a terminal, no prompt is issued. An unescaped new-line character terminates the comment text.

-p Causes **delta** to print (on the standard output in a *diff*(1) format) the SCCS file differences before and after the delta is applied.

EXTERNAL INFLUENCES

Environment Variables

LC_CTYPE determines the interpretation of text as single- and/or multi-byte characters.

LC_MESSAGES determines the language in which messages are displayed.

If **LC_CTYPE** or **LC_MESSAGES** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**. If any internationalization variable contains an invalid setting, **delta** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported.

DIAGNOSTICS

Use *sccshelp*(1) for explanations.

WARNINGS

SCCS files can be any length, but the number of lines in the text file itself cannot exceed 99 999 lines.

Lines beginning with an ASCII SOH character (octal 001) cannot be placed in the SCCS file unless the SOH is escaped. This character has special meaning to SCCS (see *sccsfile*(4)) and will cause an error.

A **get** of many SCCS files, followed by a **delta** of those files, should be avoided when the **get** generates a large amount of data. Instead, multiple **get**/**delta** sequences should be used.

If the standard input (**-**) is specified on the **delta** command line, the **-m** (if necessary) and **-y** options *must* also be present. Omission of these options causes an error.

Comments can be of multiple lines. The maximum length of the comment (total length of all comment lines) cannot exceed 1024 bytes. No line in a comment should have a length of more than 1000 bytes.

FILES

All of the auxiliary files listed below, except for the *g-file*, are created in the same directory as the *s-file* (see *get*(1)). The *g-file* is created in the user's working directory.

g-file	Existed before the execution of **delta**; removed after completion of **delta** (unless **-n** was specified).
p-file	Existed before the execution of **delta**; may exist after completion of **delta**.
q-file	Created during the execution of **delta**; removed after completion of **delta**.
x-file	Created during the execution of **delta**; renamed to SCCS file after completion of **delta**.
z-file	Created during the execution of **delta**; removed during the execution of **delta**.
d-file	Created during the execution of **delta**; removed after completion of **delta**.
/usr/bin/bdiff	Program to compute differences between the file retrieved by **get** and the *g-file*.

SEE ALSO

admin(1), bdiff(1), cdc(1), get(1), sccshelp(1), prs(1), rmdel(1), sccsfile(4).

STANDARDS CONFORMANCE

delta: SVID2, SVID3, XPG2, XPG3, XPG4

NAME

deroff - remove nroff, tbl, and neqn constructs

SYNOPSIS

deroff [**-m***x*] [**-w**] [**-i**] [*file* ...]

DESCRIPTION

deroff reads each *file* in sequence and removes all **nroff** requests, macro calls, backslash constructs, **neqn** constructs (between **.EQ** and **.EN** lines, and between delimiters — see *neqn*(1)), and **tbl** descriptions (see *tbl*(1)), replacing them with white space (blanks and blank lines), and writes the remainder of the file on the standard output. **deroff** follows chains of included files (**.so** and **.nx** **nroff**/**troff** formatter commands); if a file has already been included, a **.so** naming that file is ignored and a **.nx** naming that file terminates execution. If no input file is given, **deroff** reads the standard input.

The **-m** option can be followed by an **m**, **s**, or **l**. The **-mm** option causes the macros be interpreted such that only running text is output (that is, no text from macro lines). The **-ml** option forces the **-mm** option and also causes deletion of lists associated with the **mm** macros.

If the **-w** option is given, the output is a word list, one "word" per line, with all other characters deleted. Otherwise, the output follows the original, with the deletions mentioned above. In text, a "word" is any multi-byte character string or any string that contains at least two letters and is composed of letters, digits, ampersands (**&**), and apostrophes (**'**); In a macro call, however, a "word" is a multi-byte character string or a string that begins with at least two letters and contains a total of at least three letters. Delimiters are any characters other than letters, digits, apostrophes, and ampersands. Trailing apostrophes and ampersands are removed from "words."

If the **-i** option is specified, **deroff** ignores the **.so** and **.nx** **nroff**/**troff** commands.

EXTERNAL INFLUENCES

Environment Variables

LC_CTYPE determines the interpretation of text and filenames as single and/or multi-byte characters. Note that multi-byte punctuation characters are not recognized when using the **-w** option.

LC_MESSAGES determines the language in which messages are displayed.

If **LC_CTYPE** or **LC_MESSAGES** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**.

If any internationalization variable contains an invalid setting, **deroff** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported.

WARNINGS

deroff is not a complete **nroff** interpreter; thus it can be confused by subtle constructs. Most such errors result in too much rather than too little output.

The **-ml** option does not handle nested lists correctly.

AUTHOR

deroff was developed by the University of California, Berkeley.

SEE ALSO

neqn(1), nroff(1), tbl(1).

NAME

diff - differential file and directory comparator

SYNOPSIS

diff [**-C** *n*] [**-S** *name*] [**-lrs**] [**-bcefhintw**] *dir1 dir2*

diff [**-C** *n*] [**-S** *name*] [**-bcefhintw**] *file1 file2*

diff [**-D** *string*] [**-biw**] *file1 file2*

DESCRIPTION

Comparing Directories

If both arguments are directories, **diff** sorts the contents of the directories by name, then runs the regular file **diff** algorithm (described below) on text files that have the same name in each directory but are different. Binary files that differ, common subdirectories, and files that appear in only one directory are listed. When comparing directories, the following options are recognized:

-l Long output format; each text file **diff** is piped through **pr** to paginate it (see *pr*(1)). Other differences are remembered and summarized after all text file differences are reported.

-r Applies **diff** recursively to common subdirectories encountered.

-s **diff** reports files that are identical but otherwise not mentioned.

-S *name* Starts a directory **diff** in the middle of the sorted directory, beginning with file *name*.

Comparing Files

When run on regular files, and when comparing text files that differ during directory comparison, **diff** tells what lines must be changed in the files to bring them into agreement. **diff** usually finds a smallest sufficient set of file differences. However, it can be misled by lines containing very few characters or by other situations. If neither *file1* nor *file2* is a directory, either can be specified as **-**, in which case the standard input is used. If *file1* is a directory, a file in that directory whose filename is the same as the filename of *file2* is used (and vice versa).

There are several options for output format. The default output format contains lines resembling the following:

n1 **a** *n3***,***n4*
*n1***,***n2* **d** *n3*
*n1***,***n2* **c** *n3***,***n4*

These lines resemble **ed** commands to convert *file1* into *file2*. The numbers after the letters pertain to *file2*. In fact, by exchanging **a** for **d** and reading backwards one may ascertain equally how to convert *file2* into *file1*. As in **ed**, identical pairs where *n1* = *n2* or *n3* = *n4* are abbreviated as a single number.

Following each of these lines come all the lines that are affected in the first file flagged by **<**, then all the lines that are affected in the second file flagged by **>**.

Except for **-b**, **-w**, **-i**, or **-t** which can be given with any of the others, the following options are mutually exclusive:

-e Produce a script of **a**, **c**, and **d** commands for the **ed** editor suitable for recreating *file2* from *file1*. Extra commands are added to the output when comparing directories with **-e**, so that the result is a shell script for converting text files common to the two directories from their state in *dir1* to their state in *dir2* (see *sh-bourne*(1)

-f Produce a script similar to that of the **-e** option that is not useful with **ed** but is more readable by humans.

-n Produce a script similar to that of **-e**, but in the opposite order, and with a count of changed lines on each insert or delete command. This is the form used by **rcsdiff** (see *rcsdiff*(1)).

-c Produce a difference list with 3 lines of context. **-c** modifies the output format slightly: the output begins with identification of the files involved, followed by their creation dates, then each change separated by a line containing about twelve asterisks (*****)s. Lines removed from *file1* are marked with **-**, and lines added to *file2* are marked **+**. Lines that change from one file to the other are marked in both files with with **!**. Changes that lie within 3 lines of each other in the file are grouped together on output.

-C *n* Output format similar to **-c** but with *n* lines of context.

-h Do a fast, half-hearted job. This option works only when changed stretches are short and well separated, but can be used on files of unlimited length.

-D *string*
Create a merged version of *file1* and *file2* on the standard output, with C preprocessor controls included so that a compilation of the result without defining *string* is equivalent to compiling *file1*, while compiling the result with *string* defined is equivalent to compiling *file2*.

-b Ignore trailing blanks (spaces and tabs) and treat other strings of blanks as equal.

-w Ignore all whitespace (blanks and tabs). For example, **if (a == b)** and **if(a==b)** are treated as equal.

-i Ignores uppercase/lowercase differences. Thus **A** is treated the same as **a**.

-t Expand tabs in output lines. Normal or **-c** output adds one or more characters to the front of each line. Resulting misalignment of indentation in the original source lines can make the output listing difficult to interpret. This option preserves original source file indentation.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the locale to use for the locale categories when both **LC_ALL** and the corresponding environment variable (beginning with **LC_**) do not specify a locale. If **LANG** is not set or is set to the empty string, a default of "C" (see *lang*(5)) is used.

LC_CTYPE determines the space characters for the **diff** command, and the interpretation of text within file as single- and/or multi-byte characters.

LC_MESSAGES determines the language in which messages are displayed.

If any internationalization variable contains an invalid setting, **diff** and **diffh** behave as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported with the exception that **diff** and **diffh** do not recognize multi-byte alternative space characters.

RETURN VALUE

Upon completion, **diff** returns with one of the following exit values:

0	No differences were found.
1	Differences were found.
>1	An error occurred.

EXAMPLES

The following command creates a script file **script**:

```
diff -e x1 x2 >script
```

w is added to the end of the script in order to save the file:

```
echo w >> script
```

The script file can then be used to create the file **x2** from the file **x1** using the editor **ed** in the following manner:

```
ed x1 < script
```

The following command produces the difference output with 2 lines of context information before and after the line that was different:

```
diff -C2 x1 x2
```

The following command ignores all blanks and tabs and ignores uppercase-lowercase differences.

```
diff -wi x1 x2
```

WARNINGS

Editing scripts produced by the **-e** or **-f** option are naive about creating lines consisting of a single dot (**.**).

When comparing directories with the **-b**, **-w**, or **-i** options specified, **diff** first compares the files in the same manner as **cmp**, then runs the **diff** algorithm if they are not equal. This may cause a small amount of spurious output if the files are identical except for insignificant blank strings or uppercase/lowercase differences.

The default algorithm requires memory allocation of roughly six times the size of the file. If sufficient memory is not available for handling large files, the **-h** option or **bdiff** can be used (see *bdiff*(1)).

When run on directories with the **-r** option, **diff** recursively descends sub-trees. When comparing deep multi-level directories, more memory may be required than is currently available on the system. The amount of memory required depends on the depth of recursion and the size of the files.

AUTHOR

diff was developed by AT&T, the University of California, Berkeley, and HP.

FILES

/usr/lbin/diffh	used by **-h** option

SEE ALSO

bdiff(1), cmp(1), comm(1), diff3(1), diffmk(1), dircmp(1), ed(1), more(1), nroff(1), rcsdiff(1), sccsdiff(1), sdiff(1), terminfo(4).

STANDARDS CONFORMANCE

diff: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

NAME

diff3 - 3-way differential file comparison

SYNOPSIS

diff3 [**-exEX3**] *file1 file2 file3*

DESCRIPTION

diff3 compares three versions of a file, and prints disagreeing ranges of text flagged with these codes:

====	all three files differ
====1	*file1* is different
====2	*file2* is different
====3	*file3* is different

The type of change required to convert a given range of a given file to some other is indicated in one of these ways:

f:*n1***a**	Text is to be appended after line number *n1* in file *f*, where *f* = **1**, **2**, or **3**.
f:*n1*,*n2***c**	Text is to be changed in the range line *n1* through line *n2*. If *n1* = *n2*, the range can be abbreviated to *n1*.

The original contents of the range follows immediately after a **c** indication. When the contents of two files are identical, the contents of the lower-numbered file is suppressed.

-e	**diff3** Produces a script for the **ed** editor that can be used to incorporate into *file1* all changes between *file2* and *file3* (see *ed*(1)); i.e., the changes that normally would be flagged **====** and **====3**.
-x	Produces a script to incorporate only changes flagged **====**
-3	Produces a script to incorporate only changes flagged **====3**
-E	Produces a script that will incorporate all changes between *file2* and *file3*, but treat overlapping changes (that is, changes that would be flagged with **====** in normal listing) differently. The overlapping lines in both files will be inserted by the edit script bracketed by **<<<<<<** and **>>>>>>** lines.
-X	Produces a script that will incorporate only changes flagged **====** , but treat these changes in the manner of **-E** option.

The following command applies the resulting script to *file1*.

```
(cat script; echo '1,$p') | ed - file1
```

EXTERNAL INFLUENCES

International Code Set Support

Single- and multi-byte character code sets are supported.

WARNINGS

Text lines that consist of a single period (**.**) defeat **-e**.

Files longer than 64K bytes do not work.

FILES

/var/tmp/d3*
/usr/lbin/diff3prog

SEE ALSO

diff(1).

NAME

diffmk - mark changes between two different versions of a file

SYNOPSIS

diffmk *prevfile currfile markfile*

DESCRIPTION

diffmk compares the previous version of a file with the current version and creates a file that includes **nroff**/**troff** "change mark" commands. *prevfile* is the name of the previous version of the file and *currfile* is the name of the current version of the file. **diffmk** generates *markfile* which contains all the lines of the *currfile* plus inserted formatter "change mark" (**.mc**) requests. When *markfile* is formatted, changed or inserted text is shown by a **|** character at the right margin of each line. The position of deleted text is shown by a single *****.

If the characters **|** and ***** are inappropriate, a copy of **diffmk** can be edited to change them because **diffmk** is a shell script.

EXTERNAL INFLUENCES

International Code Set Support

Single- and multi-byte character code sets are supported.

EXAMPLES

A typical command line for comparing two versions of an **nroff**/**troff** file and generating a file with the changes marked is:

```
diffmk prevfile currfile markfile; nroff markfile | pr
```

diffmk can also be used to produce listings of C (or other) programs with changes marked. A typical command line for such use is:

```
diffmk prevfile.c currfile.c markfile.c; nroff macs markfile.c | pr
```

where the file **macs** contains:

```
.pl 1
.ll 77
.nf
.eo
```

The **.ll** request can specify a different line length, depending on the nature of the program being printed. The **.eo** request is probably needed only for C programs.

WARNINGS

Aesthetic considerations may dictate manual adjustment of some output.

diffmk does not differentiate between changes in text and changes in formatter request coding. Thus, file differences involving only formatting changes (such as replacing **.sp** with **.sp 2** in a text source file) with no change in actual text can produce change marks.

Although unlikely, certain combinations of formatting requests can cause change marks to either disappear or to mark too much. Manual intervention may be required because the subtleties of various formatting macro packages and preprocessors is beyond the scope of **diffmk**. **tbl** cannot tolerate **.mc** commands in its input (see *tbl*(1)), so any **.mc** request that would appear inside a **.TS** range is silently deleted. The script can be changed if this action is inappropriate, or **diffmk** can be run on two files that have both been run through the **tbl** preprocessor before any comparisons are made.

diffmk uses **diff**, and thus has the same limitations on file size and performance that **diff** may impose (see *diff*(1)). In particular the performance is nonlinear with the size of the file, and very large files (well over 1000 lines) may take extremely long to process. Breaking the file into smaller pieces may be advisable.

diffmk also uses the *ed*(1) editor. If the file is too large for **ed**, **ed** error messages may be embedded in the file. Again, breaking the file into smaller pieces may be advisable.

SEE ALSO

diff(1), nroff(1).

NAME

dircmp - directory comparison

SYNOPSIS

dircmp [**-d**] [**-s**] [**-w***n*] *dir1 dir2*

DESCRIPTION

dircmp examines *dir1* and *dir2* and generates various tabulated information about the contents of the directories. Sorted listings of files that are unique to each directory are generated for all the options. If no option is entered, a sorted list is output indicating whether the filenames common to both directories have the same contents.

-d Compare the contents of files with the same name in both directories and output a list telling what must be changed in the two files to bring them into agreement. The list format is described in *diff*(1).

-s Suppress messages about identical files.

-w*n* Change the width of the output line to *n* characters. The default width is 72.

EXTERNAL INFLUENCES

Environment Variables

LC_COLLATE determines the order in which the output is sorted.

If **LC_COLLATE** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**. If any internationalization variable contains an invalid setting, **dircmp** behaves as if all internationalization variables are set to "C" (see *environ*(5)).

International Code Set Support

Single- and multi-byte character code sets are supported.

EXAMPLES

Compare the two directories **slate** and **sleet** and produce a list of changes that would make the directories identical:

```
dircmp -d slate sleet
```

WARNINGS

This command is likely to be withdrawn from X/Open standards. Applications using this command might not be portable to other vendors' systems. As an alternative **diff -R** is recommended.

SEE ALSO

cmp(1), diff(1).

STANDARDS CONFORMANCE

dircmp: SVID2, SVID3, XPG2, XPG3

NAME

dmpxlt - dump iconv translation tables to a readable format

SYNOPSIS

/usr/bin/dmpxlt [**-f** *output_filename*] [*input_filename*]

DESCRIPTION

dmpxlt dumps the compiled version of the **iconv** codeset conversion tables into an ASCII-readable format that can be modified and used as input to *genxlt*(1) to regenerate the table for *iconv*(1).

Options

dmpxlt recognizes the following options:

-f *output_filename*	If this option is not selected, the data will be sent to standard output.

dmpxlt will create an output file in the prescribed format, giving the filecode mapping between the two code sets, which can be edited and reused by *genxlt*(1) to create new tables for *iconv*(1). The entries are in hexadecimal.

EXTERNAL INFLUENCES

Environment Variables

LANG provides a default value for the internationalization variables that are unset or null. If **LANG** is unset or null, the default value of "C" (see *lang*(5)) is used. If any of the internationalization variables contains an invalid setting, **dmpxlt** will behave as if all internationalization variables are set to "C". See *environ*(5).

LC_ALL If set to a non-empty string value, overrides the values of all the other internationalization variables.

LC_MESSAGES determines the locale that should be used to affect the format and contents of diagnostic messages written to standard error and informative messages written to standard output.

NLSPATH determines the location of message catalogues for the processing of **LC_MESSAGES.**

International Code Set Support

Single and multi-byte character code sets are supported.

RETURN VALUE

The following are exit values:

0	Successful completion.
>0	Error condition occurred.

EXAMPLES

This example creates the source file *genxlt_input* from the table **roma8=iso81**:

```
dmpxlt -f genxlt_input /usr/lib/nls/iconv/tables/roma8=iso81
```

FILES

/usr/lib/nls/iconv/tables	All tables must be installed in this directory.

SEE ALSO

iconv(1), genxlt(1), iconv(3C), environ(5) lang(5).

NAME

domainname - set or display name of Network Information Service domain

SYNOPSIS

domainname [*name_of_domain*]

DESCRIPTION

Network Information Service (NIS) uses domain names to refer collectively to a group of hosts. Without an argument, *domainname* displays the name of the NIS domain. Only superuser can set the domain name by providing *name_of_domain*. The domain name is usually set in the configuration file **/etc/rc.config.d/namesvrs**, by setting the **NIS_DOMAIN** variable.

DEPENDENCIES

NIS servers use the NIS domain name as the name of a subdirectory of **/var/yp**. Since the NIS domain name can be as long as 64 characters, *name_of_domain* may exceed the maximum file name length allowed on a given file system. If that length is exceeded, the subdirectory name becomes a truncated version of the NIS domain name.

The first 14 characters of all NIS domains on the network must be unique: truncated names should be checked to verify that they meet this requirement.

AUTHOR

domainname was developed by Sun Microsystems, Inc.

SEE ALSO

ypinit(1M), getdomainname(2), setdomainname(2).

NAME

dos2ux, ux2dos - convert ASCII file format

SYNOPSIS

dos2ux *file*...

ux2dos *file*...

DESCRIPTION

dos2ux and **ux2dos** read each specified *file* in sequence and write it to standard output, converting to HP-UX format or to DOS format, respectively. Each *file* can be either DOS format or HP-UX format for either command.

A DOS file name is recognized by the presence of an embedded colon (**:**) delimiter; see *dosif*(4) for DOS file naming conventions.

If no input file is given or if the argument **-** is encountered, **dos2ux** and **ux2dos** read from standard input. Standard input can be combined with other files.

EXAMPLES

Print file **myfile** on the display:

```
dos2ux myfile
```

Convert **file1** and **file2** to DOS format then concatenate them together, placing them in **file3**.

```
ux2dos file1 file2 > file3
```

RETURN VALUE

dos2ux and **ux2dos** return **0** if successful or **2** if the command failed. The only possible failure is the inability to open a specified file, in which case the commands print a warning.

WARNINGS

Command formats resembling:

```
dos2ux file1 file2 > file1
```

overwrite the data in **file1** before the concatenation begins, causing a loss of the contents of **file1**. Therefore, be careful when using shell special characters.

SEE ALSO

doschmod(1), doscp(1), dosdf(1), dosls(1), dosmkdir(1), dosrm(1), dosif(4).

NAME

doschmod - change attributes of a DOS file

SYNOPSIS

doschmod [**-u**] *mode device*:*file* ...

DESCRIPTION

doschmod is the DOS counterpart of **chmod** (see *chmod*(1)).

Options

doschmod recognizes one option:

-u Disable argument case conversion. In the absence of this option, all DOS file names are converted to uppercase.

A DOS file name is recognized by the presence of an embedded colon (**:**) delimiter; see *dosif*(4) for DOS file naming conventions.

Metacharacters *****, **?**, and **[** ... **]** can be used when specifying DOS file names. These must be quoted when specifying a DOS file name, because file name expansion must be performed by the DOS utilities, not by the shell. DOS utilities expand file names as described in *regexp*(5) under *PATTERN MATCHING NOTATION*.

The attributes of each named file are changed according to *mode*, which is an octal number in the range 000 to 0377. *mode* is constructed from the logical OR of the following modes:

200	Reserved. Do not use.
100	Reserved. Do not use.
040	Archive. Set whenever the file has been written to and closed.
020	Directory. Do not modify.
010	Volume Label. Do not modify.
004	System file. Marks files that are part of the DOS operating system.
002	Hidden file. Marks files that do not appear in a DOS directory listing using the DOS DIR command.
001	Read-Only file. Marks files as read-only.

WARNINGS

Specifying inappropriate *mode* values can make files and/or directories inaccessible, and in certain cases can damage the file system. To prevent such problems, do not change the mode of directories and volume labels.

Normal users should have no need to use *mode* bits other than 001, 002, and 040.

EXAMPLES

Mark file **/dev/rfd9122:memo.txt** as a hidden file:

```
doschmod 002 /dev/rfd9122:memo.txt
```

Mark file **driveC:autoexec.bat** read-only:

```
doschmod 001 driveC:autoexec.bat
```

SEE ALSO

chmod(1), dos2ux(1), doscp(1), dosdf(1), dosls(1), dosmkdir(1), dosrm(1), chmod(2), dosif(4).

NAME

doscp - copy to or from DOS files

SYNOPSIS

doscp [**-fvu**] *file1 file2*

doscp [**-fvu**] *file1* [*file2* ...] *directory*

DESCRIPTION

doscp is the DOS counterpart of **cp** (see *cp*(1)). **doscp** copies a DOS file to a DOS or HP-UX file, an HP-UX file to an HP-UX or DOS file, or HP-UX or DOS files to an HP-UX or DOS directory. The last name in the argument list is the destination file or directory.

A DOS file name is recognized by the presence of an embedded colon (**:**) delimiter; see *dosif*(4) for DOS file naming conventions.

Metacharacters *****, **?**, and **[** ... **]** can be used when specifying both HP-UX and DOS file names. These must be quoted when specifying a DOS file name, because file name expansion must be performed by the DOS utilities, not by the shell. DOS utilities expand file names as described in *regexp*(5) under *PATTERN MATCHING NOTATION*.

The file name **-** (dash) is interpreted to mean standard input or standard output depending upon its position in the argument list.

Options

doscp recognizes the following options:

-f Unconditionally write over an existing file. In the absence of this option, **doscp** asks permission to overwrite an existing HP-UX file.

-v Verbose mode. **doscp** prints the source name.

-u Disable argument case conversion. In the absence of this option, all DOS file names are converted to upper case.

RETURN VALUE

doscp returns 0 if all files are copied successfully. Otherwise, it prints a message to standard error and returns with a non-zero value.

EXAMPLES

Copy the files in the HP-UX directory **abc** to the DOS volume stored as HP-UX file **hard_disk**:

```
doscp abc/* hard_disk:
```

Copy DOS file **/backup/log** through the HP-UX special file **/dev/rfd9127** to HP-UX file **logcopy** located in the current directory:

```
doscp /dev/rfd9127:/backup/log  logcopy
```

Copy DOS file **zulu** on the volume stored as HP-UX file **bb** to standard output:

```
doscp bb:zulu -
```

Copy all files in directory **/dameron** with extension **txt** in the DOS volume **/dev/rdsk/c1t2d0** to the HP-UX directory **abacus** located in the current directory:

```
doscp '/dev/rdsk/c1t2d0:/dameron/*.txt' abacus
```

WARNINGS

doscp works more reliably if you use a raw device special file (**/dev/rdsk/**) than a block device special file.

To use SCSI floppy disk devices, the **sflop** device driver must be configured into the kernel. (You can use the **ioscan** command to verify the configuration.)

SEE ALSO

cp(1), dos2ux(1), doschmod(1), dosdf(1), dosls(1), dosmkdir(1), dosrm(1), ioscan(1M) dosif(4).

NAME

dosdf - report number of free disk clusters

SYNOPSIS

dosdf *device*[**:**]

DESCRIPTION

dosdf is the DOS counterpart of the **df** command (see *df*(1)). It prints the cluster size in bytes and the number of free clusters on the specified DOS volume.

SEE ALSO

df(1), dos2ux(1), doschmod(1), doscp(1), dosls(1), dosmkdir(1), dosrm(1), dosif(4).

NAME

dosls, dosll - list contents of DOS directories

SYNOPSIS

dosls [**-aAudl**] *device*:[*file*] ...

dosll [**-aAudl**] *device*:[*file*] ...

DESCRIPTION

dosls is the DOS counterpart of **ls** (see *ls*(1)).

For each directory named, **dosls** lists the contents of that directory. For each file named, **dosls** repeats its name and any other information requested. If invoked by the name **dosll**, the **-l** (ell) option is implied.

Options

dosls and **dosll** recognizes the following options:

- **-a** List all directory entries. In the absence of this option, hidden files, system files, and files whose names begin with a dot (**.**) are not listed.
- **-A** Same as **-a**, except the current directory and the parent directory are not listed. For the superuser, this option defaults to being set, and is disabled by **-A**.
- **-u** Disable argument case conversion. In the absence of this option, all DOS file names are converted to uppercase.
- **-d** If an argument is a directory, list only its name. Often used with **-l** to get the status of a directory.
- **-l** List in long format, giving file attribute, size in bytes, and the date and time of last modification for each file, as well as listing the DOS volume label. Long listing is disabled if this option is used with the **dosll** command.

A DOS file name is recognized by the presence of an embedded colon (**:**) delimiter; see *dosif*(4) for DOS file naming conventions.

Metacharacters *****, **?**, and **[** ... **]** can be used when specifying DOS file names. These must be quoted when specifying a DOS file name, because file name expansion must be performed by the DOS utilities, not by the shell. DOS utilities expand file names as described in *regexp*(5) under *PATTERN MATCHING NOTATION*.

EXAMPLES

These examples assume that a DOS directory structure exists on the device accessed through HP-UX special file **/dev/rdsk/c2t1d0**.

The following example lists all of the files in the root directory of the DOS directory structure:

```
dosls -a /dev/rdsk/c2t1d0:
```

The following example lists all of the files with extension **bat** in the root directory of the DOS directory structure:

```
dosls -a '/dev/rdsk/c2t1d0:*.bat'
```

The following example produces a long-format listing of all the information about the DOS directory **/dos/math**, but does not list the files in the directory:

```
dosls -ld /dev/rdsk/c2t1d0:/dos/math
```

SEE ALSO

dos2ux(1), doschmod(1), doscp(1), dosdf(1), dosmkdir(1), dosrm(1), ls(1), dosif(4).

d

NAME

dosmkdir - make a DOS directory

SYNOPSIS

dosmkdir [**-u**] *device*:*directory* ...

DESCRIPTION

dosmkdir is the DOS counterpart of the **mkdir** command (see *mkdir*(1)). It creates specified directories. The standard entries, **.** for the directory itself and **..** for its parent, are made automatically.

There is one option:

-u Disable argument case conversion. In the absence of this option, all DOS file names are converted to uppercase.

A DOS file name is recognized by the presence of an embedded colon (**:**) delimiter; see *dosif*(4) for DOS file naming conventions.

DIAGNOSTICS

dosmkdir returns 0 if all directories were successfully created. Otherwise, it prints a message to standard error and returns non-zero.

EXAMPLES

Create an empty subdirectory named **numbers** under the directory **/math/lib** on the device accessed through HP-UX special file **/dev/rfd9122**:

```
dosmkdir /dev/rfd9122:/math/lib/numbers
```

SEE ALSO

dos2ux(1), doschmod(1), doscp(1), dosdf(1), dosls(1), dosrm(1), mkdir(1), dosif(4).

NAME

dosrm, dosrmdir - remove DOS files or directories

SYNOPSIS

dosrm [**-friu**] *device*:*file* ...

dosrmdir [**-u**] *device*:*file* ...

DESCRIPTION

dosrm and **dosrmdir** are DOS counterparts of **rm** and **rmdir** (see *rm*(1) and *rmdir*(1), respectively).

dosrm removes the entries for one or more files from a directory. If a specified file is a directory, an error message is printed unless the optional argument **-r** is specified (see below).

dosrmdir removes entries for the named directories, provided they are empty.

Options

dosrm and **dosrmdir** recognize the following options:

-f (force) Unconditionally remove the specified file, even if the file is marked read-only.

-r Cause **dosrm** to recursively delete the entire contents of a directory, followed by the directory itself. **dosrm** can recursively delete up to 17 levels of directories.

-i (interactive) Cause **dosrm** to ask whether or not to delete each file. If **-r** is also specified, **dosrm** asks whether to examine each directory encountered.

-u Disable argument case conversion. In the absence of this option, all DOS file names are converted to uppercase.

A DOS file name is recognized by the presence of an embedded colon (**:**) delimiter; see *dosif*(4) for DOS file naming conventions.

Metacharacters *****, **?**, and **[...]** can be used when specifying DOS file names. These must be quoted when specifying a DOS file name, because file name expansion must be performed by the DOS utilities, not by the shell. DOS utilities expand file names as described in *regexp*(5) under *PATTERN MATCHING NOTATION*.

EXAMPLES

These examples assume that a DOS directory structure exists on the device accessed through the HP-UX special file **/dev/rfd9122**.

Recursively comb through the DOS directory **/tmp** and ask if each DOS file should be removed forcibly (that is, with no file mode checks):

```
dosrm -irf /dev/rfd9122:/tmp
```

Remove the DOS directory **doug** from the DOS volume stored as HP-UX file **hard_disk**:

```
dosrmdir hard_disk:doug
```

SEE ALSO

dos2ux(1), doschmod(1), doscp(1), dosdf(1), dosls(1), dosmkdir(1), rm(1), rmdir(1), dosif(4).

NAME

du - summarize disk usage

SYNOPSIS

du [**-a**|**-s**] [**-bkrx**] [**-t** *type*] [*name* ...]

DESCRIPTION

The **du** command gives the number of 512-byte blocks allocated for all files and (recursively) directories within each directory and file specified by the *name* operands. The block count includes the indirect blocks of the file. A file with two or more links is counted only once. If *name* is missing, the current working directory is used.

By default, **du** generates an entry only for the *name* operands and each directory contained within those hierarchies.

Options

The **du** command recognizes the following options:

-a	Print entries for each file encountered in the directory hierarchies in addition to the normal output.
-b	For each *name* operand that is a directory for which file system swap has been enabled, print the number of blocks the swap system is currently using.
-k	Gives the block count in 1024-byte blocks.
-r	Print messages about directories that cannot be read, files that cannot be accessed, etc. **du** is normally silent about such conditions.
-s	Print only the grand total of disk usage for each of the specified *name* operands.
-x	Restrict reporting to only those files that have the same device as the file specified by the *name* operand. Disk usage is normally reported for the entire directory hierarchy below each of the given *name* operands.
-t *type*	Restrict reporting to file systems of the specified *type*. (Example values for *type* are **hfs**, **cdfs**, **nfs,** etc.) Multiple **-t** *type* options can be specified. Disk usage is normally reported for the entire directory hierarchy below each of the given *name* operands.

EXAMPLES

Display disk usage for the current working directory and all directories below it, generating error messages for unreadable directories:

```
du -r
```

Display disk usage for the entire file system except for any **cdfs** or **nfs** mounted file systems:

```
du -t hfs /
```

Display disk usage for files on the root volume (**/**) only. No usage statistics are collected for any other mounted file systems:

```
du -x /
```

WARNINGS

Block counts are incorrect for files that contain holes.

SEE ALSO

df(1M), bdf(1M), quot(1M).

STANDARDS CONFORMANCE

du: SVID2, SVID3, XPG2, XPG3, XPG4

NAME

echo - echo (print) arguments

SYNOPSIS

echo [*arg*] ...

DESCRIPTION

echo writes its arguments separated by blanks and terminated by a new-line on the standard output. It also understands C-like escape conventions; beware of conflicts with the shell's use of \\:

\a	write an alert character
\b	backspace
\c	print line without appending a new-line
\f	form-feed
\n	new-line
\r	carriage return
\t	tab
\v	vertical tab
\\	backslash
\*n*	the 8-bit character whose ASCII code is the 1-, 2-, 3- or 4-digit octal number *n*, whose first character must be a zero.
\0*num*	write an 8-bit value that is the zero-, one-, two- or three-digit octal number *num*

echo is useful for producing diagnostics in command files and for sending known data into a pipe.

Notes

Berkeley **echo** differs from this implementation. The former does not implement the backslash escapes. However, the semantics of the **\c** escape can be obtained by using the **-n** option. The echo command implemented as a built-in function of **csh** follows the Berkeley semantics (see *csh*(1)).

EXTERNAL INFLUENCES

Environment Variables

LC_CTYPE determines the interpretation of *arg* as single and/or multi-byte characters.

If **LC_CTYPE** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**. If any internationalization variable contains an invalid setting, **echo** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported.

AUTHOR

echo was developed by OSF and HP.

SEE ALSO

sh(1).

BUGS

No characters are printed after the first **\c**. This is not normally a problem.

STANDARDS CONFORMANCE

echo: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

NAME

ed, red - line-oriented text editor

SYNOPSIS

ed [**-p** *string*] [**-s**|**-**] [**-x**] [*file*]

red [**-p** *string*] [**-s**|**-**] [**-x**] [*file*]

DESCRIPTION

The **ed** command executes a line-oriented text editor. It is most commonly used in scripts and noninteractive editing applications because, even though it can be used interactively, other editors such as **vi** and **ex** are typically easier to use in an interactive environment.

If *file* is specified, **ed** performs an **e** command (see below) on the named file; that is to say, the file is read into **ed**'s buffer so that it can be edited.

Options

The following options are recognized:

-p *string* Use *string* as the prompt string when in command mode. By default, there is no prompt string.

-s|**-** Suppress printing of byte counts by **e**, **E**, **r**, and **w** commands, and suppress the **!** prompt after a **!** *command*. The **-** option is obsolescent and will be removed in a future release.

-x Perform an **X** command first to handle an encrypted file.

File Handling

ed operates on a copy of the file it is editing; changes made to the copy have no effect on the original file until a **w** (write) command is given. The copy of the text being edited resides in a temporary file called the *buffer*. There is only one buffer.

red is a restricted version of **ed** that only allows editing of files in the current directory and prohibits executing shell commands via **!***shell-command*. Attempts to bypass these restrictions result in the error message **restricted shell**.

Both **ed** and **red** support the *fspec*(4) formatting capability. After including a format specification as the first line of *file* and invoking **ed** with the controlling terminal in **stty -tabs** or **stty tab3** mode (see *stty*(1)), the specified tab stops are automatically used when scanning *file*. For example, if the first line of a file contained

```
<:t5,10,15 s72:>
```

the tab stops would be set at columns 5, 10, and 15, and a maximum line length of 72 would be imposed.

Note: When you input text, **ed** expands tab characters as they are typed to every eighth column as a default.

Editor Commands Structure

Commands to **ed** have a simple and regular structure: zero, one, or two *addresses* followed by a single-character *command*, possibly followed by parameters to that command. These addresses specify one or more lines in the buffer. Every command that requires addresses has default addresses, so that the addresses can very often be omitted.

In general, only one command is allowed on a line. Append, change, and insert commands accept text input which is then placed in the buffer as appropriate. While **ed** is accepting text following an append, change, or insert command, it is said to be in *input mode*. While in input mode, *no* editor commands are recognized; all input is merely collected. To terminate input mode, type a period (**.**) alone at the beginning of a line.

Regular Expressions

ed supports the Basic Regular Expression (RE) syntax (see *regexp*(5)), with the following additions:

- The null RE (for example, **//**) is equivalent to the last RE encountered.
- If the closing delimiter of an RE or of a replacement string (for example, **/**) would be the last character before a newline, that delimiter can be omitted, in which case the addressed line is printed. The following pairs of commands are equivalent:

```
s/s1/s2          g/s1          ?s1
s/s1/s2/p        g/s1/p        ?s1?
```

Line Addresses

To understand line addressing, remember that **ed** maintains a pointer to the *current line*. Generally speaking, the current line is the last line affected by a command. The exact effect of a given command on the current line is discussed under the description of each command. Addresses are interpreted according to the following rules:

1. The character **.** refers to the current line.
2. The character **$** refers to the last line of the buffer.
3. A decimal number *n* refers to the *n*th line of the buffer.
4. A **'***x* refers to the line marked with the mark name character *x*, which must be a lower-case letter. Lines are marked with the **k** command described below.
5. An RE enclosed by slashes (**/***RE***/**) refers to the first line found by searching forward from the line following the current line toward the end of the buffer and stopping at the first line containing a string matching the RE. If necessary, the search wraps around to the beginning of the buffer and continues up to and including the current line, so that the entire buffer is searched. (Also see WARNINGS below.)
6. An RE enclosed by question marks (**?***RE***?**) addresses the first line found by searching backward from the line preceding the current line toward the beginning of the buffer and stopping at the first line containing a string matching the RE. If necessary, the search wraps around to the end of the buffer and continues up to and including the current line. (Also see WARNINGS below.)
7. An address followed by a plus (**+**) or minus (**-**) sign followed by a decimal number specifies that address plus or minus the indicated number of lines. The plus sign can be omitted.
8. If an address begins with **+** or **-**, the addition or subtraction is calculated with respect to the current line. For example, **-5** is interpreted as **.-5**.
9. If an address ends with **+** or **-**, 1 is added to or subtracted from the address, respectively. As a consequence of this and rule 8 above, the address **-** refers to the line preceding the current line. (To maintain compatibility with earlier versions of the editor, the circumflex (**^**) and **-** characters are interpreted identically when encountered in addresses.) Moreover, multiple trailing **+** and **-** characters have a cumulative effect, so **--** refers to the second line preceding the current line.
10. For convenience, a comma (**,**) represents the address pair **1,$**, while a semicolon (**;**) represents the pair **.,$**.

Commands require zero, one, or two addresses. Commands that do not use addresses treat the presence of an address as an error. Commands that accept one or two addresses assume default addresses when the number of addresses specified is insufficient. If more addresses are specified than a given command requires, the last one or two are used as appropriate.

Addresses are usually separated from each other by a comma (**,**). They can also be separated by a semicolon (**;**), in which case the current line (**.**) is set to the first address, after which the second address is calculated. This feature can be used to determine the starting line for forward and backward searches (see rules 5 and 6 above). The second address of any two-address sequence must correspond to a line in the buffer that follows the line corresponding to the first address.

Editor Commands

In the following list of **ed** commands, the default addresses are shown in parentheses (parentheses are not part of the address and should not be placed in an actual command except for other purposes).

It is generally illegal for more than one command to appear on a line. However, any command (except **e**, **f**, **r**, or **w**) can be suffixed by **l**, **n**, or **p** in which case the current line is respectively either listed, numbered, or printed, as discussed below under the **l**, **n**, and **p** commands.

(**.**)**a** *text* **.**	The **a** (append) command reads *text* and appends it after the addressed line. Upon completion, the new current line is the last inserted line, or, if no text was added, at the addressed line. Address 0 is legal for this command, causing the appended *text* to be placed at the beginning of the buffer.

(**.,.**)**c**
text
.

The **c** (change) command deletes the addressed lines then accepts input text to replace the deleted lines. Upon completion, the new current line is the last line in *text* or, if no text was provided, at the first line after the deleted line or lines.

(**.,.**)**d**

The **d** (delete) command deletes the addressed lines from the buffer. Upon completion, the new current line is the first line following the deleted text, or the last line in the file if the deleted line or lines were at the end of the buffer.

e *file*

The **e** (edit) command deletes the entire contents of the buffer, then reads in the named *file*. Upon completion, the new current line is the last line in the buffer. If no file name is given, the remembered file name, if any, is used (see the **f** command). The number of characters read is displayed, and *file* is remembered for possible use as a default file name in subsequent **e**, **r**, or **w** commands.

If the *file* name starts with **!**, the rest of the line is interpreted as a shell command whose standard output is to be read. Such a shell command is not remembered as the current file name.

Also see DIAGNOSTICS below.

E *file*

The **E** (forced edit) command is identical to **e** except that no check is made to ensure that the current buffer has not been altered since the last **w** command.

f *file*

If *file* is specified, the **f** (file name) command changes the remembered file name to *file*. Otherwise, it prints the remembered file name.

(**1,$**)**g/***RE***/***command-list*

The **g** (global) command first marks every line that matches the given RE. Then, for every such line, the given *command-list* is executed with the current line initially set to that line. A single command or the first of a list of commands appears on the same line as the global command. All lines of a multiple-line list except the last line must end with a backslash (\). **a**, **i**, and **c** commands and associated input are permitted. The **.** that normally terminates input mode can be omitted if it would be the last line of the *command-list*. An empty *command-list* is equivalent to the **p** command. The **g**, **G**, **v**, and **V** commands are not permitted in the *command-list*. (Also see WARNINGS below.)

(**1,$**)**G/***RE***/**

The interactive **G** (Global) command first marks every line that matches the given RE. Then, for every such line, the line is printed, then the current line is changed to that line and one command (other than **a**, **c**, **i**, **g**, **G**, **v**, or **V**) can be input and executed. After executing that command, the next marked line is printed, and so on. A newline character acts as a null command, and an **&** causes the re-execution of the most recent command executed within the current invocation of **G**. Note that the commands input as part of the execution of the **G** command may address and affect *any* lines in the buffer. The **G** command can be terminated by an interrupt signal (ASCII DEL or BREAK).

h

The **h** (help) command gives a short error message explaining the reason for the most recent **?** diagnostic.

H

The **H** (Help) command causes **ed** to enter a mode in which error messages are printed for all subsequent **?** diagnostics. It also explains the previous **?** if there was one. The **H** command alternately turns this mode on and off. Initially, it is off.

(**.**)**i**
text
.

The **i** (insert) command inserts the given *text* before the addressed line. Upon completion, the current line is the last inserted line, or, if there were none, the addressed line. This command differs from the **a** command only in the placement of the input text. Address 0 is not legal for this command.

(**.,.+1**)**j**

The **j** (join) command joins contiguous lines by removing the appropriate newline characters. If exactly one address is given, this command does nothing.

(**.**)**k***x*

The **k** (mark) command marks the addressed line with the name *x*, which must be a lowercase letter. The address **'***x* then addresses this line. Upon completion, the new current line remains unchanged from before.

(**.,.**)**l**

The **l** (list) command writes the addressed lines to standard output in a visually unambiguous form. Characters listed in the following table are written as the corresponding escape sequence. Nonprintable characters not in the table are written as a three-digit octal number (with a preceding backslash character) for each byte in the character (most significant byte first).

Long lines are folded with the point of folding indicated by writing a backslash character followed by a newline. The end of each line is marked with a **$**. An **l** (ell) command can be appended to any command other than **e**, **E**, **f**, **q**, **Q**, **r**, **w**, or **!**. The current line number is set to the address of the last line written.

Escape Sequence	**Represents**	**ASCII Name**	**Escape Sequence**	**Represents**	**ASCII Name**
\	backslash	\	**\r**	carriage return	CR
\a	alert	BEL	**\t**	horizontal tab	HT
\b	backspace	BS	**\v**	vertical tab	VT
\f	formfeed	FF			

(**.**,**.**)**m***a* The **m** (move) command repositions the addressed lines after the line addressed by *a*. Address 0 is legal for *a*, causing the addressed lines to be moved to the beginning of the file. It is an error if address *a* falls within the range of moved lines; Upon completion, the new current line is the last line moved.

(**.**,**.**)**n** The **n** (number) command prints the addressed lines, preceding each line by its line number and a tab character. Upon completion, the new current line is the last line printed. The **n** command can be appended to any command other than **e**, **f**, **r**, or **w**.

(**.**,**.**)**p** The **p** (print) command prints the addressed lines. Upon completion, the new current line is the last line printed. The **p** command may be appended to any other command other than **e**, **E**, **f**, **q**, **Q**, **r**, **w**, or **!**. For example, **dp** deletes the current line and prints the new current line.

P The **P** (prompt) command causes **ed** to prompt with an asterisk (*****) (or with *string* if the **-p** option was specified in the command line) for all subsequent commands. The **P** command alternately turns this mode on and off. It is initially on if the **-p** option was specified; otherwise, off. The current line number is unchanged.

q The **q** (quit) command causes **ed** to exit. No automatic write of a file is done (but see DIAGNOSTICS below).

Q The editor exits unconditionally without checking for changes in the buffer since the last **w** command.

(**$**)**r** *file* The **r** (read) command reads the specified *file* into the buffer after the addressed line. If no file name is given, the remembered file name, if any, is used (see the **e** and **f** commands). The remembered file name is not changed unless *file* is the very first file name mentioned since **ed** was invoked. Address 0 is legal for **r** and places the contents of *file* at the beginning of the buffer. If the read is successful, the number of characters read is displayed. Upon completion, the new current line is the last line read into the buffer. If the *file* name starts with **!**, the rest of the line is interpreted as a shell command whose standard output is to be read. For example, **$r !ls** appends a listing of files in the current directory to the end of the file being edited. A shell command is not remembered as the current file name.

(**.**,**.**)**s/***RE***/***replacement***/***flags*

The **s** (substitute) command searches each addressed line for an occurrence of the specified RE. In each line in which a match is found, all (nonoverlapped) matched strings are replaced by *replacement* if the global replacement indicator **g** appears after the command. If the global indicator does not appear, only the first occurrence of the matched string is replaced. If a number *n* appears after the command, only the *n*th occurrence of the matched string on each addressed line is replaced. It is an error for the substitution to fail on all addressed lines. Any character other than space or newline can be used instead of **/** to delimit the RE and *replacement*. Upon completion, the new current line is the last line on which a substitution occurred. (Also see WARNINGS below.)

If an ampersand (**&**) appears in *replacement*, it is replaced by the string matching the RE on the current line. The special meaning of **&** in this context can be suppressed by preceding it with ****.

As a more general feature, the characters *****n*, where *n* is a digit, are replaced by the text matched by the *n*th regular subexpression of the specified RE enclosed between **\(** and **\)**. When nested parenthesized subexpressions are present, *n* is determined by counting occurrences of **\(**, starting from the left.

When the character **%** is the only character in *replacement*, the *replacement* used in the most recent substitute command is used as the *replacement* in the current substitute command. The **%** loses its special meaning when it is in a replacement string containing more than one character or when preceded by a \.

A line can be split by substituting a newline character into it. The newline in *replacement* must be escaped by preceding it by \. Such substitution cannot be done as part of a **g** or **v** command list.

The value of *flags* is zero or more of:

n Substitute for the *n*th occurrence only of the RE found on each addressed line.

g Substitute for all nonoverlapped occurrences of the RE on each addressed line.

l Write to standard output the final line in which a substitution was made. The line is written in the format specified for the **l** command.

n Write to standard output the final line in which a substitution was made. The line is written in the format specified for the **n** command.

p Write to standard output the final line in which a substitution was made. The line is written in the format specified for the **p** command.

(**.,.**)**t***a* Same as **m** command, except that a copy of the addressed lines is placed after address *a* (which can be 0). Upon completion, the new current line is the last line of the copy.

u The **u** (undo) command nullifies the effect of the most recent command that modified anything in the buffer, that is, the most recent **a**, **c**, **d**, **g**, **G**, **i**, **j**, **m**, **r**, **s**, **t**, **v**, or **V** command. All changes made to the buffer by a **g**, **G**, **v**, or **V** global command are "undone" as a single change; if no changes were made by the global command (such as with **g/***RE***/p**), the **u** command has no effect. The current line number is set to the value it had immediately before the command started.

(**1,$**)**v/***RE***/***command-list*
The complement of the global command **g** in that the lines marked during the first step are those that do *not* match the RE.

(**1,$**)**V/***RE***/** The complement of the interactive global command **G** in that the lines marked during the first step are those that do *not* match the RE.

(**1,$**)**w** *file* The **w** (write) command writes the addressed lines into the named file. If the file does not exist, it is created with mode 666 (readable and writable by everyone), unless the current **umask** setting dictates otherwise (see *umask*(1). The remembered file name is not changed unless *file* is the very first file name encountered since **ed** was invoked. If no file name is given, the remembered file name, if any, is used (see the **e** and **f** commands). Upon completion, the current line address is unchanged. If the command is successful, the number of characters written is displayed.

If the *file* name starts with **!**, the rest of the line is interpreted as a shell command whose standard input is the addressed lines. Such a shell command is not remembered as the current file name.

X A key string is demanded from the standard input. Subsequent **e**, **r**, and **w** commands will encrypt and decrypt the text with this key, using the algorithm of *crypt*(1). An explicitly empty key turns off encryption.

(**$**)**=** The line number of the addressed line is displayed. The current line address is unchanged by this command.

!*shell-command*
The remainder of the line after the **!** is sent to the shell to be interpreted and executed as a command. Within the text of that command, the unescaped character **%** is replaced with the remembered file name. If a **!** appears as the first character of the shell command, it is replaced with the text of the previous shell command. Thus, **!!** repeats the last shell command. If any expansion is performed, the expanded line is echoed. Upon completion, the current line address is unchanged.

(**.+1**) *newline* An address alone on a line causes the addressed line to be printed. A newline alone is equivalent to **.+1p**. This technique is useful for stepping forward through the buffer.

If an interrupt signal (ASCII DEL or BREAK) is sent, **ed** prints a **?** and returns to its command level.

The following size limitations apply: 256 characters per global command list, 64 characters per file name, and 32 MB characters in the buffer. The limit on the number of lines depends on the amount of user memory: each line takes 1 word.

EXTERNAL INFLUENCES

Environment Variables

SHELL determines the preferred command-line interpreter for use in all **!**-style commands. If this variable is null or not set, the POSIX shell, **/usr/bin/sh**, is used (see *sh-posix*(1)).

When set, **TMPDIR** specifies a directory to be used for temporary files, overriding the default directory, **/tmp**.

LANG provides a default value for internationalization variables that are unset or null. If **LANG** is unset or null, the default value is "C" (see *lang*(5)). If any internationalization variable contains an invalid setting, all internationalization variables default to "C". See *environ*(5).

If **LC_ALL** is set to a nonempty string value, it overrides the values of all the other internationalization variables, including **LANG**.

LC_CTYPE determines the interpretation of text as single- and/or multibyte characters, the classification of characters as printable, and the characters matched by character class expressions in regular expressions.

LC_MESSAGES determines the locale that should be used to affect the format and contents of diagnostic messages written to standard error and informative messages written to standard output.

NLSPATH determines the location of message catalogues for the processing of **LC_MESSAGES**.

International Code Set Support

Single- and multibyte character code sets are supported.

DIAGNOSTICS

?	Command error. Use **h** or **H** to get a detailed explanation.
?*file*	Inaccessible file. Use **h** or **H** to get a detailed explanation.

If changes have been made in the buffer since the last **w** command that wrote the entire buffer, **ed** warns you if you attempt to destroy the buffer with an **e** or **q** command. **ed** displays **?** or **warning: expecting 'w'**, then continues normal editing unless you enter a second **e** or **q** command, in which case the second command is executed. The **-s** or **-** command-line option inhibits this feature.

EXAMPLES

Make a simple substitution in **file-1** from a shell script, changing the first occurrence of **abc** in any line to **xyz**, and save the changes in **file-2**.

```
cat - << EOF | ed -s file-1
1,$ s/abc/xyz/
w file-2
q
EOF
```

Note that, if a command fails, the editor exits immediately.

WARNINGS

A **!** command cannot be subject to a **g** or a **v** command.

The **!** command and the **!** escape from the **e**, **r**, and **w** commands cannot be used if the the editor is invoked from a restricted shell (see *sh*(1)).

The sequence **\n** in a regular expression does not match a newline character.

The **l** command does not handle DEL correctly.

Files encrypted directly with the **crypt** command with the null key cannot be edited (see *crypt*(1)).

If the editor input is coming from a command file (e.g., **ed file < ed-cmd-file)**, the editor exits at the first failure of a command in the command file.

When reading a file, **ed** discards ASCII NUL characters and all characters after the last newline. This can cause unexpected behavior when using regular expressions to search for character sequences containing NUL characters or text near end-of-file.

AUTHOR

ed was developed by HP and OSF.

FILES

/tmp/e*p*	Temporary buffer file where *p* is the process number.
ed.hup	Work is saved here if the terminal is hung up.

SEE ALSO

awk(1), csh(1), crypt(1), ex(1), grep(1), ksh(1), sed(1), sh(1), sh-bourne(1), sh-posix(1), stty(1), vi(1), fspec(4), environ(5), lang(5), regexp(5).

The **ed** section in *Text Processing: User's Guide*.

STANDARDS CONFORMANCE

ed: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

red: SVID2, SVID3, XPG2, XPG3

NAME

elfdump - dump information contained in object files.

SYNOPSIS

elfdump [**-acCfghHLoprsStuU**] [**-D** *num*] [**+D** *num2*] [**-n** *name*] [**+s** *section*] [**-T** *num*] [**+T** *num2*] *files*...

DESCRIPTION

elfdump takes one or more object files or libraries and dumps information about them. The following options are supported:

-a	Dumps archive headers from an archive library.
-c	Dumps the string table(s).
-C	This modifier causes **elfdump** to demangle C++ symbol names before printing them. This modifier is valid with **-c**, **-r**, **-s**, and **-t**. If specified with **-H**, this modifier is ignored. If specified with **-n** *name*, the symbol whose unmangled name matches *name* will be printed, and its symbol name will be printed as a demangled name.
-D *num*	This modifier causes **elfdump** to print the section whose index is *num*.
+D *num2*	This modifier causes **elfdump** to print the sections in the range 1 to *num2*. If used with **-D**, the sections in the range *num* to *num2* are printed. Valid with **-h**, **-r**, **-s**. If used with **-r**, only the relocations which apply to the section(s) in the range are printed.
-f	This option is used to dump the file header.
-g	This option is used to dump global symbols from an archive.
-h	This option causes **elfdump** to dump the section headers.
-H	This option dumps things in hexadecimal, octal, or decimal format.
-L	This options dumps the .dynamic section in dlls and dynamically linked program files.
-n *name*	This option causes **elfdump** to dump information about the specified section or symbol *name*. This option is valid with **-h**, **-r**, **-s**, and **-t**. If used with **-t**, *name* pertains to a symbol name and **elfdump** will only dump the symbol entry whose name matches *name*. If used with the other options, *name* pertains to a section name and **elfdump** will only dump the section whose name matches it.
-o	Dumps the optional headers (program headers).
-p	Do not print titles.
-r	This option causes **elfdump** to dump the relocations.
-s	Dumps the section contents.
+s *name*	Dumps the section specified by *name*. Valid with **-c** and **-t** only.
-S	This option is used to dump things in short format. Valid with the **-h** and **-o** options.
-t	This option causes **elfdump** to dump the symbol table entries.
-T *num*	This modifier causes **elfdump** to print the symbol whose index is *num*.
+T *num2*	This modifier causes **elfdump** to print the symbols in the range 0 to *num2*. If used with **-T**, print the symbols in the range *num* to *num2*. Valid with **-t**.
-u	Prints the usage menu.
-U	Prints the unwind table.

SEE ALSO

System Tools

ld(1)	Invoke the link editor

Miscellaneous

a.out(4)	Assembler, compiler, and linker output
elf(3e)	Executable and Linking Format

NAME

elm - process electronic mail through a screen-oriented interface

SYNOPSIS

elm [**-aKkmtVz**] [**-f** *folder*]

elm [**-s** *subject*] *address-list*

elm -c [*alias-list*]

elm -h

elm -v

DESCRIPTION

The **elm** program is a screen-oriented electronic mail processing system. It supports the industry-wide MIME standard for nontext mail, a special forms message and forms reply mechanism, and an easy-to-use alias system for individuals and groups. **elm** operates in three principal modes:

- Interactive mode, running as an interactive mail interface program. (First syntax.)
- Message mode, sending a single interactive message to a list of mail addresses from a shell command line. (Second syntax.)
- File mode, sending a file or command output to a list of mail addresses via a command-line pipe or redirection. (Second syntax.)

In all three cases, **elm** honors the values that are set in your **elmrc** initialization file, in your **elm** alias database, and in the system **elm** alias database.

The modes are described below in inverse order (shortest description to longest).

Options

The following options are recognized:

-a Set **arrow=ON**. Use the arrow (**->**) instead of the inverse bar to mark the current item in the various indexes. This overrides the setting of the **arrow** boolean variable (see the ELM CONFIGURATION section).

-c Check alias. Check the aliases in *alias-list* against your personal **elm** alias database and the system **elm** alias database. The results are written to standard output. Errors are reported first, in the form:

(alias "*alias***" is unknown)**

Successes are reported in a header-entry format, with group aliases replaced by their members, in the form:

Expands to: *alias-address* **(***fullname***),**
alias-address **(***fullname***),**
...
alias-address **(***fullname***)**

If there is no *fullname*, the " **(***fullname***)**" portion is omitted.

-f *folder* Folder file. Read mail from the *folder* file rather than from the incoming mailbox. A folder file is in the standard mail file format, as created by the mail system or saved by **elm** itself.

-h Help. Display an annotated list of command-line options.

-k Set **softkeys=OFF**. Disable the use of softkeys (HP 2622 function keys). This overrides the setting of the **softkeys** boolean variable (see the ELM CONFIGURATION section).

-K Set **keypad=OFF** and **softkeys=OFF**. Disable the use of softkeys and arrow cursor keys. If your terminal does not have the HP 2622 function key protocols, this option is required. This overrides the settings of the **keypad** and **softkeys** boolean variables (see the ELM CONFIGURATION section).

-m Set **menu=OFF**. Do not display the command menus on several Interactive Mode screens. This overrides the setting of the **menu** boolean variable (see the ELM

CONFIGURATION section).

-s *subject* Subject. Specify the subject for a File Mode or Message Mode message.

-t Set **usetite=OFF**. Do not use the **termcap ti**/**te** and **terminfo cup** cursor-positioning entries. This overrides the setting of the **usetite** boolean variable (see the ELM CONFIGURATION section).

-V Verbose transmission. Pass outbound messages to the **sendmail** mail transport agent using the **-v** option (see *sendmail*(1M)).

-v Version. Print out the **elm** version information. This displays the revision number and build date as well as the compilation features that were specified or omitted.

-z Zero. Do not enter **elm** if there is no mail in the incoming mailbox.

Operands

The following operands are recognized:

address-list A blank-separated list of one or more mail addresses, your **elm** user aliases, or **elm** system aliases.

alias-list A blank-separated list of one or more of your **elm** user aliases or **elm** system aliases.

Terminology

The following terms are used throughout this manpage.

blank

A space or a tab character, sometimes known as linear white space.

body

The body of a message. See **message**.

boolean variable

See **configuration variable**.

configuration variable

A boolean, numeric, or string variable that defines default behavior in the **elm** mail system. See the ELM CONFIGURATION section.

elm system alias text file

The source file, **/var/mail/.elm/aliases.text**, for the **elm** system alias database.

elm user alias text file

The source file , **$HOME/.elm/aliases.text**, for a user's own **elm** alias database.

elm user headers file

A file, **$HOME/.elm/elmheaders**, where a user can specify special header entries that are included in all outbound messages.

elmrc configuration file

A file, **$HOME/.elm/elmrc**, that defines the initial values for **elm** configuration variables.

environment variable

A global variable set in the shell that called **elm**. See the EXTERNAL INFLUENCES section.

folder

A file that contains mail messages in the format created by **sendmail** or **elm**.

full name

The first and last name of a user, as extracted from an alias text file or from the **/etc/passwd** file.

header

The header of a message. See **message**.

header entry

An entry in the header portion of a message, sometimes called a header field.

incoming mailbox

The mailbox where you receive your mail, usually **/var/mail/***loginname*.

mail directory

The directory, defined by the **maildir** string variable, where a user normally stores mail messages in folders.

mail transport agent (MTA)

The program that sends and receives mail messages to and from other systems. On HP-UX systems, the MTA is **sendmail** (see *sendmail*(1M)).

mailcap

A file that contains information on how to compose and display mail messages that are not just seven- and eight-bit ASCII characters.

metamail

A system program that processes nontext mail messages.

message

In a folder, a sequence of text lines comprised of a message delimiter, a header, and a body. The message delimiter is a line in the form:

From *sender date*

The **header** starts after the message delimiter and ends with the first null line. The **body** begins at the null line and ends at the next message delimiter. A body can have subsections, called **attachments** or **body parts**, which have are comprised of a boundary delimiter, a header, and a body. This process can be recursive. See the METAMAIL CONFIGURATION section for more details.

numeric variable

See **configuration variable**.

sendmail alias database

The alias database, **/etc/mail/aliases**, that is used by the **sendmail** MTA to direct local mail.

signature file

A file that is appended to your outbound messages, usually containing information about yourself. You can have two signature files, one for messages to your local machine and one for other messages. See the **localsignature** and **remotesignature** string variables.

string variable

See **configuration variable**.

user name

Usually the login or mailbox name of someone you send mail to.

variable

See **configuration variable** and **environment variable**.

FILE MODE

If standard input is connected to a pipe or to a file, and an *address-list* is specified, **elm** operates in File Mode.

The output of the previous command in the pipe, or the content of the file, is mailed to the members of the *address-list*. The *address-list* is expanded, based on your **elm** alias database and the system **elm** alias database, and placed in the **To:** header entry.

If **-s** is omitted or *subject* is null, *subject* defaults to:

no subject (file transmission)

The expressed or default value of *subject* is placed in the **Subject:** header entry.

See the EXAMPLES section.

MESSAGE MODE

If standard input is connected to your terminal, and an *address-list* is specified, **elm** operates in Message Mode.

The *address-list* is expanded, based on your **elm** alias database and the system **elm** alias database, and placed in the **To:** header entry. The **To:** header entry is displayed, in the same form as for the Message Menu **m** (mail) command in Interactive Mode.

The value of *subject*, if nonnull, or a null string, is placed in the **Subject:** header entry and the **Subject:** line is displayed for modification.

If **askcc** is **ON** in your **elmrc** file, you are prompted for **Copies to:**.

Then the editor defined by the **editor** string variable (if a signature file is not added) or the **alteditor** string variable (if a signature file is added) is started so that you can write your message.

When you leave your editor, you enter the Send Menu, as described for Interactive Mode.

If you choose the Send Menu **s** (send) command, the message is sent and the program terminates. If you select the Send Menu **f** (forget) command, the message is stored in **$HOME/Canceled.mail** and the program terminates. If you select other commands, the appropriate action occurs.

See the EXAMPLES section.

INTERACTIVE MODE

If standard input is connected to your terminal, and there is no *address-list*, **elm** operates in a screen-oriented Interactive Mode.

If you do not have a **$HOME/.elm** directory, or if you do not have a mail directory, defined by the **maildir** string variable, you are asked in turn if they should be created. You can answer **y** for *yes*, **n** for *no*, or **q** for *quit*. For **y** or **n**, the directories are created or not, as appropriate, and the program continues. For **q**, the program terminates.

Overview

When invoked, **elm** reads customized variables from file **$HOME/.elm/elmrc** (if it exists) to initialize parameters. This file can be saved from within **elm** and some of these variables can also be modified with the Message Menu **o** (option) command.

elm first displays the Main or Message Menu, which shows index entries for the messages in your incoming mailbox or selected mail folder. Among other options, you can read, print, reply to, and forward these messages, as well as initiate new mail messages to other users.

You can also move to the Alias Menu, where you can create, modify, and delete your personal aliases. From the Alias Menu, you can select one or more of your aliases and send a message to the corresponding users.

When you send a message, you can include attachments in a number of formats, such as PostScript, images, audio, and video, as well as plain text. The attachments are managed separately, which can be convenient both for you and your correspondents.

Sending Messages

When you send a message, you use the editor defined by the **editor** or **alteditor** string variable. If **builtin** is your editor, a set of commands described in the Built-In Editor subsection is available while composing your message

If the **elmheaders** file exists (see the HEADER FILE section), all nonblank lines in the file are copied to the headers of all outbound mail. This is useful for adding special information headers such as **X-Organization:**, **X-Phone:** , and so forth.

MIME Support

elm supports the MIME protocols for headers and messages (RFC 1521 and RFC 1522) enabling it to view and send mail containing other than normal ASCII text. For example, the mail contents can be audio, video, images, etc., or a combination of these.

This also enables conformance with SMTP (RFC 821), which allows only 7-bit characters in the message, by using MIME-encoding (**base64** and **quoted-printable**) to convert 8-bit data to 7-bit.

elm also provides a facility to view multipart MIME messages. If **elm** receives a message whose type is not **text/plain**, it invokes **metamail**, which invokes the appropriate utility (for example, **ghostview**, **xv**, an audio editor, **mpeg**) to display the different mail parts according to the content type (for example, **application/postscript**, **image**, **audio**, **video**).

Aliases

elm has its own alias system that supports both personal and system-wide aliases. Personal aliases are specific to a single user; system aliases are available to everyone on the system where the system aliases reside (see *newalias*(1)). You can access the Alias Menu by executing the Message Menu **a** (alias) command. You can then create and save an alias for the current message, create and check other aliases, and send messages to one or more aliases.

Aliases are limited to 2500 bytes. If you wish to create a group alias that is longer than 2500 bytes, please ask your system administrator to create it for you in the **sendmail** system alias file, **/etc/mail/aliases** (see *sendmail*(1M)).

INTERACTIVE MODE MENUS AND COMMANDS

This section begins with the Message Menu, which is the main screen for Interactive Mode. The rest of the menus are presented alphabetically.

Message Menu

The Message Index is displayed on the Message Menu. You can use the following commands to manipulate and send messages. Some commands use a series of prompts to complete their action. You can use **Ctrl-D** to cancel their operations.

The commands are:

!*command*	Shell Escape. Send *command* to the shell defined by the **shell** string variable without leaving **elm**.
#	Display all known information about the current message.
$	Resynchronize the messages without leaving **elm**. If there are any messages marked for deletion, you are asked if you want to delete them. If any messages are deleted or any status flags have changed, the messages are written back to the mailbox file. All tags are removed.
%	Display the computed return address of the current message.
*****	Set the current message pointer to the last message.
+	Display the next message index page, when applicable.
-	Display the previous message index page, when applicable.
/*pattern*	Pattern match. Search for *pattern* in the *from* and *subject* fields of the current message index. The search starts at the current message and wraps around to the beginning of the index. The current message pointer is set to the first message that matches. Uppercase and lowercase are treated as equivalent.
//*pattern*	Pattern match. Search for *pattern* through all the lines of the current folder. The search starts at the current message and wraps around to the beginning of the folder. The current message pointer is set to the first message that matches. Uppercase and lowercase are treated as equivalent.
<	Calendar. Scan message for calendar entries and add them to your calendar file. A calendar entry is defined as a line whose first nonblank characters are **->**, as in: **->***calendar-entry* The delimiter **->** and surrounding blanks are removed before the entry is added to the calendar file. Resultant blank lines are ignored. You can define the calendar file name in your **elmrc** file or with the Options Menu.
=	Set the current message pointer to the first message.
>	Save in folder. Same as the Message Menu **s** (save) command.

?*key* ...	Help on key. Display a one-line description of what each *key* does. **?** displays a summary listing for each command available. A period (**.**) returns you to the Message Menu.
@	Display a summary of the messages indexed on the current screen.
|	Pipe the current message or the set of tagged messages through other filters as desired. Use the shell defined by the **shell** string variable.
n	New current message. Change the current message pointer to the one indexed as *n*. If the message is not on the current page of headers, the appropriate page displayed.
Return	Read current message. The screen is cleared and the current message is displayed by the pager defined by the **pager** string variable.
a	Alias. Switch to the Alias Menu.
b	Bounce mail. This is similar to forwarding a message, except that you do not edit the message and the return address is set to the original sender's address, rather than to your address.
c	Change folder. This command is used to change the file whose messages are displayed on the Message Menu. You are asked for a file name. The file must be in message format; otherwise, **elm** aborts. You can use the customary wildcards for your shell, as well as the following special names:

!	Your incoming mail folder.
>	Your received folder, defined by the **receivedmail** string variable.
<	Your sent folder, defined by the **sentmail** string variable.
.	The previously used folder.
@*alias*	The default folder for the login name associated with the *alias* alias.
=*filename*	A file in the directory defined by the **maildir** string variable.

C	Copy message. Save the current message or the set of tagged messages to a folder. You are prompted for a file name with a default value. The default value is a file in the **maildir** directory with the user name of the sender of the first message in the set being saved. Any tags are cleared. Unlike the **>** and **s** commands, the messages are not marked for deletion and the current message pointer is not moved.
d	Delete. Mark the current message for deletion. See also **Ctrl-D**, **u**, and **Ctrl-U**.
Ctrl-D	Delete. Mark all messages for deletion that contain a specified pattern in the **From:** and **Subject:** header entries. See also **d**, **u**, and **Ctrl-U**.
e	Edit. Allows you to physically edit the current mail folder using the editor defined by the **editor** string variable. When you exit from your editor, **elm** resynchronizes your mail folder (see the **$** command).
f	Forward the current message. You are asked if you want to edit the outbound message. If you answer **y**, the characters defined by the **prefix** string variable are prefixed to each line of the message and the editor defined by the **editor** string variable will be invoked to allow you to edit the message. If you answer **n**, the characters are not prefixed and the editor will not be invoked. In either case, you are prompted for **To:** recipients, allowed to edit the **Subject:** header entry, and, if the **askcc** boolean variable is **ON**, you are prompted for **Cc:** recipients.

If the **userlevel** numeric variable is **1** (intermediate) or **2** (expert), and there was a previous sent or forgotten message in this session, you are asked if you would like to

```
Recall last kept message instead? (y/n)
```

If you answer **y**, the previous message is returned to the send buffer. If you answer **n**, the current message is copied into the send buffer and your signature file (if any) is appended.

Then the editor is invoked if you chose to edit the outbound message (above). When you leave the editor, or if it was not invoked, the Send Menu is displayed.

g	Group reply. The reply is automatically sent **To:** the sender of the message, with **Cc:** to all the original **To:** and **Cc:** recipients. Otherwise, the action is the same as for the **r** command.
h	Same as **Return**, except that the message is displayed with all headers.
j	Move down. Move the current message pointer down to the next message.
J	Move down. Move the current message pointer down to the next undeleted message.
k	Move up. Move the current message pointer up to the previous message.
K	Move up. Move the current message pointer up to the previous undeleted message.
l (ell)	Limit the displayed messages to those that contain certain string values. You are prompted with **Enter criteria:**. To set, add to, or clear the limiting criteria, type one of:

all	Clear all the criteria and restore the normal display.
from *string*	Restrict to entries that contain *string* in the **From:** header.
subject *string*	Restrict to entries that contain *string* in the **Subject:** header.
to *string*	Restrict to entries that contain *string* in the **To:** header.

You can add limiting criteria by repeating the **l** command.

Ctrl-L	Redraw the screen.
m	Mail. Send mail to one or more addresses. You are prompted for **To:** recipients, a **Subject:** and, if the **askcc** boolean variable is **ON**, **Cc:** recipients.

If the **userlevel** numeric variable is **1** (intermediate) or **2** (expert), and there was a previous sent or forgotten message in this session, you are asked if you would like to

Recall last kept message instead? (y/n)

If you answer **y**, the previous message is returned to the send buffer. If you answer **n**, the signature file (if any) is copied into the send buffer.

Then, the editor defined by the **editor** string variable is invoked. After you exit from your editor, the Send Menu is displayed.

n	Next message. Advances the current message pointer to the next message, and displays that message as for the **Return** command.
o	Options. Invokes the Options Menu, permitting you to change certain configuration options. The changeable options are defined by the **configoptions** string variable.
p	Print. Print the current message or the set of tagged messages using the command defined by the **print** string variable. The current message pointer does not move. Tagged messages remain tagged.
q	Quit. Gracefully terminate, performing message cleanup according to defined personal preferences. You can choose to actually delete messages marked for deletion. For your incoming mailbox, you can choose to keep undeleted mail in the mailbox or move it to the received folder defined by the **receivedmail** string variable.

If the **ask** boolean variable is **ON**, you may be asked the following questions. The actions described are all performed after you have answered all the relevant questions.

Delete messages? (y/n)

This question is asked if you have messages marked for deletion. The default answer is provided by the **alwaysdelete** boolean variable (**ON** means **y** (yes) and **OFF** means **n** (no)).

If you answer **y**, all messages marked for deletion will be deleted.

If you answer **n**, all messages marked for deletion will be restored to their former read, unread, or new state.

Move read messages to "received" folder? (y/n)

This question is asked if you are reading your incoming mailbox and if you have messages that have been read. The default answer is provided by the **alwaysstore** boolean variable (**ON** means **y** (yes) and **OFF** means **n** (no)).

If you answer **y**, undeleted messages that have been read will be moved to the folder defined by the **receivedmail** string variable and the next question will also be asked.

If you answer **n**, all undeleted messages are returned to your incoming mailbox and the next question is not asked.

Keep unread messages in incoming mailbox? (y/n)

This question is asked if you are reading your incoming mailbox, if you answered **y** to the **Move read messages**... question (or it was not asked), and if you have messages that have not been read. The default answer is provided by the **alwayskeep** boolean variable (**ON** means **y** (yes) and **OFF** means **n** (no)).

If you answer **y**, all undeleted unread (new and old) messages are returned to your incoming mailbox.

If you answer **n**, all undeleted unread messages will be moved to the folder defined by the **receivedmail** string variable.

If the **ask** boolean variable is **OFF**, the answers to the questions (which are not displayed) are taken automatically from the values of the **alwaysdelete**, **alwaysstore**, and **alwayskeep** boolean variables, respectively.

Q Quick quit. This is equivalent to executing the **q** command with the **ask** boolean variable set to **OFF**.

r Reply to the sender of the current message. If the **autocopy** boolean variable is **OFF**, you are asked if the source message should be copied into the edit buffer. If it is **ON**, the message is copied automatically. If copied in, all lines from the message are preceded by the prefix string defined by the **prefix** string variable. The **To:** header is set to the sender of the message (or the address in the **Reply-To:** header, if one was set), the **Subject:** is set to the subject of the message, preceded by **Re:**, and presented for you to edit. If the **askcc** boolean variable is **ON**, you are prompted for **Cc:** recipients. Then, the editor defined by the **editor** string variable is invoked. After you exit from your editor, the Send Menu is displayed.

s Save in folder (same as **>**). Save the current message or the set of tagged messages to a folder. You are prompted for a file name with a default value. The default value is a file in the **maildir** directory with the login name of the sender of the first message in the set being saved. Any tags are cleared and the messages are marked for deletion. The current message pointer is moved to the first undeleted message after the last saved message.

t Tag toggle. Tag the current message for a later operation and move the current message pointer to the next undeleted message. The operation can be one of **|**, **C**, **p**, and **s**.

Or, remove the tag from a tagged message. See also the **Ctrl-T** command.

T Tag toggle. Tag the current message for a later operation and remain at the current message. The operation can be one of **|**, **C**, **p**, and **s**.

Or, remove the tag from a tagged message. See also the **Ctrl-T** command.

Ctrl-T Tag all messages containing the specified pattern. Or remove the tags from all tagged messages.

If any messages are currently tagged, you are asked if the tags should be removed. Answer **y** to remove the old tags; answer **n** to keep them. In either case, you are prompted for a string to match in either the **From:** or **Subject:** line of each message. All messages that match the criterion are tagged. If you enter a null string (carriage-return alone), no more messages are tagged.

u	Undelete. Remove the deletion mark from the current message. See also **d**, **Ctrl-D**, and **Ctrl-U**.
Ctrl-U	Undelete. Remove any deletion mark from all messages that contain a specified pattern in the **From:** and **Subject:** header entries. See also **d**, **Ctrl-D**, and **u**.
v	View attachments. Invoke the Attachment View Menu for the current message.
x	Exit. Exit without changing the mailbox. If changes are pending, such as deletions, you are asked if they can be abandoned. If you answer **y**, the changes are abandoned and the program terminates. If you answer **n** the exit is abandoned and you return to the Message Menu command prompt.
X	Exit immediately without changing the mailbox. All pending changes are abandoned.

Message Index

The messages in the current folder are indexed on the Message Menu, one per line, in the format:

sssnum mmm d from **(***lines***)** *subject*

defined as:

sss	A three-character status field, described in the Message Status subsection.
num	The ordinal message index number.
mmm	The month from the last **Date:** header entry, or from the **From** message header.
d	The day from the last **Date:** header entry, or from the **From** message header.
from	Either the sender name from the last **From:** header entry or from the **From** message header.
lines	The number of lines in the message.
subject	The subject description from the first **Subject:** header entry, truncated to fit your screen.

The current message index entry is either highlighted in inverse video or marked in the left margin with an arrow (**->**). See the **-a** option in the Options subsection and the **arrow** string variable in the ELM CONFIGURATION section.

Message Status

The first three characters of each message index entry describe the message status. Each can be blank or one of the values described below in descending order of precedence.

When a message has more than one status flag of a particular type set, the highest-precedence indicator is displayed on the index line. For example, if a forms message (**F**) is also marked as company confidential (**C**), the **C** rather than the **F** status character is displayed.

Column One: Variable Status

D Deleted. The message is marked for deletion.

E Expired. The date specified in the **Expires:** header entry is older than today. **elm** accepts the following date formats:

Mon, 11 Jun 90	(format produced by **elm** in the Header Menu)
Jun 11, 90	
11 Jun, 90	
9006111324GMT	(ISO X.400 format: *YYMMDDhhmmzzz*)

N New. The message was received after your last **elm** session or during the current session. The message has not been read.

O Old. The message was received before or during your last **elm** session. It was marked **N** in your last session and it was not read.

Blank. The message has been read.

Column Two: Permanent Status

C	Confidential. The **Sensitivity: 3** header entry is present. The message is considered company confidential, as specified by the ISO X.400 standard. You can set this value for outbound mail with the user-defined option of the Header Menu.
U	Urgent. The message contains a **Priority:** header entry.
P	Private. The **Sensitivity: 2** header entry is present. The message is considered private, as specified by the ISO X.400 standard. You can set this value for outbound mail with the user-defined option of the Header Menu.
A	Action. The message contains an **Action:** header entry.
F	Forms. The message is an **elm** forms message. The message contains a **Content-Type: mailform** header entry.
M	MIME. The message or its attachments is in a MIME format that can be displayed using **metamail**.
?	MIME. The message or its attachments is in a MIME format whose version is not supported.
	Blank. Normal status.

Column Three: Tagged Status

+	Tagged. Tagged messages are handled as a group by some commands. See **t** and other commands in the Message Menu subsection.
	Blank. The message is not tagged.

Built-In Editor

When you are composing an outbound message with the **builtin** built-in editor, it prompts you for text lines with an empty line. Enter a period (**.**) to end the message and continue with the Send Menu.

Built-in editor commands are lines that begin with an escape character, defined by the **escape** string variable. The default escape character is tilde (˜).

Note: Some remote login programs use tilde as their default escape character when it is the first character on a line. (You can tell, because the tilde does not print.) Usually, the tilde is transmitted when you enter a second character that is not recognized by the program or when you enter a second tilde. See the program documentation for further information.

The built-in editor commands are:

~! [*command*]	Execute the shell *command*, if one is given (as in **˜!ls**), or start an interactive shell, using the shell defined by the **shell** string variable.
~< *command*	Execute the shell *command* and place the output of the command into the editor buffer. For example, "**˜< who**" inserts the output of the **who** command in your message.
~?	Print a brief help menu.
~˜	Start a line with a single tilde (˜) character.
~b	Prompt for changes to the Blind-Carbon-Copy (**Bcc:**) list.
~c	Prompt for changes to the Carbon-Copy (**Cc:**) list.
~e	Invoke the editor defined for the **easyeditor** string variable on the message, if possible.
~f [*options*]	Add the specified list of messages or the current message. This uses **readmail** which means that all **readmail** options are available (see *readmail*(1)).
~h	Prompt for changes to all the available headers (**To:**, **Cc:**, **Bcc:**, and **Subject:**).
~m [*options*]	Same as **˜f**, but each line is prefixed with the current prefix. See the **prefix** string variable.
~o	Prompt for the name of an editor to use on the message.
~p	Print out the message as typed in so far.

~r *filename*	Include (read in) the contents of the specified file.
~s	Prompt for changes to the **Subject:** line.
~t	Prompt for changes to the **To:** list.
~v	Invoke the editor defined for the **visualeditor** string variable on the message, if possible.

Alias Menu

The Alias Menu is invoked with the Message Menu **a** command. The source text for your alias file is stored in the file **$HOME/.elm/aliases.text**. You can edit this file directly or with the following commands.

The aliases currently compiled into your database and the system database are displayed in an indexed list similar to the Message Menu. The entry format is described in the Alias Index subsection. The index is sorted in the order defined by the **aliassortby** string variable.

The commands are:

$	Resynchronize your alias text file and your alias database by rebuilding the database from the text file by running **newalias**. Aliases marked for deletion are removed, tagged aliases are untagged, and new and changed aliases are recognized. The alias screen is updated to reflect these changes.
+	Display the next alias index page, when applicable.
-	Display the previous alias index page, when applicable.
/*pattern*	Pattern match. Search for *pattern* in the alias and user name fields of the alias list. The search starts at the current alias and wraps around to the beginning of the alias list. The current alias pointer is set to the first alias that matches. Uppercase and lowercase are treated as equivalent.
//*pattern*	Pattern match. Search for *pattern* through all the fields of the alias list (alias, user name, comment, and address). The search starts at the current alias and wraps around to the beginning of the alias list. The current alias pointer is set to the first alias that matches. Uppercase and lowercase are treated as equivalent. **/***pattern* Pattern match. This command allows you to search through all the alias and username entries in the alias list, starting at the current alias and continuing through the end. If the first character of the pattern is a **/**, then the comment and the fully expanded address fields are also included in the search. The search is case-insensitive. This allows you to find a specific alias in a situation where there are a large number of aliases.
?*key ...*	Help on key. Display a one-line description of what each *key* does. **?** displays a summary listing for each command available. A period (**.**) returns you to the Alias Menu.
a	Alias current message. This allows you to create an alias that has the return address of the current message as the address field of the alias. It prompts for a unique alias name and allows you to edit the comment and address fields.
c	Change the current user alias. The old values of the alias fields are used as the defaults in the prompts for the new values. You cannot change the alias name. If the alias name is one of a multiple-alias record, it is removed from that record and stored as a separate record. The old alias is marked **N**. Changes are effective after the next alias resynchronization.
d	Mark the current user alias for deletion. The deletions are made when you exit from the Alias Menu with an **q**, **r**, or **i** command or you resynchronize your alias database with the **$** command. (You cannot delete a system alias in this way.)
Ctrl-D	Delete user aliases with a specified search pattern.
e	Edit your **aliases.text** file, using the editor defined in the **editor** string variable. Your aliases are resynchronized when you finish editing (see the **$** command).
f	Display a fully expanded alias. The currently selected alias is fully expanded and displayed.

`i`,`I`	See the Alias Menu `q` and `Q` commands.
`j`	Move down. Move the current alias pointer down to the next alias.
`J`	Move down. Move the current alias pointer down to the next undeleted alias.
`k`	Move up. Move the current alias pointer up to the previous alias.
`K`	Move up. Move the current alias pointer up to the previous undeleted alias.
`l` (ell)	Limit the displayed aliases to certain types or those that contain certain string values. You are prompted with `Enter criteria:`. To set, add to, or clear the limiting criteria, type one of:

`all`	Clear all the criteria and restore the normal display.
`alias` *string*	Restrict to alias names containing *string*.
`name` *string*	Restrict to full names (first name and last name) containing *string*.
`group`	Restrict to group aliases (can include system and user aliases).
`person`	Restrict to person aliases (can include system and user aliases).
`system`	Restrict to system aliases (can include group and person aliases).
`user`	Restrict to system aliases (can include group and person aliases).

You can add limiting criteria by repeating the `l` command.

Ctrl-L	Redraw the screen.
`m`	Mail to the current alias or to the set of tagged aliases. The corresponding expanded addresses are placed in the `To:` header entry, and processing continues as for the Message Menu `m` (mail) command. The tags are cleared.
`n`	Make a user alias. `elm` prompts for a unique alias name, then for an address. The information provided is added to your individual alias_text file (`$HOME/.elm/aliases.text`), then added to the database.
`q`	Exit. Return to the Message Menu. If aliases are marked for deletion, you are asked if you want to delete them. The alias index pointer is retained. If the alias text file was changed, the database is resynchronized.
`Q`	Exit. Return to the Message Menu. If aliases are marked for deletion, the mark is retained and the alias is not deleted. The alias index pointer is retained. If the alias text file was changed, the database is resynchronized.
`r`,`R`	See the Alias Menu `q` and `Q` commands.
`t`	Tag the current alias for a later operation and move the current alias pointer to the next undeleted alias. The operation can be one of `c`, `m`, or `n`. Or, remove the tag from a tagged alias. See also the **Ctrl-T** command.
`T`	Tag. Tag the current alias for a later operation and remain at the current alias. The operation can be one of `c`, `m`, or `n`. Or, remove the tag from a tagged alias. See also the **Ctrl-T** command.
Ctrl-T	Tag all aliases containing a specified pattern for a later operation. The operation can be one of `c`, `m`, or `n`. Or remove the tags from all tagged aliases. If any aliases are currently tagged, you are asked if the tags should be removed. Answer `y` to remove the old tags; answer `n` to keep them. In either case, you are prompted for a string to match in either the alias name or user name fields of each alias. All aliases that match the criterion are tagged. If you enter a null string (carriage-return alone) no more aliases are tagged.
`u`	Undelete. Remove the deletion mark from the current alias. See also `d`, **Ctrl-D**, and **Ctrl-U**.
Ctrl-U	Undelete. Remove any deletion mark from all messages that contain a specified pattern in the `From:` and `Subject:` header entries. See also `d`, **Ctrl-D**, and `u`.

e

v View. Display the *address-list* for the current alias.

x Exit from the alias menu without processing any deletions. Aliases marked for deletion are unmarked and **newalias** is not run, even if alias additions have been made.

Alias Index

The aliases in the current database are indexed on the Alias Menu, one per line. The database values are defined in *newalias*(1).

ssnum fullname[**,** *comment*] *type* [**(S)**] *alias*

defined as:

ss A two-character status field. The first character can be:

- **D** Delete. The alias is marked for deletion.
- **N** New. The alias is new or changed in the alias text file but is not included in the current database. Resynchronization is needed.
- Blank. The alias is in the current database.

The second character can be:

- **+** Tag. The alias is tagged.
- Blank. The alias is not tagged.

num The index number of the alias.

fullname The full name for the alias, as it will be used in an expanded address. It has the form:

firstname lastname

firstname The first name, from the alias database.

lastname The last name, from the alias database.

comment Comment, from the alias database.

type Type of alias. This is **Person** for an alias with a single address or **Group** for an alias with two or more addresses.

(S) If present, the entry is from the **elm** system alias database. If absent, the entry is from your personal alias database.

alias The alias name. If the record has multiple alias names, there is one index entry per name.

Attachment Configuration Menu

The Attachment Configuration Menu is invoked with the Attachment Send Menu **a** (add) or **m** (modify) command. The menu displays the default or current specification for an attachment. If it is called with the **a** command, it automatically prompts for a file name. The commands are:

d Description. The value is placed in a **Content-Description:** body-part header entry. The default is the file name.

e Content-Transfer-Encoding. This is the method by which the file is encoded to allow it to pass through various Mail Transport Agents. The choices are:

7bit Unencoded, normal **US-ASCII** text.

8bit Unencoded 8-bit characters with the high-order bit set.

quoted-printable
Text with control characters and high-order-bit characters converted to a string in the form **=***hh*, where *hh* is the hexadecimal representation of the character. An **=** at the end of a line indicates that the source line was broken into two lines.

base64 Any file type with bits encoded in 6-bit groups and rendered in numeric order as the **US-ASCII** characters **A**–**Z**, **a**–**z**, **0**–**9**, **+**, and **/**. The last line may be padded to a multiple of 4 characters with **=** characters.

binary Unencoded binary data.

The value is placed in a **Content-Transfer-Encoding:** body-part header entry. The default is **7bit**.

f File name. The name of the file to be attached. **elm** examines the file and sets the values of Content-Transfer-Encoding, Content-Disposition, and Content-Type accordingly.

p Content-Disposition. The value is placed in a **Content-Disposition:** body-part header entry. The default is **attachment; filename=***filename*.

t Content-Type. The type of the file and supporting parameters, in the form:

*type***/***subtype* [**;** *parameter*]...

The *type* can be one of **application**, **audio**, **image**, **message**, **text**, or **video**, as defined in RFC 1521. Although **multipart** is also a valid type, you cannot specify it directly; **elm** provides it as necessary and handles messages that contain it. The value is placed in a **Content-Type:** body-part header entry. The default is:

text/plain; charset=US-ASCII

Some common entries are described below. See the METAMAIL CONFIGURATION section for additional information.

text/*subtype* [**; charset=***charset*]

This is relatively readable text that may be formatted with embedded text characters, as for possible subtypes **richtext** or **html**. The default *subtype* is **plain** (unformatted in any way). The default *charset* is **US-ASCII**.

application/octet-stream

This is a catch-all for files such as program binary, or files that contain control characters or characters with high-order bits set.

application/postscript

The file can be displayed with a PostScript-equipped printer or viewer.

message/rfc822

This specifies that the file is in message format, as described in the Terminology subsection.

image/jpeg, **image/gif**

These are picture formats that require a display program.

audio/basic

This is an audio format that requires a reproduction program.

video/mpeg

This is an audio/video format that requires a reproduction program.

Attachment Send Menu

The Attachment Send Menu is invoked with the Send Menu **a** command. This menu displays a list of the attachments that will be sent in a message, one per line, as described in the Attachment Index subsection.

The commands are:

a	Add attachments. Call the Attachment Configuration Menu and prompt for a file name.
d	Delete an attachment.
e	Edit an attachment. Call the editor associated with the attachment if it is editable.
j	Move the current attachment pointer down to the next attachment.
k	Move the current attachment pointer up to the previous attachment.
Ctrl-L	Redraw the screen.
m	Modify the attributes of an attachment. Call the Attachment Configuration Menu.
p	Print an attachment. See the Message Menu **p** (print) command.
q	Quit. Return to the Send Menu.
s	Save an attachment. See the Message Menu **C** (copy) command.

Attachment View Menu

The Attachment View Menu is invoked with the Send Menu **v** command. This menu displays a list of the attachments in a folder message, one per line, as described in the Attachment Index subsection.

The commands are:

Return	Display the current attachment.
j	Move the current attachment pointer down to the next attachment.
k	Move the current attachment pointer up to the previous attachment.
Ctrl-L	Redraw the screen.
p	Print the attachment. See the Message Menu **p** (print) command.
q	Quit. Return to the previous attachment level or the Message Menu.
s	Save the attachment. The attachment is saved in the form it was received, as with the Message Menu **s** (save) command.
v	View the subattachment list, if any.

Attachment Index

Attachments are listed on the Attachment Send Menu and the Attachment View Menu in the following format:

num filename **(***size***)** *format* **[***encoding***]**

defined as:

num	The index number of the attachment.
filename	The name of the attached file.
size	The size of the attachment in bytes, computed from the file or the message.
*type***/***subtype*	The type and subtype of the attachment. This value is placed or found in a **Content-Type:** header.
encoding	The encoding type. This value is placed or found in a **Content-Transfer-Encoding:** header.

Header Menu

The Header Menu is invoked with the Send Menu **h** command. It allows you to add, change, and delete a set of common header entries in your message. In general, if an item is empty, it is not included in the message.

The commands are:

Return	Return to Send Menu.
!*command*	Shell. Execute *command* with the shell defined by the **shell** string variable.
a	**Action:** header. Enter any string. If this entry is present in a received message, **elm** displays an **A** in the Permanent Status column of the Message Index.
b	**Bcc:** header. Enter a list of aliases and actual addresses. Aliases are expanded and shown as addresses and user names.
c	**Cc:** header. Enter a list of aliases and actual addresses. Aliases are expanded and shown as addresses and user names.
d	Domainize. Convert non-Internet addresses to Internet format. The UUCP **!** format (*host***.***domain***!***user*) becomes the Internet **@** format (*user***@***host***.***domain*). If **.***domain* is omitted, it defaults to **.uucp**.
e	**Expires:** header. Enter any numeric value from 1 to 56 (8 weeks). If this entry is present in a received message, **elm** displays an **E** in the Variable Status column of the Message Index when the computed date has passed.
i	**In-Reply-To:** header. Enter a string.
n	**Precedence:** header. Enter a precedence name. If the **precedences** string variable is set and nonnull, the name must be one defined by the variable. If the

name is associated with a priority, and the **Priority:** header is null, the priority value is inserted in the **Priority:** header. If **precedences** is null or not set, you can enter any value.

If the precedence name matches one defined in the **sendmail** configuration file **/etc/mail/sendmail.cf**, the transmission priority is modified accordingly. If there is no match, the priority is not changed.

p **Priority:** header. Enter a string. If this entry is present in a received message, **elm** displays a **U** in the Permanent Status column of the Message Index

r **Reply-To:** header. Enter a personal alias or a single address. If it is present, **elm** and other mailers use this header instead of the **From:** header when choosing the address for a reply (Message Menu **r** (reply) command).

s **Subject:** header. Enter a string.

t **To:** header. Enter a list of aliases and actual addresses. Aliases are expanded and shown as addresses and user names.

u User-defined header. Define your own header entry in the form:

*header-name***:** *header-string*

*header-name***:** must not contain blanks. You can use this command to create a **Sensitivity:** header entry, as described in the Message Status subsection, or a different header, but only one. See the HEADER FILE section for another way to include user-defined header entries.

Options Menu

The Options Menu is invoked with the Message Menu **o** command. It displays a list of the options, defined by the **configoptions** string variable, that you can modify while **elm** is running. Enter the appropriate letter (in upper- or lowercase) that is followed with a right parenthesis (**)**) and follow the directions on the screen. The full set of option prompts and the corresponding variables is listed below. The default options are marked with an *****.

A)rrow cursor	The **arrow** string variable. *
B)order on copy	The **prefix** string variable.
C)alendar file	The **calendar** string variable. *
D)isplay mail using	The **pager** string variable. *
E)ditor (primary)	The **editor** string variable. *
F)older directory	The **maildir** string variable. *
H)old sent message	The **copy** boolean variable.
J) reply editor	The **alteditor** string variable.
K) pause after pager	The **promptafter** boolean variable.
A(l)ias Sorting	The **aliassortby** string variable.
M)enu display	The **menu** boolean variable. *
N)ames only	The **names** boolean variable. *
O)utbound mail saved	The **sentmail** string variable. *
P)rint mail using	The **print** string variable. *
R)eply copies msg	The **autocopy** boolean variable.
S)orting criteria	The **sortby** string variable. *
T)ext editor (˜e)	The **easyeditor** string variable.
U)ser level	The **userlevel** numeric variable. *
V)isual Editor (˜v)	The **visualeditor** string variable. *
W)ant Cc: prompt	The **askcc** boolean variable.
Y)our full name	The **fullname** string variable. *
Z) signature dashes	The **sigdashes** boolean variable.

Note: The menu displays the first *screen-height***-6** lines from the defined set. *screen-height* is the number of text lines on the screen. If an option is not displayed, it cannot be modified.

When you are done, enter one of the following values:

> Save the current configuration values in your configuration file, **$HOME/.elm/elmrc**. If the file does not exist, it is created. This is a convenient way to make an configuration file that you can edit directly, as well as with the Options Menu.

i,**I**	Return to the Message Menu.
q,**Q**	Return to the Message Menu.
x,**X**	Exit immediately from **elm** without changing the mailbox. All pending changes are abandoned.

Send Menu

The Send Menu is invoked when your outbound message has been prepared to be mailed after a Message Menu **f**, **g**, **m**, or **r** command or the Alias Menu **m** command.

The commands are:

!*command*	Shell. Execute a shell command. See the Message Menu **!** (shell) command.
a	Attachments. Invoke the Attachments Send Menu.
c	Copy. Copy to a file. See the Message Menu **C** (copy) command.
e	Edit. Invoke your editor, as defined by the **alteditor** string variable, to revise the message.
f	Forget. Do not send the message. At user levels **1** and **2**, the message may be returned to the send buffer when you execute a subsequent Message Menu **f**, **g**, **m**, or **r** command or the Alias Menu **m** command.
h	Edit the header entries. Invoke the Header Menu.
m	Make form. Convert the message to the forms message format. See the FORMS MESSAGES section. This command is only available if the **forms** boolean variable is **ON** and the **userlevel** numeric variable is either **1** or **2**.
s	Send. Send the message.

FORMS MESSAGES

A feature that is unique to **elm** is the ability to compose and reply to forms messages.

Creating a Forms Message

- In your **elmrc** file, set **forms=ON**.
- Set your **userlevel** numeric variable to **1** (moderately experienced) or **2** (expert). You can do this in your **elmrc** file or on the default Options Menu.
- As you compose the message, define the fields to be filled in by the recipient with a colon (**:**), followed by either the number of spaces allowed for the field value, or a newline to indicate that the field may fill the remainder of the line.

  A colon on a line by itself indicates that the recipient will be prompted for multiline input. There can be no blanks before the colon.

  Every line containing a colon is a prompt line. During the response process, all text starting at the first nonblank character after the last colon on each line is deleted and the line is evaluated for response fields.
- After you have created the message, enter the Send Menu **m** (make form) command to set up the special format. Then enter the Send Menu **s** (send) command to send the message.

Here is an example of a simple forms message:

```
On-Line Phone and Address Database
Please fill out and return as soon as possible.
Name:
Manager:
Department:                                 Division:
Your home address:
Home phone number:
Thank you for your cooperation.
```

Replying to a Forms Message

When you receive a forms message, the message index entry is flagged with an **F** status letter. You can view it in the normal way with the **Return** or **h** commands.

To reply, use the Message Menu **r** (reply) command (you cannot use the Message Menu **g** (group reply) command). **elm** prompts you for each field, with any text present between the fields displayed as appropriate. The example above is presented line-by-line; user input is in italic type:

```
On-Line Phone and Address Database
Please fill out and return as soon as possible.
Name: my name
Manager: my manager
Department: my department
Division: my division
Your home address: home address
Home phone number: phone number
Thank you for your cooperation.
```

The received message would look like this:

```
On-Line Phone and Address Database
Please fill out and return as soon as possible.
Name: my name
Manager: my manager
Department: my department                 Division: my division
Your home address: home address
Home phone number: phone number
Thank you for your cooperation.
```

HEADER FILE

The **$HOME/.elm/elmheaders** file provides you with a way to specify special information headers such as **X-Organization:**, **X-Phone:**, and so forth. The nonblank lines from this file are added to the headers of all outbound mail.

Entries in the **elmheaders** file should have the following format:

*header-name***:** *header-string*

*header-name***:** must not contain blanks. *header-string* can be continued over several lines by preceding each continuation line with blanks, as indicated by the output below.

Within the **elmheaders** file, you can use backquotes (left apostrophes) to execute shell commands when the file is read, so that an entry of the form:

```
X-Operating-System: ‘uname -a‘
```

would produce a header entry like:

```
X-Operating-System:
        HP-UX hpulpc17 B.10.10 A 9000/710 2012505939 two-user license
```

According to RFC 822, user-defined header names should begin with **X-** or **x-**. Otherwise, they risk having their usage overridden if the name is later standardized with some other meaning.

Defined Headers

The following header names are defined for the message header in RFC 822 and RFC 1521.

Bcc: (822)	**Cc:** (822)
Comments: (822)	**Content-Description:** (1521)
Content-ID: (1521)	**Content-Transfer-Encoding:** (1521)
Content-Type: (1521)	**Date:** (822)
Encrypted: (822)	**From:** (822)
In-Reply-To: (822)	**Keywords:** (822)
MIME-Version: (1521)	**Message-ID:** (822)
Received: (822)	**References:** (822)
Reply-To: (822)	**Resent-Bcc:** (822)
Resent-Cc: (822)	**Resent-Date:** (822)
Resent-From: (822)	**Resent-Message-ID:** (822)
Resent-Reply-To: (822)	**Resent-Sender:** (822)
Resent-To: (822)	**Return-Path:** (822)
Sender: (822)	**Subject:** (822)

To: (822)	**X-***user-defined***:** (822)

Other commonly used headers:

Action:	**Apparently-To:**
Content-Disposition:	**Content-Length:**
Expires:	**Mailer:**
Newsgroups:	**Precedence:**
Priority:	**Sensitivity:**
Status:	**X-Mailer:**

ELM CONFIGURATION

elm supports user configuration by means of the **$HOME/.elm/elmrc** configuration file. You can create the configuration file with the Options Menu **>** command. It can contain any combination of the string, numeric, and boolean variables described below.

String Variables

String variables have the form

string-name **=** *string-value*

The following string variables are defined.

aliassortby The sort order for the alias index in the Alias Menu. The recognized values are:

alias	Sort by alias name.
name	Sort by the full name of the alias, last name first.
text	Sort by the order of the aliases in the alias text file.

Prefix the value with **reverse-** to reverse the sort order. The default is **name**.

alteditor The name of the editor to use for messages that have initial text (a copied message in a reply, a signature in any outbound message, etc.). when the **editor** string variable is set to **none** or **builtin**. The default is the value of the **EDITOR** environment variable, if set and nonnull, or **/usr/bin/vi** otherwise. See also the **editor** string variable.

alternatives A list of other machine and user name combinations that you receive forwarded mail from. **elm** uses this information when a group reply is being processed to ensure that a reply message is not sent to a user and/or machine address that would simply forward the reply message back to you. The default is none.

attribution Attribution string for replies. When you reply to a message and include the message in the reply, this string is placed at the top of the message. The characters **%s** are replaced by the full name of the author of the original message. The default is none. For example:

```
attribution = %s wrote:
```

calendar The name of your calendar file. This is used in conjunction with the Message Menu **<** (calendar) command, which scans messages for calendar entries. The default is **$HOME//calendar**.

charset The name of the character set used with the MIME **Content-Type:** header for the **text/plain** type. It can be any Internet-defined character set name that is a superset of **US-ASCII**. The default is **US-ASCII**. For example,

```
Content-Type: text/plain; charset=US-ASCII
```

compatcharsets A list of Internet-defined character sets that are supersets of US-ASCII, so that messages with **charset=US-ASCII** can be displayed without the help of **metamail**. The default is a string containing the following values:

```
Extended_UNIX_Code_Packed_Format_for_Japanese
ISO-2022-JP
ISO-8859-1
```

ISO-8859-2
ISO-8859-3
ISO-8859-4
ISO-8859-5
ISO-8859-7
ISO-8859-8
ISO-8859-9
KOI8-R
Shift_JIS

configoptions
A string of options that can be configured on the Options Menu. Specify the options as a single letter each, in the order they should be displayed. The default is "**^_cdefsopyv_am_un**". The defaults are marked below with an *****.

The option characters include:

^	The menu title.
_	A blank line.
a	The **arrow** string variable. *****
b	The **prefix** string variable.
c	The **calendar** string variable. *****
d	The **pager** string variable. *****
e	The **editor** string variable. *****
f	The **maildir** string variable. *****
h	The **copy** boolean variable.
j	The **alteditor** string variable.
k	The **promptafter** boolean variable.
l	The **aliassortby** string variable.
m	The **menu** boolean variable. *****
n	The **names** boolean variable. *****
o	The **sentmail** string variable. *****
p	The **print** string variable. *****
r	The **autocopy** boolean variable.
s	The **sortby** string variable. *****
t	The **easyeditor** string variable.
u	The **userlevel** numeric variable. *****
v	The **visualeditor** string variable. *****
w	The **askcc** boolean variable.
y	The **fullname** string variable. *****
z	The **sigdashes** boolean variable.

displaycharset
The name of the character set supported by the display. This is independent of the **charset** string variable. This is also copied to the **MM_CHARSET** environment variable when **metamail** is called. The default is **US-ASCII**.

easyeditor
The name of the editor for the **~e** command of the built-in editor. See also the **editor** string variable. The default is none.

editor
The name of the editor to use when creating new mail. The default is the value of the **EDITOR** environment variable, if set and nonnull, or **/usr/bin/vi** otherwise.

You can use **none** or **builtin** to specify the built-in editor. The built-in editor is available for all outbound mail that does not already have text in the send buffer (no forwarded message, no copied message in a reply, no signature in any outbound message, etc.). If there is text in the send buffer and **builtin** is specified, the editor defined by the **alteditor** variable is used instead.

See also the **alteditor**, **easyeditor**, and **visualeditor** string variables.

escape
The escape character used in the built-in editor. The default is tilde (**~**).

e

e

fullname
The name the mailer will use when sending mail from you. The default is the full name portion (everything up to the first comma) of the **pw_gecos** field from your entry in the **/etc/passwd** file. This field can be set with the **chfn** command (see *chfn*(1), *finger*(1), and *passwd*(4)).

localsignature
A signature file that is automatically appended to outbound mail to the local host before the editor is invoked. This usually contains personal data about the sender. See also the **remotesignature** string variable. The default is none.

All the addresses in the **To:** header must be apparently for the local host. Local addresses are those that, after any **elm** alias conversion, do not contain a domain name. That is, they have only a user name (for example, **santaclaus**) or a user name and the local host name (for example, **santaclaus@northpole**).

santaclaus@northpole.arcticsea.org is considered to be a remote address, even if it points to the local host. A user name that is readdressed by the **sendmail** system alias list is treated as local if it matches the preceding criteria.

maildir
Your mail directory, where you usually store your folders for received and outbound mail. The default is **$HOME//Mail**.

In **elm**, you can use the **=** metacharacter to specify this directory. For example, if you save a message to file **=/archive**, the **=** is expanded to the current value of **maildir**. (The slash (**/**) is optional.)

When you start **elm**, if the directory specified by **maildir** does not exist, you are asked if you want to create it. If you answer **y** (yes), the directory is created, with access permissions set to **700**.

pager
The program to display each message. The default is the value of the **PAGER** environment variable, if set and nonnull, or the built-in pager, **builtin+**, otherwise.

The built-in pager, **builtin+**, also allows you to execute some Message Menu commands while you are viewing the message and it has some simple forward and backward scrolling commands. While it is active, enter **?** for a list of commands. An alternative is the **more** utility.

precedences
A list of precedence values that you can place in a **Precedence:** header entry in outbound mail, using the Header Menu. Each precedence value can be optionally paired with a priority value that is automatically placed in a **Priority:** header entry, causing the received message to be marked as urgent. The default is none.

The HP-UX mail transport agent, **sendmail**, recognizes this header. If the precedence value is defined by a **P** control line in the **sendmail** configuration file, **/etc/mail/sendmail.cf**, the transmission priority of the message is adjusted accordingly. See *sendmail*(1M).

The format of the entry is

precedences = *precedence*[**:***priority*] [*precedence*[**:***priority*]] ...

precedence is a precedence name. The default list defined in **/etc/mail/sendmail.cf** is:

first-class	Transmission priority 0, the default
special-delivery	Transmission priority 100
list	Transmission priority –30
bulk	Transmission priority –60
junk	Transmission priority –100

priority is an arbitrary string that is placed in a **Priority:** header entry.

prefix
The prefix for an included line in an outbound message. When you reply to a message or forward a message to another person, you can optionally include the original message. This prefix marks the included line. The

default is **>_** (the _ is interpreted as a space character).

print
The command to run when the **p** (print) command is executed from various menus. There are two possible formats for this string: If the string contains the special variable **%s**, the variable is replaced by the name of a temporary file that contains the messages, and the command is executed by the shell defined by the **shell** string variable. If the string does not contain **%s**, the temporary file name is appended to it, and the command is executed. The default is

```
cat %s | lp
```

receivedmail
The file where the received messages will be saved. The default is **=received**, the file **received** in the directory defined by **maildir**.

remotesignature
A signature file that is automatically appended to all outbound mail to remote hosts before the editor is invoked. This usually contains personal data about the sender. See also the **localsignature** string variable. The default is none.

If any of the addresses in the **To:** header entry are not local, as described for the **localsignature** string variable, the remote signature file is attached.

savecharset
The character set to be used to save a message in a folder. Possible values are **JIS**, **SJIS**, and **EUC**. If a value is not specified, the message will be saved according to your locale (given by the **LC_TYPE** and/or **LANG** environmental variables). This option is applicable only for the Japanese locale. The default is none. See also the **jisconversion** boolean variable.

sentmail
The file where copies of outbound mail can be saved. One possibility is your incoming mailbox, **/var/mail/***loginname*. The default is **=sent**, the file **sent** in the directory defined by **maildir**.

See the **copy** boolean variable for further details.

shell
The shell to use for **!** escapes and other such operations. The default is the value of the **SHELL** environment variable, if set and nonnull, or **/usr/bin/ksh** otherwise.

sortby
The way to sort the index of the current folder. The choices are:

from	The name of the sender.
sent	The date the message was sent.
received	The date the message was received.
subject	The subject of the message. A leading **Re:** (and some others) is ignored, so replies sort with original messages.
lines	The number of lines in the message.
status	The read status: blank, **O**, and **N**.

You can prefix these values with **reverse-** to reverse the order of the sort. The value can be modified on the Options Menu. The default is **reverse-sent**.

textencoding
Type of encoding to put into the MIME **Content-Transfer-Encoding:** header entry. The choices are **7bit** or **8bit**. The default is **7bit**.

tmpdir
Where to create temporary files. The default is the value of the **TMPDIR** environment variable, if set and nonnull, or to **/tmp/** otherwise.

visualeditor
Name of the editor to use for the **~v** command of the built-in editor. The default is the value of the **VISUAL** environment variable, if set and nonnull, or **/usr/bin/vi** otherwise.

weedout
A list of header-entry initial strings that you don't want to see when you are reading mail. This list is made effective by setting the **weed** boolean

variable to **ON**.

The list can continue for as many lines as desired, as long as the continued lines all have leading blanks. To include blanks in a string, enclose it in quotation marks (**"**). The strings you specify are normally appended to the default list, which is:

```
>From
Apparently-To:
Content-Length
Content-Transfer-Encoding
Content-Type:
From
In-Reply-To:
MIME-Version
Mailer:
Message-Id:
Newsgroups:
Received:
References:
Status:
X-Mailer:
```

There are two special values:

clear-weed-list

Clear the default list. The default headers are removed from the **weedout** list, allowing you to completely define your own list.

end-of-user-headers

Mark the end of the **weedout** list, in case any following lines could be mistaken for headers in the list.

The default value of **weedout** is ***end-of-user-headers***.

The underscore (_) character can be used to specify a space.

Note that **From** weeds out both **From** and **From:**. If, for example, you want to weed out **From** but not **From:**, specify ***clear-weed-list*** followed by **From_** and any other headers that you don't want to see.

Numeric Variables

Numeric variables have the form

numeric-name **=** *numeric-value*

The following numeric variables are defined.

bounceback — Threshold for returning copies of remote UUCP messages. If the destination host is greater than the specified number of hops (gateways) from your local host, the destination host sends you a copy of the message when it is received. If the value is **0**, this feature is disabled. The default is **0**.

builtinlines — Determines if the built-in pager should be used on some messages, even if you usually use an external pager, defined in the **pager** string variable. There are two ways of defining whether the built-in pager should be used.

- If you want to use the built-in pager on any message that is shorter than *n* lines, set the value to *n*.
- If you want to use the built-in pager on any message that is *m* lines shorter than the number of lines on your screen, set the value to **-***m*.

If you set the value to **0**, the message will always be sent through the external pager. The default is **-3**.

noencoding — This enables you to send raw 8-bit or binary data when the mail transfer agent doesn't support the **8BITMIME** and the **-B8BITMIME** options. The default is **0**.

The possible values are:

0 Always convert 8-bit and binary messages to 7-bit before sending them.

1 Convert 8-bit messages to 7-bit, but depend on **sendmail** to handle binary messages.

2 Depend on **sendmail** to handle both 8-bit and binary messages.

readmsginc The value by which the **Reading in** *folder***, message:** counter is incremented while reading a new folder. If you set this value to a number larger than one, it will speed up the time it takes to read a large folder when you are using a slow terminal. The default is **1**.

sleepmsg The time in seconds that **elm** will wait after displaying a diagnostic message before erasing it. The value can be **0** or a positive integer. The default is **2**.

timeout The interval, in seconds, after which **elm** rechecks the incoming mailbox for new mail. The default is **600** (10 minutes).

userlevel The relative level of your user sophistication. Acceptable values are:

0 Novice user (the default). Command menus are a small verbose subset of the available commands.

1 Moderately experienced user. Command menus are a larger terse subset of the available commands. Outbound message commands allow you to recover previously unsent messages as the text of the current outbound message.

2 Expert. The features are the same as for **1**.

Level **1** or **2** is required if you want to send a forms message.

Boolean Variables

Boolean variables have the forms

boolean-name **= ON** –and– *boolean-name* **= OFF**

The following boolean variables are defined.

alwaysdelete If **ON**, the default answer of the Message Menu **q** (quit) prompt

```
Delete messages? (y/n)
```

is set to **y** (yes). If **OFF**, the default answer is set to **n** (no). The default is **OFF**. See the Message Menu **q** command.

alwayskeep If **ON**, the default answer of the Message Menu **q** (quit) prompt

```
Keep unread messages in incoming mailbox? (y/n)
```

is set to **y** (yes). If **OFF**, the default answer is set to **n** (no). The default is **ON**. See the Message Menu **q** command.

alwaysstore If **ON**, the default answer of the Message Menu **q** (quit) prompt

```
Move read messages to "received" folder? (y/n)
```

is set to **y** (yes). If **OFF**, the default answer is set to **n** (no). The default is **OFF**. See the Message Menu **q** command.

arrow If **ON**, use an arrow (**->**) to mark the current item in a menu index. If **OFF**, use an inverse bar. If the program is invoked with the **-a** command line option, **arrow** is set to **ON**. The default is **OFF**.

ask If **ON**, you are asked the questions

```
Delete messages? (y/n)
Move read messages to "received" folder? (y/n)
Keep unread messages in incoming mailbox? (y/n)
```

e

(as appropriate) each time you leave the program with the Message Menu **q** (quit) command. See that command for details of the process. If **OFF**, or if you use the Message Menu **Q** command, **elm** uses the values defined by the **alwaysdelete**, **alwaysstore**, and **alwayskeep** boolean variables, respectively, without prompting. The default is **ON**.

askcc If **ON**, **elm** prompts you for "carbon copies" with the prompt **Copies To:** each time you send, forward, or reply to a message. If **OFF**, the prompt is omitted. In either case, you can still explicitly include **Cc:** addresses with the **~c** command in the built-in editor, or with the Header Menu commands. The default is **ON**.

autocopy If **ON**, **elm** automatically copies the text of the message you are replying to into the edit buffer. If **OFF**, **elm** prompts you with **Copy message?**. The default is **OFF**.

confirmappend If **ON**, you are asked to confirm before messages are appended to an existing file, whether it is a file in your mail directory or a file in another directory. If **OFF**, see **confirmappend** and **confirmfiles** Operation below. The default is **OFF**.

confirmcreate If **ON**, you are asked to confirm before a new file is created. whether it is a file in your mail directory or a file in another directory. If **OFF**, see **confirmcreate** and **confirmfolders** Operation below. The default is **OFF**.

confirmfiles If **ON**, you are asked to confirm before messages are appended to an existing file that is not in your mail directory. This does not affect files in your mail directory. If **OFF**, see **confirmappend** and **confirmfiles** Operation below. The default is **OFF**.

confirmfolders If **ON**, you are asked to confirm before a new file is created in your mail directory. This does not affect files in other directories. If **OFF**, see **confirmcreate** and **confirmfolders** Operation below. The default is **OFF**.

confirmcreate and **confirmfolders** Operation

confirmcreate=ON and **confirmfolders=ON**
Confirm before creating a file in your mail directory. Confirm before creating a file another directory.

ON and **OFF** Confirm before creating a file in your mail directory. Confirm before creating a file another directory.

OFF and **ON** Confirm before creating a file in your mail directory. Do not confirm before creating a file another directory.

OFF and **OFF** Do not confirm before creating a file in your mail directory. Do not confirm before creating a file another directory.

confirmappend and **confirmfiles** Operation

confirmappend=ON and **confirmfiles=ON**
Confirm before appending to a file in your mail directory. Confirm before appending to a file in another directory.

ON and **OFF** Confirm before appending to a file in your mail directory. Confirm before appending to a file in another directory.

OFF and **ON** Confirm before appending to a file in your mail directory. Do not confirm before appending to a file in another directory.

OFF and **OFF** Do not confirm before appending to a file in your mail directory. Do not confirm before appending to a file in another directory.

copy If **ON**, save silent copies of all outbound mail on the outbound step. If **OFF**, do not save copies. The default is **OFF**.

If **ON**, and the **savename** boolean variable is **ON**, **elm** first tries to save it to a file named as defined by **savename**. If the file exists, the message is saved. If the file does not exist, but the **forcename** boolean variable is **ON**, the file is created and the message is saved. If **forcename=OFF**, the message is saved to the file defined by the **sentmail** string variable. If **savename=OFF**, the message is saved to the file defined by the **sentmail** string variable.

forcename
If **ON**, create the folder when saving outbound messages by the login name of the recipient, even if the folder doesn't already exist. If **OFF**, do not create the folder. The default is **OFF**.

See the **copy** boolean variable for further details.

forms
If **ON**, and the **userlevel** numeric variable is **1** or **2**, you can create a forms message. The Send Menu **m** (make form) command converts your message into a forms message. If **OFF**, you cannot. The default is **ON**.

jisconversion
If **ON**, convert outbound mail to JIS (Japanese Industrial Standard) before sending it. If **OFF**, do not convert it. This option is applicable only to the Japanese locales, **ja_JP.SJIS** and **ja_JP.eucJP**. The default is **OFF**. **savecharset** string variable.

keepempty
If **ON**, keep folders from which all the messages are deleted. If **OFF**, delete empty folders. The default is **OFF**.

keypad
If **ON**, enable the HP 2622 terminal cursor keys. If **OFF**, disable the cursor keys. If the program is invoked with the **-K** command line option, **keypad** is set to **OFF**. See also the **softkeys** boolean variable. The default is **ON**.

menu
If **OFF**, this inhibits the menu display on all program screen displays. If **ON**, the menus are displayed. If the program is invoked with the **-m** command line option, **menu** is set to **OFF**. The default is **ON**.

metoo
If **ON**, you are sent a copy of the message that you send to an alias that includes your name also. If **OFF**, the copy is not sent. The default is **OFF**.

mimeforward
If **ON**, a forwarded message is sent as an attachment. If **OFF**, a forwarded message is sent as part of the main message. The default is **ON**.

movepage
If **ON**, commands that move through the mailbox by pages (the **+** and **-** commands) also move the current index pointer to the top of the new page of messages. If **OFF**, moving through the pages does not alter the current message pointer location. The default is **ON**.

names
If **ON**, show only the user names when expanding the **To:** aliases for an outbound message. If OFF , show the entire expanded addresses. The default is **ON**.

nohdrencoding
If **ON**, don't do RFC 1522 encoding for header values that contain 8-bit or multibyte characters. If **OFF**, do the encoding. The default is **OFF**.

noheader
If **ON**, do not include the headers of messages when copying a message into a file buffer for replying to or forwarding. If **OFF**, copy all headers. The default is **ON**.

noheaderfwd
If **ON**, do not include headers when copying a message into a file buffer for forwarding. If **OFF**, copy all headers. For forwarding, this option overrides the setting of **noheader**. The default is **OFF**.

pagemultipart
If **ON**, use the value of the **pager** variable to display MIME multipart messages with unknown subparts or with unknown subtypes. If **OFF**, call **metamail** to view multipart messages. The default is **OFF**.

pointnew
If **ON**, automatically point to the first new message in your message index at start-up. If **OFF**, point to the first message. In either case, if the start-up folder is not your incoming mailbox, or if there are no new messages, point to the first message. The default is **ON**.

promptafter	If **ON**, prompt for a command after the external pager exits. If **OFF**, return to the calling menu. The default is **ON**.
resolve	If **ON**, move the pointer to the next message in the index, after deleting, undeleting, saving, or forwarding a message. If **OFF**, keep the pointer at the current message. The default is **ON**.
savename	If **ON**, and you are saving a message, **elm** constructs a suggested file name in your **maildir** directory from the user name of the person who sent the message, in the form **=***username*. If **OFF**, no file name is suggested. If **ON**, and you are sending a message that will be saved, **elm** constructs a file name based on the user name of the first entry in the **To:** list, in the same form as above. If **OFF**, no file name is constructed. See the **copy** boolean variable for further details. The default is **ON**.
sigdashes	If **ON**, insert two dashes above the signature text, included from a local or remote signature file. This is a common convention. If **OFF**, omit the dashes. The default is **ON**.
softkeys	If **ON**, enable the HP 2622 terminal function-key protocol. If **OFF**, disable the function-key protocol. If the program is invoked with the **-k** or **-K** command line option, **softkeys** is set to **OFF**. See also the **keypad** boolean variable. The default is **OFF**.
titles	If **ON**, title a displayed message with a line in the form: **Message** *number***/***total sendername* *date time* *sendername*, *date*, and *time* are extracted from the message headers in the manner described in Message Index. This is useful if you have suppressed the relevant header entries with the **weedout** list. If **OFF**, the message is not titled. The default is **ON**.
usetite	If **ON**, use the **termcap ti**/**te** and **terminfo cup** cursor-positioning entries (see *terminfo*(4)). If **OFF**, do not use those entries. If the program is invoked with the **-t** command line option, **usetite** is set to **OFF**. The default is **ON**.
weed	If **ON**, do not display the headers defined by **weedout** variable when displaying a message for reading. If **OFF,** display all headers. The default is **ON**.

e

METAMAIL CONFIGURATION

MIME (Multipurpose Internet Mail Extensions) encoding classifies the message and its attachments according to a Content-Transfer-Encoding, which is the encoding, if any, that is used to make the message mailable, and a Content-Type, which is the type and form of the message part after it has been decoded. The encoding and types are described in more detail in the Attachment Configuration Menu subsection and in RFC 1521.

elm provides built-in support for the following Content-Types:

text/plain [**; charset=***charset*]

The text is all in the displayable character set *charset* which defaults to **US-ASCII**.

multipart/mixed ; boundary=*boundary-string*

The message is composed of a number of individual "body parts", separated by **--***boundary-string*, each having optional headers defining Content-Type and Content-Transfer-Encoding. The default Content-Type is **text/plain**.

multipart/digest ; boundary=*boundary-string*

This is similar to **multipart/mixed**, except that the default Content-Type is **message/rfc822**.

multipart/report ; boundary=*boundary-string*

message/rfc822

The message consists of another message in standard message format.

metamail is a system program that is invoked by **elm** to manage the display of messages and attachments that are not displayable in ordinary ASCII text.

metamail provides external support for other Content-Types, as defined in one or more **mailcap** files. The system **mailcap** file is **/etc/mail/mailcap**. You can define your own default **mailcap** file in **$HOME/.mailcap**. You can also specify your own list of **mailcap** files by setting the **MAILCAPS** environment variable. The **mailcap** files are searched in order until an entry is found that matches the Content-Type and any qualifications.

A minimum **mailcap** entry consists of a line in the form:

content-type **;** *command*

The *command* is the command that you would type to view a file of the indicated Content-Type, with the string **%s** replaced by a file name. For example, to view body part that was HTML source text and had the Content-Type **text/html**, you could have the entry

```
text/html; netscape %s
```

Similarly, for a GIF image file, you could have the entry

```
image/gif; xv %s
```

RFC 1521 defines a number of Content-Types that **elm** leaves for **metamail** to handle:

```
text/richtext
multipart/alternative
multipart/parallel
multipart/digest
message/partial
message/external-body
image/jpeg
image/gif
audio/basic
video/mpeg
application/octet-stream
application/postscript
```

Check the system **mailcap** file for entries that handle many of them.

EXTERNAL INFLUENCES

Environment Variables

HOME	Your home (login) directory.
EDITOR	If set and nonnull, provides a default value for the **alteditor** and **editor** string variables.
LANG	If set and nonnull, determines the language in which messages are displayed. The default is **C**. See *environ*(5).
MAILCAPS	If set, defines the search path for **mailcap** files used by **metamail**. The default is **$HOME/.mailcap:/etc/mail/mailcap**
PAGER	If set and nonnull, provides a default value for the **pager** string variable.
SHELL	If set and nonnull, provides a default value for the **shell** string variable.
TMPDIR	If set and nonnull, provides a default value for the **tmpdir** string variable.
VISUAL	If set and nonnull, provides a default value for the **visualeditor** string variable.

International Code Set Support

Single- and multibyte character code sets are supported.

EXAMPLES

Message Mode Example

To send a message without loading the main **elm** mail-processing program, use the simple command form consisting of the name of the program followed by the recipient's login name and optional address. **elm** prompts for subject and copies, then starts an editor so you can compose the message (user responses are in italic type):

```
$ elm j_doe
To: doe (John Doe)
Subject: this is a test
Copies To: ...
...invokes editor, you compose message, then...
Your options now are:
a)ttachments e)dit message edit h)eader s)end it f)orget it.

What is your choice? s
mail sent!
```

If you "forget" the message, it is saved in **$HOME/Canceled.mail**.

File Mode with Redirection

To send a file by use of command-line redirection, use a command like:

```
$ elm j_doe < help.c
```

which reads file **help.c** and sends it with the default subject.

File Mode with a Pipe

To mail the output of a command and include a subject line:

```
$ ls -a | elm -s "Directory Listing" j_doe
```

WARNINGS

Using two separate mail programs to access the same mail file simultaneously (usually inadvertently from two separate windows) can cause unpredictable results.

AUTHOR

elm was developed by Hewlett-Packard Company.

FILES

$HOME/.elm	Directory for the user's **elm** alias, configuration, header, and other files
$HOME/.elm/aliases	User alias database data table
$HOME/.elm/aliases.dir	User alias database directory table
$HOME/.elm/aliases.pag	User alias database hash table
$HOME/.elm/aliases.text	User alias source text
$HOME/.elm/elmheaders	User-defined additional headers
$HOME/.elm/elmrc	User configuration file
$HOME/Canceled.mail	Canceled message in noninteractive use.
/tmp/alias.*pid*	Temporary file for deleting alias
/tmp/form.*pid*	Editor buffer for forms message
/tmp/mbox.*loginname*	Temporary mailbox for user *logname*
/tmp/print.*pid*	Temporary file for printing message
/tmp/snd.*pid*	Outgoing mail message edit buffer
/tmp/sndh.*pid*	Outgoing mail header edit buffer
/usr/lib/nls/msg/C/elm.cat	Location of the message catalog
/usr/share/lib/elm/elmrc-info	Comment file for **$HOME/.elm/elmrc** file
/var/mail	Directory for incoming mail; it must have mode **755** and group ID **mail**
/var/mail/.elm	Directory for **elm** mailer system aliases
/var/mail/.elm/aliases	System alias database data table
/var/mail/.elm/aliases.dir	System alias database directory table
/var/mail/.elm/aliases.pag	System alias database hash table
/var/mail/.elm/aliases.text	System alias source text

/var/mail/*loginname*	Incoming mailbox for user; it must have mode **660** and group ID **mail**
/var/mail/*loginname***.lock**	Lock for mail directory

SEE ALSO

answer(1), chfn(1), elmalias(1), fastmail(1), finger(1), mailfrom(1), newalias(1), newmail(1), readmail(1), vi(1), sendmail(1M), passwd(4), terminfo(4), environ(5).

RFC 821 "Simple Mail Transfer Protocol (SMTP)"

RFC 822 "Standard for the Format of Internet Text Messages"

RFC 1521 "MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies"

RFC 1522 "MIME (Multipurpose Internet Mail Extensions) Part Two: Message Header Extensions for Non-ASCII Text"

NAME

elmalias - display and verify elm user and system aliases

SYNOPSIS

elmalias [**-dersu**] [**-a**|**-f** *format*|**-n**|**-v**|**-V**] [*alias-name-list*]

Remarks

The former functionality of the **elmalias** command has been taken over by the **newalias** command (see *newalias*(1)).

DESCRIPTION

The **elmalias** command displays and verifies user and system **elm** aliases.

The system database must have been created by the **newalias** command (see *newalias*(1)). The user database must have been created by either the **newalias** command or the **elm** mail system (see *elm*(1)). If the same alias is in both databases, the user version is used. Missing database files are silently ignored.

Each database entry can have the following fields, which are described in detail in *newalias*(1):

alias-list	A list of one or more aliases for the entry.
address-list	A list of one or more addresses for the entry. An address can be an alias from another entry's *alias-list*.
comment	An optional field containing information about the entry. This field is not included in outbound mail.
firstname	An optional field interpreted as the first name of the person or group. It is used in *fullname*.
lastname	An optional field interpreted as the last name of the person or group. It is used in *fullname*.
fullname	A combination value made up from the *firstname* and *lastname* fields, in the form: *firstname lastname*.

elmalias recognizes three types of alias names:

person	A database entry that has one address in *address-list*. **elmalias** assumes this address is a valid mailing address.
group	A database entry that has two or more addresses in *address-list*. **elmalias** assumes initially that these addresses are aliases to **person** or **group** entries.
unknown	An address in a **group** entry or an alias in *alias-name-list* that is not an alias in the database. In both cases, the item is reported as both the alias name and its address.

With no options or operands, **elmalias** displays the *address-list* field for each alias in the two databases. If an entry has more than one alias, the *address-list* field is displayed multiple times.

With an *alias-name-list* and no options, **elmalias** displays the *address-list* field of each alias name in the list. If an alias name is not found in the databases, it is treated as **unknown**, without comment.

Options

elmalias recognizes the following options:

-a Change the display to alias name followed by *address-list* field.

-d Turn debugging on.

-e Fully expand **group** aliases. This option can be used only when an *alias-name-list* is given.

If an address in a **group** address list is an alias name, it is replaced by that alias entry. The process is recursive until the results are either **person** or **unknown** types. If a **group** address is not an alias name, it is reported as both the alias name and the address, with type **unknown**. Duplicate alias names are reported only once. **person** entries are not expanded, even if their addresses are actually aliases.

-f *format* Display the *format* string for each alias name in the file or in the *alias-name-list*. The following character pairs are replaced in the *format* by the corresponding value for each alias.

%a	The alias name.
%c	The *comment* field.
%l	The *lastname* field.
%n	The *fullname* value.
%t	The alias type: **person**, **group**, or **unknown**.
%v	The *address-list* field.

-n Change the display to *address-list* followed by *fullname*, if any, in parentheses.

-r Report an error if a name in *alias-name-list* does not correspond to an alias in the database. Display a message for each unknown name and exit with a nonzero status.

-s Use the system alias database only, unless **-u** is also specified.

-u Use the user alias database only, unless **-s** is also specified.

-v Use a verbose output format. Change the display to alias name, followed by *address-list*, followed by *fullname*, if any, in parentheses.

-V Use a very verbose, multiline output format with the following titles, corresponding to the format codes of the **-f** option. If a field is empty, the title is omitted.

```
Alias:
Address:
Type:
Name:
Last Name:
Comment:
```

EXIT STATUS

elmalias sets the following exit status values:

0 Normal completion.

<>0 An error occurred. You may have specified:

- An invalid option.
- The **-e** option without an *alias-name-list*.
- The **-f** option without a *format*.
- The **-r** option with an unknown alias name in *alias-name-list*.

EXAMPLES

Consider a user database that contains the following entries:

```
# sample alias file
mom = My Mommy, Work: x2468 = my_mother@a.computer
dad,father,pop = Father; Dear, Work: x1357 = host!otherhost!dad
parents = The Folks = mom dad parent@host
siblings = The Kids = brother1
    brother2
    sister
brother1 = Son; First = bro1@kid.computer
```

Since **brother2** and **sister** do not refer to alias entries, they are typed as **unknown.**

elmalias with no options or operands produces the following output.

```
my_mother@a.computer
host!otherhost!dad
host!otherhost!dad
host!otherhost!dad
mom,dad,parent@host
brother1,brother2,sister
bro1@kid.computer
```

elmalias -v produces the alias names, address, and full name, as follows:

```
    mom                     my_mother@a.computer (My Mommy)
    dad                     host!otherhost!dad (Dear Father)
    father                  host!otherhost!dad (Dear Father)
    pop                     host!otherhost!dad (Dear Father)
    parents                 mom,dad,parent@host (The Folks)
    siblings                brother1,brother2,sister (The Kids)
    brother1                bro1@kid.computer (First Son)
```

To expand a set of aliases and format them with field titles, use the **-e** and **-f** options, as in the following command:

```
    elmalias -ef "Alias: %a  Address: %v  Type: %t" parents siblings
```

producing:

```
    Alias: mom  Address: my_mother@a.computer  Type: Person
    Alias: dad  Address: host!otherhost!dad  Type: Person
    Alias: parent@host  Address: parent@host  Type: Unknown
    Alias: brother1  Address: bro1@kid.computer  Type: Person
    Alias: brother2  Address: brother2  Type: Unknown
    Alias: sister  Address: sister  Type: Unknown
```

AUTHOR

elmalias was developed by HP.

FILES

$HOME/.elm/aliases	User alias database data table
$HOME/.elm/aliases.dir	User alias database directory table
$HOME/.elm/aliases.pag	User alias database hash table
$HOME/.elm/aliases.text	User alias source text
/var/mail/.elm/aliases	System alias database data table
/var/mail/.elm/aliases.dir	System alias database directory table
/var/mail/.elm/aliases.pag	System alias database hash table
/var/mail/.elm/aliases.text	System alias source text

SEE ALSO

elm(1), newalias(1).

NAME

enable, disable - enable/disable LP printers

SYNOPSIS

enable *printers*

disable [**-c**] [**-r**[*reason*]] *printers*

DESCRIPTION

The **enable** command activates the named *printers*, enabling them to print requests taken by **lp**. Use **lpstat** to find the status of printers (see *lp*(1) and *lpstat*(1)).

disable deactivates the named *printers*, disabling them from printing requests taken by **lp**. By default, any requests that are currently printing on the designated printers are reprinted in their entirety either on the same printer or on another member of the same class. Use **lpstat** to find the status of printers.

Options

disable recognizes the following options:

-c	Cancel any requests that are currently printing on any of the designated printers.
-r[*reason*]	Associate a *reason* with the deactivation of the printers. This reason applies to all printers mentioned up to the next **-r** option. If the **-r** option is not present or the **-r** option is given without a reason, a default *reason* is used. *reason* is reported by **lpstat**. The maximum length of the *reason* message is 80 bytes.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages are displayed.

If **LANG** is not specified or is null, it defaults to **C** (see *lang*(5)).

If any internationalization variable contains an invalid setting, all internationalization variables default to **C** (see *environ*(5)).

International Code Set Support

Single- and multibyte character code sets are supported.

EXAMPLES

Enable printer **snowwhite** to accept requests:

```
enable snowwhite
```

Deactivate printer **snowwhite** and cancel any logged jobs:

```
disable -c snowwhite
```

WARNINGS

If the restrict cancel feature (selected by the **lpadmin -orc** option — see *lpadmin*(1M)) is enabled, **disable** ignores the **-c** option.

enable and **disable** perform their operation on the local system only.

FILES

/etc/lp/*
/usr/lib/lp/*
/var/adm/lp/*
/var/spool/lp/*

SEE ALSO

lp(1), lpstat(1), accept(1M), lpadmin(1M), lpsched(1M), rcancel(1M), rlp(1M), rlpdaemon(1M), rlpstat(1M).

NAME

env - set environment for command execution

SYNOPSIS

env [**-**] [**-i**] [*name* **=** *value*] ... [*command* [*arguments* ...]]

DESCRIPTION

env obtains the current *environment*, modifies it according to its arguments, then executes the command with the modified environment. Arguments of the form *name***=***value* are merged into the inherited environment before the command is executed. The **-i** option causes the inherited environment to be ignored completely so that the command is executed with exactly the environment specified by the arguments. The **-** option is obsolete and has the same effect as the **-i** option.

If no command is specified, the resulting environment is printed, one name-value pair per line.

RETURN VALUE

If *command* is invoked, the exit status of **env** is the exit status of *command*; otherwise, **env** exits with one of the following values:

0	**env** completed successfully.
1-125	**env** encountered an error.
126	*command* was found but could not be invoked.
127	*command* could not be found.

EXTERNAL INFLUENCES

Environment Variables

LC_MESSAGES determines the language in which messages are displayed.

If **LC_MESSAGES** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**.

If any internationalization variable contains an invalid setting, **env** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported.

WARNING

The **-** option is obsolete. Use **-i** instead.

SEE ALSO

sh(1), exec(2), profile(4), environ(5).

STANDARDS CONFORMANCE

env: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

NAME

eucset - sets and gets EUC code widths for ldterm

SYNOPSIS

eucset [**-p**]

eucset [[**-c** *HP15-codeset*] or [*cswidth*]]

DESCRIPTION

The **eucset** command sets or gets (reports) the encoding and display widths of the Extended UNIX Code (EUC) characters processed by the current input terminal. EUC is an encoding method for codesets composed of single or multiple bytes. It permits applications and the terminal hardware to use the 7-bit US ASCII code and up to three single byte or multibyte code sets simultaneously.

The **eucset** command without any options, first tries to set the codeset to one of the four HP15 codesets. If unsuccessful, 7-bit US ASCII is used as the default codeset. This command must be used to specify any other EUC codesets, whether they are single byte or multibyte. See the WARNINGS section, for special warnings on the values of the *cswidth* argument.

Options

The **eucset** command recognizes the following options and arguments:

-p	Displays the current settings of the EUC character widths for the terminal.
-c	Sets the codeset to one of the four HP15 codesets. The codesets supported are **SJIS**, **CCDC**, **GB**, and **BIG5**.

EUC Code Set Classes

EUC divides codesets into four classes. Each codeset has two characteristics: the number of bytes for encoding the characters in the codeset, and the number of display columns to display the characters in the codeset. All characters within a codeset possess the same characteristics.

- Codeset 0 consists of all 7-bit, single byte ASCII characters. The most significant bit of each of these characters is 0 (zero). Characters in codeset 0 require one byte for encoding, and occupy one display column. These values are fixed for codeset 0 (zero). The 7-bit US ASCII code is the primary EUC codeset, which is available to users without direct specification.
- Codeset 1 is a supplementary EUC codeset. Codeset 1 characters have an initial byte whose most significant bit is 1. Characters in codeset 1 may require more than one byte for encoding, and may require more than one display column. The **eucset** command must be used to set the characteristics for codeset 1.
- Codesets 2 and 3 are supplementary EUC codesets. Characters in these codesets have an initial byte of SS2 or SS3, respectively. They require more than one byte for encoding, and may require more than one display column. The **eucset** command must be used to set the characteristics for codesets 2 and 3.

The *cswidth* argument in the **eucset** command line is a character string that describes the character widths for codesets 1 through 3. This command does not allow the user to modify the settings for codeset 0. The character string is of the following format:

X1[**:***Y1*]**,***X2*[**:***Y2*]**,***X3*[**:***Y3*]

The value *X1* is the number of bytes required to encode a character in codeset class 1. *Y1* is the number of display columns needed to display characters in this class. *X2* is the number of bytes required to encode a character in codeset 2, not counting the SS2 byte, and *Y2* is the number of display columns for codeset 2 characters. *X3* is the number of bytes needed to encode characters in codeset 3, not counting the SS3 byte, and *Y3* is the number of display columns required for these characters. The values for the column widths may be omitted if they are equal to the number of encoding bytes. If the encoding value of any of the EUC codesets is set to **0** (zero), this indicates that the codeset does not exist. See the WARNINGS section for special warnings on the values of the *cswidth* argument.

If no *cswidth* argument is supplied, the **eucset** command uses the value of the **CSWIDTH** environment variable. If this variable is not present, the following default string is substituted:

```
1:1,0:0,0:0
```

This default string designates that the environment uses a single byte EUC codeset that has characters in the EUC codeset 1 format. If the environment uses a multibyte EUC codeset in the codeset 1 format, single byte or multibyte EUC codesets in the codeset 2 or 3 format, or both, the default setting cannot be used.

EXTERNAL INFLUENCES

Environment Variables

LANG	Provide a default value for the internationalization variables that are unset or null. If **LANG** is not specified or is set to the empty string, a default of **C** (see *lang*(5)) is used instead of **LANG**. If any of the internationalization variables contain an invalid setting, **eucset** behaves as if all internationalization variables are set to **C**. See *environ*(5).
LC_ALL	If set to a nonempty string value, override the values of all other internationalization variables.
LC_MESSAGES	Determines the locale that should be used to affect the format and contents of diagnostic messages written to standard error and informative messages written to standard output.
NLSPATH	Determines the location of message catalogs for the processing of **LC_MESSAGES**.

e

EXAMPLES

To display the encoding and display widths for the EUC codesets 1 to 3 in your environment, enter:

```
eucset -p
```

Assuming **eucset** has been previously used to set for **ja_JP.eucJP**, the entry generates the following:

```
cswidth 2:2,1:1,2:2
```

To change the current settings of the encoding and display widths for the EUC characters in codesets 1 and 2 to two bytes each, enter one of the following:

```
eucset 2:2,2:2,0:0
eucset 2,2,0
```

To set the encoding and display widths for the EUC characters in the locale **ja_JP.eucJP**, enter:

```
eucset 2:2,1:1,2:2
```

For **zh_TW.eucTW**, enter:

```
eucset 2:2,3:2
```

For **ko_KR.eucKR**, enter:

```
eucset 2:2
```

WARNINGS

The *cswidth* argument does not include the SS2 or SS3 bytes in the byte width values.

This command is not specified by standards, may not be available on other vendor's systems, and may be subject to change or obsolescence in a future release.

AUTHOR

eucset was developed by OSF and HP.

SEE ALSO

dtterm(1), ldterm(1).

NAME

ex, edit - extended line-oriented text editor

SYNOPSIS

ex [**-**] [**-l**] [**-r**] [**-R**] [**-t** *tag*] [**-v**] [**-w***size*] [**-x**] [**-C**] [**+***command*] [*file* ...]

XPG4 Synopsis

ex [**-rR**] [**-s** | **-v**] [**-c** *command*] [**-t** *tag*] [**-w** *size*] [*file* ...]

Obsolescent Options

ex [**-rR**] [**-**| **-v**] [+*command*] [**-t** *tag*] [**-w** *size*] [*file* ...]

edit [**-**] [**-l**] [**-r**] [**-R**] [**-t** *tag*] [**-v**] [**-w***size*] [**-x**] [**-C**] [**+***command*] [*file* ...]

Remarks

The program names **ex**, **edit**, **vi**, **view**, and **vedit** are separate personalities of the same program. This manual entry describes the behavior of the **ex**/**edit** personality. On many HP-UX and other similar systems, **e** is a synonym for **ex**.

DESCRIPTION

The **ex** program is the line-oriented personality of a text editor that also supports screen-oriented editing (see *vi*(1)).

(XPG4 only.) Certain block-mode terminals do not have all the capabilities necessary to support the complete **ex** definition, such as the full-screen editing commands (**visual** mode or **open**mode). When these commands cannot be supported on such terminals, this condition shall neither produce an error message such as "not an editor command" nor report a syntax error.

The **edit** program is identical to **ex**, except that some editor option defaults are altered to make the editor somewhat friendlier for beginning and casual users (see Editor Options below).

Options and Arguments

ex recognizes the following command-line options and arguments:

- (Obsolescent) Suppress all interactive-user feedback. This is useful when editor commands are taken from scripts.

-s (XPG4 only.)

Suppress all interactive-user feedback. This is useful when editor commands are taken from scripts.

Ignore the value of the **TERM** and any implementation terminal type and assume the terminal is a type incapable of supporting visual mode.

Suppress the use of the **EXINIT** environment variable and the reading of the **.exrc** file.

-l Set the **lisp** editor option (see Editor Options below).

-r Recover the specified *file*s after an editor or system crash. If no *file* is specified, a list of all saved files is printed. You must be the owner of the saved file in order to recover it (superuser cannot recover files owned by other users).

-R Set the **readonly** editor option to prevent overwriting a file inadvertently (see Editor Options below).

-t *tag* (XPG4 only.) Edit the file containing the specified *tag* and proceed as if the first command were **:tag** *tag*. The tags represented by the **-t** *tag* and the **ta** command is optional. It shall be provided on any system that also provides a confirming implementation of **ctags**, Otherwise, the use of the **-t** produces undefined results.

Execute the **tag** *tag* command to load and position a predefined file. See the **tag** command in Command Descriptions and the **tags** editor option in Editor Options below.

-v Invoke visual mode (**vi**).

-w *size* Set the value of the **window** editor option to *size* (see Editor Options below). If *size* is omitted, it defaults to **3**.

-x	Set encryption mode. You are prompted for a key to initiate the creation or editing of an encrypted file (see the **crypt** command in Command Descriptions below).
-C	Encryption option. Same as the **-x** option, except that all text read in is assumed to have been encrypted.
-c *command*	(XPG4 only.)
+*command*	(Obsolescent) Begin editing by executing the specified **ex** search or positioning *command*.
file	Specify the file or files to be edited. If more than one *file* is specified, they are processed in the order given. If the **-r** option is also specified, the files are read from the recovery area.

(XPG4 only.) If both the **-t** *tag* and **-c** *command* options are given, the **-t** *tag* shall be processed first;i.e, the file containing the tag is selected by the **-t** and then the command is executed.

Definitions

Current file. The name of the file being edited by **ex** is called the current file. Text from the current file is read into a work area, and all editing changes are performed on this work area. Changes do not affect the original file until the work area is explicitly written back to the file. If the **%** character is used as a file name, it is replaced by the current file name.

Alternate file. The alternate file is the name of the last file mentioned in an editor command, or the previous current file name if the last file mentioned becomes the current file. If the **#** character is used as a file name, it is replaced by the alternate file name.

Buffers. Twenty-six buffers named **a** through **z** can be used for saving blocks of text during the edit. If the buffer name is specified in uppercase, text is appended to the existing buffer contents rather than overwriting it.

Readonly flag. The **readonly** flag can be cleared from within the editor by setting the **noreadonly** editor option (see Editor Options below). Writing to a different file is allowed even when the **readonly** flag is set. Also, a write can be forced to a **readonly** file by using **!** after the write command (see the **write** command in Command Descriptions below).

Interrupt. If an interrupt signal is received, and commands are being supplied from a keyboard, **ex** returns to command mode. If editor commands are coming from a file, an interrupt signal causes **ex** to abort.

System crash. If the system crashes or **ex** aborts due to an internal error or unexpected signal, **ex** attempts to preserve the work area if any unwritten changes were made. Use the **-r** command-line option to retrieve the saved changes.

Command mode/input mode. **ex** starts up in command mode, as indicated by the colon (**:**) prompt. **ex** switches to input mode whenever an **append**, **change**, or **insert** command is encountered. To terminate input mode and return to command mode, type a period (**.**) alone at the beginning of a line.

Comments. Command lines beginning with a quotation mark (**"**) are ignored (this is useful for placing comments in an editor script).

Multiple commands can be combined on a single line by separating them with a vertical bar character (**|**). However, global commands, comments, and the shell escape command must be the last command on a line because they cannot be terminated by a **|** character.

Addressing

(XPG4 only.) Addressing in **ex** relates to the current line. In general, the current line shall be the last line affected by the command; the exact effect on the current line is discussed under the description of each command. When the buffer contains no lines, the current line shall be set to zero.

ex recognizes the following line address forms:

.	Dot or period (**.**) refers to the current line. There is always a current line whose position can be the result of an explicit movement command or the result of a command that affects multiple lines (in which case it is usually the last line affected).
n	The *n*th line in the work area. Lines are numbered sequentially, starting at line 1.
$	The last line in the work area.
%	Abbreviation for **1,$**, meaning the entire work area.

+*n*, **+**[**+**]... **-***n*, **-**[**-**]...	An offset relative to the current line or the preceding line specification. **+** means forward; **-** means backward. For example, the forms **.+3**, **+3**, and **+++** are equivalent.
/*re***/** **?***re***?**	The line containing the pattern *re*, scanning forward (**/**) or backward (**?**). The trailing **/** or **?** can be omitted if the line is only being displayed. If *re* is omitted, **ex** uses the more recently set of either the scanning string or the substitution string (see Regular Expressions below).
'*x*	Lines can be marked using single lowercase letters (see the **mark** command in Command Descriptions below). **'***x* refers to the line marked with *x*. In addition, the previous current line is marked before each nonrelative motion. This line can be referred to by using **'** for *x* (thus **''** refers to the previous current line). (XPG4 only.) Commands require zero, one or two addresses. Commands that require zero addresses shall regard the presence of an address as an error.

(XPG4 only.) Adjacent address in a **range** shall be separated from each other by a comma (,) or a semicolon(;). In the latter case, the current line(.) shall be set to the first address, and only then is the second address calculated. This feature can be ued to determine the starting line for forwards and backwards searches. The second address of any two-address sequence shall correspond to the first address. The first address shall be less than or equal to the second address. The first address shall be greater than or equal to the first line of the editing buffer, and the last address shall be less than or equal to the last line of the editing buffer. Any other case shall be an error.

Addresses for commands consist of a series of line addresses (specified as above), separated by a comma (**,**) or semicolon (**;**). Such address lists are evaluated left-to-right. When the separator is a semicolon, the current line is set to the value of the previous address before the next address is interpreted. If more addresses are given than the command requires, then all but the last one or two are ignored. Where a command requires two addresses, the first line addressed must precede the second one in the work area. A null (missing) address in a list defaults to the current line.

Regular Expression

The editor maintains copies of two regular expression strings at all times: the substitution string, and the scanning string. The substitute command sets the substitution string to the regular expression used. Both the global-command and the regular-expression form of line addressing (see Addressing above) for all commands set the scanning string to the regular expression used. These strings are used as default regular expressions as described under Addressing, the **global** command, and the **substitute** command.

The editor supports Basic Regular Expressions (see *regexp*(5)) with the following modifications:

\<	The **\<** matches the beginning of a "word"; that is, the matched string must begin in a letter, digit, or underline, and must be preceded by the beginning of the line or a character other than the above. This construct can only be used at the beginning of a regular expression (as in *\<word*), but not in the middle (*word1 \<word2*).
\>	The **\>** matches the end of a "word" (see previous paragraph). This construct can only be used at the end of a regular expression (as in *word\>*), but not in the middle (*word1\> word2*).
~	Match the replacement part of the last **substitute** command.
[*string***]**	The positional quoting within bracket expressions defined by Basic Regular Expressions is replaced by the use of the backslash (****) to quote bracket-expression special characters.
nomagic	When the editor option **nomagic** is set, the only characters with special meanings are **^** at the beginning of a pattern, **$** at the end of a pattern, and \. The characters ., *, **[**, and **~** lose their special meanings unless escaped by a ****.

Replacement Strings

The character **&** in the replacement string stands for the text matched by the pattern to be replaced. Use **\&** if the **nomagic** editor option is set.

The character **~** is replaced by the replacement part of the previous **substitute** command. Use **\~** if the **nomagic** editor option is set.

The sequence *****n*, where *n* is an integer, is replaced by the text matched by the subpattern enclosed in the *n*th set of parentheses **\(** and **\)**.

The sequence `\u` (`\l`) causes the immediately following character in the replacement to be converted to uppercase (lowercase), if the character is a letter. The sequence `\U` (`\L`) turns case conversion on, until the sequence `\E` or `\e` is encountered, or the end of the replacement string is reached.

Command Names and Abbreviations

The following table summarizes the line-mode commands. The commands whose names are enclosed in parentheses are available only in their abbreviated forms.

Command	Abbr.	Command	Abbr.	Command	Abbr.
`abbreviate`	`ab`	`next`	`n`	`tag`	`ta`
`append`	`a`	`number`	`nu #`	`unabbreviate`	`una`
`args`	`ar`	`open`	`o`	`undo`	`u`
`change`	`c`	`pop`		`unmap`	`unm`
`chdir`	`chd cd`	`preserve`	`pre`	`version`	`ve`
`copy`	`co t`	`print`	`p`	`visual`	`vi`
`crypt`	`cr X`	`put`	`pu`	`write`	`w wq`
`delete`	`d`	`quit`	`q`	`xit`	`x`
`edit`	`e ex`	`read`	`r`	`yank`	`ya`
`file`	`f`	`recover`	`rec`		
`global`	`g v`	`rewind`	`rew`	(execute buffer)	`* @`
`insert`	`i`	`set`	`se`	(line number)	`=`
`join`	`j`	`shell`	`sh`	(left shift)	`<`
`list`	`l`	`source`	`so`	(right shift)	`>`
`map`		`stop`	`st ^Z`	(scroll)	`^D`
`mark`	`ma k`	`substitute`	`s sr & ~`	(shell escape)	`!`
`move`	`m`	`suspend`	`su ^Z`	(window)	`z`

Command Descriptions

In the following command descriptions, some arguments appear frequently. They are described below.

line A single line address, in any of the forms described in Addressing above. The default is the current line.

range A pair of line addresses separated by a comma or semicolon, as described in Addressing above. The default is the current line (`.,.`).

count A positive integer specifying the number of lines to be affected by the command. The default is 1 or the number of lines in *range*.

When *count* is specified, *range* is ineffective. Instead, only a line number should be specified to indicate the first line affected by the command. (If a range is given, the last line of the range is interpreted as the starting line for the command.)

flags One or more of the characters `#`, `p`, and `l`. The corresponding command to print the line is executed after the command completes. Any number of `+` or `-` characters can also be given with these flags. The default is no flags.

These modifiers are all optional.

When only a *line* or a *range* is specified (with a null command), the implied command is `print`. If a null line is entered, the next line is printed (equivalent to `.+1p`)

buffer *XPG4 Feature*. One of a number of named areas for saving text. The named buffers are specified by the lowercase letters of the POSIX locale. Specifying `buffer` shall cause the area of the text affected by the command to be stored into the buffer as it was before the command took effect. This argument is also used on the `put` command and the visual mode "put" commands (`p` and `P`), to specify the buffer that shall provide the text to insert.

If the buffer name is specified in uppercase, and the buffer is to be modified, the buffer shall be appended to rather than being overwritten. If the buffer is not to be modified, the buffer name can be specified in lowercase or uppercase with the same results. There shall be also one unnamed buffer, which is the repository for all text deleteed or yanked when no buffer is specified.

There are also numbered buffers, 1 through 9, which shall be accessible only from visual mode. These buffers are special in that, in the visual mode, when deleted text is placed in the unnamed buffer, it also shall be placed in buffer 1, the previous contents buffer 1 shall be placed in buffer 2 and so on. Any text in the buffer 9 shall be lost. Text that is yanked into the unnamed buffer shall not modify the numbered buffers. Text cannot be placed directly into the numbered buffered, although it can be retrieved from them by using a visual mode "put" command with the buffer name given as s number. When the buffer modifier is not used in the commands below, the unnamed buffer shall be the default.

word *XPG4 Feature*. In the POSIX Locale, a **word** consists of a maximal sequence of letters, digits and underscores, delimited at both ends by characters other than letters, digits, or underscores, or by the beginning or end of a word or the file. **!** A character that can be appended to the command to modify its operation, as detailed in the individual command descriptions.

If both a **count** and **range** is specified for a command that uses them, the number of lines affected shall be taken from the **count** value rather than the **range**. The starting line for the command shall be taken to be the first line addressed by the range.

When only a **line** or **range** is specified with no command, the implied command shall be either **print**, **list**, or **number** (**p**, **l**, or **#**). The command selected shall be the last of these three commands to be used. When no range or count is specified and the command line is a blank line, the current line shall be written, and the current line shall be set to **.+1**.

Zero or mode <blank> characters can precede or follow the addresses, count or command name. Any object following a command name (such as buffer, file etc) that begins with an alphabetic character shall be separated from the command name with at least one <blank>.

For each of the commands listed below, the command can be entered as the abbreviation (those characters in the Synopsis command word preceding the []), the full command (all characters shown for the command word, omitting the [and]), or any subset of the characters of the full command down to the abbreviation.

abbreviate **ab**[**breviate**] *word replacement*

Add the named abbreviation to the current list. In visual mode, if *word* is typed as a complete word during input, it is replaced by the string *replacement*.

append *line* **a**[**ppend**][**!**]

Enter input mode; the input text is placed after the specified line. If line 0 is specified, the text is placed at the beginning of the work area. The last input line becomes the current line, or the target line if no lines are input.

Appending **!** to the command toggles the **autoindent** editor option setting for this insert only.

args **ar**[**gs**]

Prints the argument, placing the current argument between **[** and **]**.

change *range* **c**[**hange**][**!**] *count*

Enter input mode; the input text replaces the specified lines. The last input line becomes the current line; if no lines are input, the effect is the same as a delete.

Appending **!** to the command toggles the **autoindent** editor option setting for this insert only.

chdir **chd**[**ir**][**!**] [*directory*]
cd[**!**] [*directory*]

Change the working directory to *directory*. If *directory* is omitted, the value of the **HOME** environment variable is used. If the work area has been modified since the last write and the name of the file being edited does not begin with a slash (**/**), a warning is issued and the working directory is not changed. To force a change of directory in this case, append the character **!** to the command.

copy *range* **co**[**py**] *line flags*
range **t** *line flags*

A copy of the specified lines (*range*) is placed after the specified destination *line*; line 0 specifies that the lines are to be placed at the beginning of the work area. (The letter **t** is an alternative abbreviation for the **copy** command.)

crypt

cr[**ypt**]
X

The user is prompted for a key with which to enter encryption mode. This command can also be used to change the key entered from a previous **crypt** command or the **-x** command line option. If no key is supplied in response to the prompt (that is, only carriage return is pressed), encryption mode is canceled and the work area is written out in plain-text form by subsequent write commands.

While in encryption mode, all file input is decrypted using the current key. However, while an input file is being processed, if a block of text (approximately 1024 bytes) is encountered that contains only 7-bit ASCII characters, that block of text is assumed to be plain-text and is not decrypted. All file output, except that piped via a **!** shell escape to another command, is encrypted using the current key.

The temporary file used by the editor to manage the work area is not encrypted until the current work area is discarded (or written out) and editing begins on a new file. When creating a new file that requires encryption protection, ensure that the work area file is also encrypted by specifying the **-x** option when invoking the editor.

cr[**ypt**]
C

Encryption option. Same as the **X** command, except that all text read in is assumed to have been encrypted.

delete

range **d**[**elete**] *buffer count*

The specified lines are deleted from the work area. If a named *buffer* is specified, the deleted text is saved in it. If no buffer is specified, the unnamed buffer is used (that is, the buffer where the most recently deleted or yanked text is placed by default). The new current line is the line after the deleted lines or the last line of the file if the deleted lines were at the end of the file.

edit

e[**dit**][**!**] [**+** *line*] *file*
ex[**!**] [**+** *line*] *file*

Begin editing a new file (**ex** is an alternative name for the **edit** command). If the current work area has been modified since the last write, a warning is printed and the command is aborted. This action can be overridden by appending the character **!** to the command (**e!** *file*). The current line is the last line of the work area unless it is executed from within *vi*, in which case the current line is the first line of the work area. If the **+***line* option is specified, the current line is set to the specified position, where *line* can be a number (or **$**) or specified as **/***re* or **?***re*.

file

f[**ile**]

Print the current file name and other information, including the number of lines and the current position.

global

range **g**[**lobal**][**!**] **/***re***/** *command*...
range **v** **/***re***/** *command*...

Perform *command* on lines within *range* (or on the entire work area if no *range* is given) that contain *re*. First mark the lines within the given *range* that match the pattern *re*. If the pattern is omitted, the more recently set of either the substitution string or the scanning string is used (see Regular Expressions above). Then the given *command*s are executed with **.** set to each marked line. Any character other than a letter or a digit can be used to delimit the pattern instead of the **/**.

command can be specified on multiple lines by hiding new-lines with a backslash. If *command* is omitted, each line is printed. **append**, **change**, and **insert** commands are allowed; the terminating dot can be omitted if it ends *command* or *command*s. The **visual** command is also permitted (unless the **global** command itself has been issued from visual mode), and takes input from the terminal. (If *command* contains a visual-mode command (that is, **open** or **visual**), the visual-mode command must be terminated by the visual-mode **Q** command in order to proceed to the next marked line.)

The **global** command itself and the **undo** command are not allowed in *command*. The editor options **autoprint**, **autoindent**, and **report** are inhibited.

Appending a **!** to the **global** command (that is, **g!** ...) or using the alternate name **v** causes *command* to be run on the lines within *range* that do not match the pattern.

insert

line **i**[**nsert**][**!**]

Enter input mode; the input text is placed before the specified line. The last line input becomes the current line, or the line before the target line, if no lines are input.

Appending **!** to the command toggles the **autoindent** editor option setting for this insert only.

join

range **j**[**oin**][**!**] *count flags*

Join together the text from the specified lines into one line. White space is adjusted to provide at least one blank character (two if a period appears at the end of a line, or none if the first character of a line is a closing parenthesis (**)**)). Extra white space at the beginning of a line is discarded.

Appending a **!** to the command causes a simpler join with no white-space processing.

list

range **l**[**ist**] *ount flags*

Print the specified lines with tabs displayed as **^I** and the end of each line marked with a trailing **$**. (The only useful flag is **#** for line numbers.) The last line printed becomes the current line.

map

map key|**#***n action*
map! key|**#***n action*

The **map** and **map!** commands define macros for use in visual mode. The first argument, *key*, can be a single character or a multicharacter sequence. In the special sequence, **#***n*, *n* is a digit referring to the function key *n*. Special characters, whitespace, and newline must be escaped with a **^V** to be entered in the arguments. The *key* argument cannot contain a colon (**:**) as its first character, nor can a multicharacter sequence begin with an alphabetic character.

Macros defined by **map** are effective in visual command mode. Macros defined by **map!** are effective in visual input mode. When *key* or the function key corresponding to **#***n* is entered, the editor interprets the operation as though *action* were typed.

The **map** or **map!** command without options displays the corresponding current list of macros.

See also the editor options **keyboardedit**, **keyboardedit!**, **timeout**, and **timeoutlen** in Editor Options below.

mark

line **ma**[**rk**] *x*
line **k** *x*

The specified line is given the specified mark *x*, which must be a single lowercase letter (**a**-**z**). x must be preceded by a space or tab. The current line position is not affected. **k** is an alternate name for **mark**.

move

range **m**[**ove**] *line*

Move the specified lines (*range*) to follow the target *line*. The first line moved becomes the current line.

next

n[**ext**][**!**] [*file* ...]

The next file from the command line argument list is edited. Appending a **!** to the command overrides the warning about the work area having been modified since the last write (and discards any changes unless the **autowrite** editor option is set). The argument list can be replaced by specifying a new one on this command line.

number

range **nu**[**mber**] *count flags*
range **#** *count flags*

(The **#** character is an alternative abbreviation for the **number** command.) Print the lines, each preceded by its line number (the only useful flag is **l**). The last line printed becomes the current line.

open *line* **o**[**pen**] **/***re***/** *flags*

Enter open mode, which is similar to visual mode with a one-line window. All the visual-mode commands are available. If a match is found in *line* for the optional regular expression, the cursor is placed at the start of the matching pattern. Use the visual mode command **Q** to exit from open mode. For more information, see *vi*(1).

pop **pop**[**!**]

Load the file whose name is stored at the top of the tag stack and set the current line to the stored location. The top entry of the tag stack is deleted. (The current file name is placed on the stack when you execute the line mode **tag** command or the visual mode **^]** command.)

! overrides the warning about the work area having been modified since the last write; any changes are discarded unless the **autowrite** editor option is set).

preserve **pre**[**serve**]

The current editor work area is saved as if the system had just crashed. Use this command in emergencies, for example when a write does not work and the work area cannot be saved in any other way. Use the **-r** command-line option to recover the file. After the file has been preserved, a mail message shall be sent to the user. The message shall contain the name of the file, the time of preservation and an **ex** command that could be used to recover the file. Additional information may be included in the mail message.

print **range p**[**rint**] *count*

Print the specified lines, with non-printing characters printed as control characters in the form **^x**; DEL is represented as **^?**. The last line printed becomes the current line.

put *line* **pu**[**t**] *buffer*

Place deleted or "yanked" lines after *line*. A buffer can be specified; otherwise, the text in the unnamed buffer (that is, the buffer in which deleted or yanked text is placed by default) is restored. The current line indicator shall be set to the first line put back.

quit **q**[**uit**][**!**]

Terminate the edit. If the work area has been modified since the last write, a warning is printed and the command fails. To force termination without preserving changes, append **!** to the command.

read *line* **r**[**ead**] *file*

Place a copy of the specified *file* in the work area after the target line (which can be line 0 to place text at the beginning). If no *file* is named, the current file is the default. If no current file exists, *file* becomes the current file. The last line read becomes the current line except in visual mode where the first line read becomes the current line.

If **file** is given as **!***string*, *string* is interpreted as a system command and passed to the command interpreter; the resultant output is read into the work area. A blank or tab must precede the **!**.

recover **rec**[**over**][**!**] *file*

Recover *file* from the save area, after an accidental hangup or a system crash. If the current work area has been modified since the last write, a warning is printed and the command is aborted. This action can be overridden by appending the character **!** to the command (**rec!** *file*).

rewind **rew**[**ind**][**!**]

The argument list is rewound, and the first file in the list is edited. This shall be equivalent to a **next** command with the current argument list as its operands. If the current buffer has been modified since the last write, a warning shall be written and the command shall be aborted. Any warnings can be overridden by appending a **!**. The current indicator line shall be affected by the editor options, **autowrite** and **writeany**.

set

se[**t**] [**all**]
se[**t**] [**no**]*boolean-option***?**
se[**t**] *value-option*[**?**]
se[**t**] *boolean-option*
se[**t**] **no***boolean-option*
se[**t**] *value-option***=***value*

Set and display the values of the editor options (see Editor Options below).

With no arguments, the command prints those editor options whose values have been changed from the default settings. If **all** is specified, it prints all current option values.

The second and third forms display the current value of the specified option. The **?** is necessary only for Boolean options.

The fourth form turns a Boolean option on. The fifth form turns a Boolean option off.

The sixth form assigns values to string and numeric options. Spaces and tabs in strings must be escaped with a leading backslash (****).

The last five forms can be combined; interpretation is left-to-right.

shell

sh[**ell**]

Execute the command interpreter specified by the **shell** editor option (see Editor Options below). Editing is resumed when you exit from the command interpreter.

source

so[**urce**] *file*

Read and execute commands from the specified *file*. **so** commands can be nested. The maximum supported nesting depths is implementation defined, but shall be at least one.

substitute

range **s**[**ubstitute**] **/***re***/***repl***/** *options count flags*
range **s** *options count flags*
range **&** *options count flags*
range **sr** *options count flags*
range **~** *options count flags*
range **s\?***repl*
range **s\&***repl*

On each specified line, the first instance of the pattern *re* is replaced by the string *repl*. (See Regular Expressions and Replacement Strings above.) Any character other than a letter or a digit can be used to delimit the pattern instead of the **/**.

If you include the **g** (global) option, all instances of the pattern in the line are substituted.

If you include the **c** (confirm) option, you are queried about whether to perform each individual substitution, as follows: Before each substitution the line is displayed with the pattern to be replaced marked underneath with carets (**^**). Type **y** to cause the substitution to be performed; any other input to abort it. The last line substituted becomes the current line.

If the substitution pattern *re* is omitted (**s//***repl***/**), the more recently set of either the substitution string or the scanning string is used (see Regular Expressions above).

If the **s** or **&** forms of the command are used, the substitution pattern defaults to the previous substitution string and the replacement string defaults to the previous replacement string used.

If the **sr** or **~** forms of the command are used, the substitution pattern defaults to the more recently set of either the substitution string or the scanning string and the replacement string defaults to the previous replacement string used.

The form **s\?***repl* is equivalent to **s/***scan-re***/***repl***/**, where *scan-re* is the previous scanning string.

The form **s\&***repl* is equivalent to **s/***subs-re***/***repl***/**, where *subs-re* is the previous substitution string.

suspend **su**[**spend**][**!**]
stop **st**[**op**][**!**]
susp

Suspend the editor job and return to the calling shell. **stop** and *susp* are equivalent to **suspend**. *susp* is the user process control suspend character, which is typically the character **^Z** (ASCII SUB) (see *stty*(1)). This command is disabled if the calling shell does not support job control or has disabled it.

The work area is written to the current file before the editor is suspended if the **autowrite** editor option is set, the **readonly** editor option is not set, and the work area has been modified since the last write. To override this action, append the **!** character to the **suspend** or **stop** command.

tag **ta**[**g**][**!**] *tag*

Search the files specified by the **tags** editor option (see Editor Options below) sequentially until a tag definition for *tag* is found. If *tag* is found, load the associated file into the work area and set the current position to the address specified in the tag definition.

The work area is written to the current file before the new file is loaded if the new file is different from the current file, the **autowrite** editor option is set, the **readonly** editor option is not set, and the work area has been modified since the last write. To override this action, append the **!** character to the command.

If the **tagstack** editor option is set, the current file name and line number is pushed onto the tag stack for later recall with the line mode **pop** command or the visual mode **^]** command.

unabbreviate **una**[**bbreviate**] *word*

Delete *word* from the list of abbreviations (see the **abbreviate** command above).

undo **u**[**ndo**]

Reverse the changes made by the previous editing command. For this purpose, **global** and **visual** are considered single commands. Commands that affect the external environment, such as **write**, **edit**, and **next**, cannot be undone. An **undo** can itself be reversed.

unmap **unm**[**ap**][**!**] *key*

The macro definition for *key* is removed (see the **map** command above).

version **ve**[**rsion**]

Print the current version information for the editor.

visual *line* **vi**[**sual**] *type count flags*

Enter visual mode at the specified *line*.

The *type* can be one of the characters **+**, **-**, **.**, or **^**, as in the **z** (window) command, to specify the position of the specified line on the screen window The default is to place the line at the top of the screen window.

A *count* specifies an initial window size; the default is the value of the editor option **window**.

The flags **#** and **l** (ell) cause the lines in the visual window to be displayed in the corresponding mode (see the **number** and **list** commands).

Use the **Q** command to exit visual mode. For more information, see *vi*(1).

write [*range*] **w**[**rite**][**!**][**>>**] *file*
[*range*] **wq**[**!**][**>>**] *file*

Write the specified lines (or the entire work area, if no *range* is given) out to *file*, printing the number of lines and characters written. If *file* is not specified, the default is the current file (the command fails with an error message if there is no current file and no file is specified).

If an alternate file is specified and the file exists, the write fails, but can be forced by appending **!** to the command. To append to an existing file, append **>>** to the command. If the file does not exist, an error is reported.

If the file is specified as **!***string*, *string* is interpreted as a system command, the command interpreter is invoked, and the specified lines are passed as standard input to the command.

The command **wq** is equivalent to a **w** followed by a **q**. **wq!** is equivalent to **w!** followed by **q**. **wq>>** is equivalent to **w>>** followed by **q**.

xit **x**[**it**][**!**][**>>**] *file*

If changes have been made to the work area, a **write** command is executed with any options (such as **!**, **>>**, or *file*) used by the **write** command. Then (in any case) the **quit** command is executed.

yank *range* **ya**[**nk**] *buffer count*

Place the specified lines in the named *buffer*. If no buffer is specified, the unnamed buffer is used (that is, the buffer where the most recently deleted or yanked text is placed by default).

(execute buffer) ***** [*buffer*]
@ [*buffer*]

Execute the contents of *buffer* as an editor command. *buffer* can be the letter of a named buffer (**a**–**z**) or ***** or **@**. The ***** and the **@** forms of this command are equivalent. If a buffer is not specified or *buffer* is ***** or **@**, the buffer last named in a ***** or **@** command is executed.

(line number) *line* **=** *flags*

Print the line number of the specified *line*. The default is the last line. The current line position is not affected.

(scroll) **^D**

Print the next *n* lines, where *n* is the value of the **scroll** editor option.

(shell escape) **!** *command*
range **!** *command*

Pass the remainder of the line after the **!** to the system command interpreter for execution. A warning is issued if the work area has been changed since the last write. A single **!** is printed when the command completes. The current line position is not affected.

Within the text of *command*, **%** and **#** are expanded as file names, and **!** is replaced with the text of the previous **!** command. Thus, **!!** repeats the previous **!** command. When such expansion is performed, the expanded line is echoed.

If you specify *range*, the specified lines are passed to the command interpreter as standard input. The output from the *command* replaces the specified lines.

(shift left) *range* **<** *count flags*

Shift the specified lines to the left. The number of spaces to be deleted is determined by the editor option **shiftwidth**. Only whitespace (blanks and tabs) is lost in shifting; other characters are not affected. The last line changed becomes the current line.

(shift right) *range* **>** *count flags*

Shift the specified lines to the right by inserting whitespace The number of spaces inserted is determined by the editor option **shiftwidth**. The last line changed becomes the current line.

(window) *line* **z** *type count flags*

The number of lines specified by *count* are displayed. The default for *count* is the value of the editor option **window**.

If *type* is omitted, *count* lines following the specified *line* are printed.

If *type* is specified, it must be one of the following characters:

+	Display a window of lines following the addressed line.
-	Place the addressed line at the bottom of the window of displayed lines.
.	Place the addressed line at the center of the window.
^	Display a window of lines that is two windows prior to the addressed line.
=	Display the addressed line at the center of the window with a line of dashes above and below the addressed line.

The last line printed becomes the current line, except for the **=**, where the addressed line becomes the current line.

Editor Options

The command **ex** has a number of options that modify its behavior. These options have default settings, which can be changed using the **set** command (see above). Options can also be set at startup by putting a **set** command string in the environment variable **EXINIT**, or in the file **.exrc** in the **HOME** directory, or in **.exrc** in the current directory. If **EXINIT** exists, the **.exrc** file in the **HOME** directory is not executed. If the current directory is not the **HOME** directory and the **exrc** editor option is set (see below), the **.exrc** file in the current directory is executed after **EXINIT** or the **HOME** directory **.exrc**.

The editor obtains the horizontal and vertical size of the terminal screen from the **terminfo** database (see *terminfo*(4)). These values can be overridden by setting the **UNIX95** environment variable, which specifies to use the XPG4 behavior for this command. **COLUMNS** and **LINES** environment variables. See the **window** editor option below for more information.

The following table shows the defaults that differ for the various editor personalities:

Name	Default Editor Options				
edit	**nomagic**	**novice**	**noreadonly**	**report=1**	**showmode**
ex	**magic**	**nonovice**	**noreadonly**	**report=5**	**noshowmode**
vedit	**nomagic**	**novice**	**noreadonly**	**report=1**	**showmode**
vi	**magic**	**nonovice**	**noreadonly**	**report=5**	**noshowmode**
view	**magic**	**nonovice**	**readonly**	**report=5**	**noshowmode**

Editor options are Boolean unless otherwise specified. Abbreviations are shown in parentheses.

autoindent (**ai**) Indent each line in input mode (using blanks and tabs) to align with the previous line. Indentation begins after the line appended, or before the line inserted or the first line changed. Additional indentation can be provided as usual. Succeeding lines are automatically indented to the new alignment.

Reducing the indent is achieved by typing **^D** one or more times: the cursor is moved back to the next multiple of **shiftwidth** spaces for each **^D**. A **^** followed by a **^D** removes all indentation temporarily for the current line. A **0** followed by a **^D** removes all indentation.

Reversed by **noautoindent** (**noai**). The default is **noautoindent**.

autoprint (**ap**) The current line is printed after each command that changes work area text. Autoprint is suppressed in **global** commands. Reversed by **noautoprint** (**noap**). The default is **autoprint**.

autowrite (**aw**) The work area is written out to the current file if the work area has been modified and a **next**, **rewind**, or **!** command is given. Reversed by **noautowrite** (**noaw**). The default is **noautowrite**.

beautify (**bf**) Cause all control characters other than tab, newline, and formfeed to be discarded from the input text. Reversed by **nobeautify** (**nobf**). The default is **nobeautify**.

directory=*dirname* (**dir**)

Specify the directory in which the editor work area should be placed. This option only takes effect when a new work area is created. It should be set in **EXINIT** or **.exrc** to affect the location of the work area file for the edit file specified on the command line. The default is **/var/tmp**.

If the specified directory is set from **EXINIT** or a **.exrc** file and is not writable by the user, the editor quits; if set interactively by the user, the editor issues an error message.

doubleescape When set, two consecutive ESC (escape) characters are required to leave input mode. In input mode, a single ESC character followed by a different character causes **vi** to issue an audible or visual warning (see the **flash** editor option) and insert both characters into the work area. Reversed by **nodoubleescape**. The default is **nodoubleescape**.

The character sequences transmitted by the keyboard editing keys of some terminals are identical to some sequences of **vi** user commands. If the mapping of these keys is enabled (see the **keyboardedit** and **keyboardedit!** options), **vi** might not be able to reliably distinguish between the character sequence transmitted by an editing key and the same character sequence typed by a user. This problem is most likely to occur when the user types ESC to terminate input mode immediately followed by another **vi** command. If you set the **doubleescape** option, the ambiguity of this case is removed.

edcompatible (**ed**)
Cause the presence of **g** and **c** suffixes on substitute commands to be remembered, and toggled by repeating the suffixes. Reversed by **noedcompatible** (**noed**). The default is **noedcompatible**.

errorbells (**eb**) When set, error messages are preceded with a bell only on terminals that do not support a standout or highlighting mode such as inverse video. If the terminal supports highlighting, the bell is never used prior to error messages and this option has no effect. Note that visual-mode errors are signaled by the bell (regardless of the setting of this option) without an accompanying error message.

Reversed by **noerrorbells** (**noeb**). The default is **noerrorbells**.

exrc When set, the **.exrc** file in the current directory is processed during editor initialization if the current directory is not the **HOME** directory. This option is not set by default and must be set in the **EXINIT** environment variable or the **HOME** directory **.exrc** file to have any effect. See the Editor Options introductory text above. Reversed by **noexrc**. The default is **noexrc**.

flash (**fl**) When set, the screen flashes instead of beeping, provided an appropriate **flash_screen** entry is present in the **/usr/share/lib/terminfo** database for the terminal being used. Reversed by **noflash** (**nofl**). The default is **flash**.

hardtabs=*number* (**ht**)
Define the spacing between hardware tab settings and the number of spaces used by the system when expanding tab characters. Tab stops are placed in each column number (starting at the left edge of the screen) that corresponds to an integer multiple of *number*. The default is **hardtabs=8**.

ignorecase (**ic**) All uppercase characters in the text are mapped to lowercase in regular expression matching. Also, all uppercase characters in regular expressions are mapped to lowercase, except in character class specifications. Reversed by **noignorecase** (**noic**). The default is **noignorecase**.

keyboardedit When set, any keyboard editing key mappings that are loaded automatically at initialization for command-mode use are enabled. If not set, these mappings are disabled (but not deleted). Use the **map** command to get a list of the currently enabled command-mode mappings. Reversed by **nokeyboardedit**. The default is **keyboardedit**.

keyboardedit! When set, the keyboard editing key mappings automatically loaded at initialization for input mode use are enabled. If not set, these mappings are disabled (but not deleted). Use the **map!** command to list the currently enabled input-mode mappings. Reversed by **nokeyboardedit!**. The default is **nokeyboardedit!** for terminals whose keyboard editing keys send HP-style escape sequences (an ESC followed by a single letter). The default is **keyboardedit!** for all other terminals.

lisp Modify **autoindent** mode and the **(**, **)**, **[[**, **]]**, **{**, and **}** commands in visual mode for **lisp** source code. Reversed by **nolisp**. The default is **nolisp**.

list Display all printed lines with tabs shown as **^I**, and the end of line marked by a **$**. Reversed by **nolist**. The default is **nolist**.

magic Affect the interpretation of characters in regular expressions and substitution replacement strings (see Regular Expressions and Replacement Strings above). Reversed by **nomagic**. The **ex**, **vi**, and **view** default is **magic**. The **edit** and **vedit** default is **nomagic**.

mesg Allows other users to use the **write** command (see *write*(1)) to send messages to your terminal, possibly disrupting the screen display. Unsetting this option (**nomesg**) blocks write permission to your terminal from other system users while you are using the editor. Reversed by **nomesg**. The default is **mesg**.

modelines (**ml**) If set when the editor reads in a file, any **ex** commands embedded in the first five and last five lines of the file are executed after **.exrc** and **EXINIT** commands are processed but before editing control is given to the user. The **ex** commands must be prefixed by **ex:** or **vi:** and terminated by **:** in a single line. Any number of other characters with the exception of the colon (**:**) can precede or follow the embedded command. Reversed by **nomodelines** (**noml**). The default is **nomodelines**.

novice Use the version of the editor available for novices, known as **edit** or **vedit**. Reversed by **nonovice**. The **ex**, **vi**, and **view** default is **nonovice**. The **edit**, and **vedit** default is **novice**.

number (**nu**) Cause lines to be printed with line numbers. Reversed by **nonumber** (**nonu**). The default is **nonumber**.

optimize (**opt**) Suppress automatic carriage returns on terminals that do not support direct cursor addressing. This streamlines text output in certain situations such as when printing multiple lines that contain leading whitespace. Reversed by **nooptimize** (**noopt**). The default is **nooptimize**.

paragraphs=*pair-string* (**para**)
The value of this option is a string whose successive pairs of characters specify the names of text-processing macros that begin paragraphs. (A macro appears in the text in the form **.***xx*, where the **.** is the first character in the line.)

If any macros have a single-character name, use a space character to substitute for the missing second character in the name. To type a space character in such situations, precede the space with a backslash (****) to prevent the editor from interpreting it as a delimiter.

The default is **paragraphs=IPLPPPQPP\ LIpplpipnpbp**.

prompt When set, command mode input is prompted for with a colon (**:**); when unset, no prompt is displayed. Reversed by **noprompt**. The default is **prompt**.

readonly (**ro**) Set the **readonly** flag for the file being edited, thus preventing accidental overwriting at the end of the session. This option is equivalent to invoking **ex**, **edit**, **vi**, or **vedit** with the **-R** option or using the **view** command. Reversed by **noreadonly** (**noro**). The **ex**, **edit**, **vi**, and **vedit** default is **noreadonly**. The **view** default is **readonly**.

redraw Simulate an intelligent terminal on a dumb terminal. During input mode, lines are continuously reprinted as text is entered. Since this is likely to require a large amount of output to the terminal, it is useful only at high transmission speeds. If **noredraw** is set, lines are reprinted only when input mode is terminated and deleted lines are marked with an **@** in the left margin. Reversed by **noredraw**. The default is **redraw**.

remap If set, macro translation allows for macros defined in terms of other macros; translation continues until the final product is obtained. If unset, a one-step translation only is done. Reversed by **noremap**. The default is **remap**.

report=*n* The value of *n* gives the number of lines that must be changed by a command before a report is displayed on the number of lines affected. If *n* is 5, then changes are reported for 6 or more lines. The **ex**, **vi**, and **view** default is **report=5**. The **edit**, and **vedit** default is **report=1**.

scroll=*n* The value of *n* determines the number of lines scrolled by a **^D** command and the number of lines displayed by the **z** command (twice the value of scroll). The default is half the value of the **window** option.

sections=*pair-string*
: The value of this option is a string, in that successive pairs of characters specify the names of text-processing macros that begin sections. See the **paragraphs** editor option above. The default is **sections=NHSHH\ HUuhsh+c**.

shell=*filename* (**sh**)
: Set the file name of the shell to be used for the **!** shell escape and the **shell** command. It defaults to the value of your **SHELL** environment variable, if set, and otherwise to **/usr/bin/sh**.

shiftwidth=*n* (**sw**)
: Sets the indentation step value used by **autoindent** and the shift (**<** and **>**) commands. The default is **shiftwidth=8**.

showmatch (**sm**)
: In visual mode, jump momentarily to the matching **(** or **{** when you type a **)** or **}**, if the match is still on the screen. Reversed by **noshowmatch** (**nosm**). The default is **noshowmatch**.

showmode (**smd**)
: Display the current editor mode (such as **INPUT MODE**, **REPLACE 1 CHAR**, **REPLACE MODE**) in the lower right-hand corner of the screen during visual and open mode. Reversed by **noshowmode** (**nosmd**). The **ex**, **vi**, and **view** default is **noshowmode**. The **edit**, and **vedit** default is **showmode**.

slowopen (**slow**)
: In visual mode, **slowopen** prevents screen updates during input to improve throughput on unintelligent terminals. Reversed by **noslowopen** (**noslow**). The default is **noslowopen**.

tabstop=*n* (**ts**)
: Sets the spacing of the software tab stops used by the editor to expand tabs in the input file. The default is **tabstop=8**.

taglength=*n* (**tl**)
: Set the maximum number of characters that should be treated as significant in a tag. Characters beyond the limit are ignored. A value of zero means that all characters in the tag are significant. The default is **taglength=0**.

tags=[*filename*]...
: Specify the tags files to be used by the **tag** command and the **-t** command-line option. The default is **tags=tags /usr/lib/tags**, specifying the file **tags** in the current directory and the file **/usr/lib/tags**. File names are separated by whitespace.

  Each line of a tags file contains the following three fields separated by whitespace: the tag name, the name of the file to be edited, and an address specification (see Addressing above). A **tags** file must be sorted in order by tag name.

  The **ctags** command (see *ctags*(1)) creates tags files from C, Pascal and FORTRAN source files.

tagstack (**tgst**)
: Enable the pushdown stack of activated tags. Reversed by **notagstack** (**notgst**). The default is **tagstack**.

  When you enter a line mode **tag** command or visual mode **^]** command, the current line number and file name are stored on the tag stack. A future line mode **pop** command or visual mode **^T** command will return to the stored file name at the stored line number.

  If the tag stack is disabled and then reenabled again, the stack continues where it left off. The **pop** command does not work when the tag stack is disabled.

term=*termtype*
: Define the type of terminal being used with the editor. The default value is obtained from the **TERM** environment variable. If **TERM** is unset or null, **term** is set to **unknown**. There is no difference between the **term** and **ttytype** editor options. Setting either one results in both being changed.

terse
: Use shorter error messages. Reversed by **noterse**. The default is **noterse**.

timeout (**to**)
: If set, require that all the characters of a multicharacter macro name (the first argument in a **map** command) must be received within the amount of time specified by the **timeoutlen** option in order to be accepted as a match for the macro name. If not set, no limit is placed on how long to wait for the completion of a macro name. Reversed by **notimeout** (**noto**). The default is **timeout**.

timeoutlen=*n* Set, in milliseconds (ms), the length of the macro timeout period (see the **timeout** editor option). This option has no effect unless **timeout** is set. The value of *n* must be at least 1. The default is **timeoutlen=500** (half a second).

ttytype=*termtype* (**tty**)
Define the type of terminal being used with the editor. See the **term** editor option for details. There is no difference between the **term** and **ttytype** editor options. Setting either one results in both being changed.

warn Before executing a **!** or **shell** command escape, display the message **[No write since last change]** if the work area has been modified since it was last loaded or fully written to a file. Reversed by **nowarn**. The default is **warn**.

window=*lines* (**wi**)
Set the number of lines in a text window in visual mode. The default value is one less than the size of your terminal screen (as defined by the **LINES** environment variable, if set, or the entry for your terminal in the *terminfo*(4) data base otherwise). However, if the terminal baud rate (see *stty*(1) is set to less than 1200 or 2400, the default value is reduced to a maximum of 8 or 16 lines, respectively. The startup value can be specified with the **-w** command-line option.

w300=*lines* If the terminal baud rate is less than 1200, set the **window** editor option to the value specified.

w1200=*lines* If the terminal baud rate is greater than or equal to 1200 but less than 2400, set the **window** editor option to the value specified.

w9600=*lines* If the terminal baud rate is greater than or equal to 2400, set the **window** editor option to the value specified.

wrapmargin=*n* (**wm**)
In visual mode only, if *n* is greater than zero, a newline is automatically inserted in an input line at a word boundary, so that lines end at least *n* spaces from the right margin of the terminal screen. The default is **wrapmargin=0**.

wrapscan (**ws**) When set, editor searches using **/***re***/** (or **?***re***?**) continue silently from the beginning (or end) of the file upon reaching the end (or beginning) of the file (that is, the scan "wraps around"). When unset, editor searches stop at the beginning or the end of the file, as appropriate. Reversed by **nowrapscan** (**nows**). The default is **wrapscan**.

writeany (**wa**) Inhibits the checks otherwise made before write commands, allowing a write to any file (provided the system allows it). Reversed by **nowriteany** (**nowa**). The default is **nowriteany**.

EXTERNAL INFLUENCES

Environment Variables

COLUMNS This variable shall override the system-selected horizontal screen size.

LINES overrides the system-selected vertical screen size, used as the number of lines in a screenful and as the vertical screen size in visual mode.

PATH determines the search path for the shell command specified in the editor commands, **shell**, **read**, and **write**.

SHELL is variable that shall be interpreted as the preferred command-line interpreter for use in **!**, **shell**, **read**, and other commands with an operand of the form **!string**. For the **shell** command, the program shall be invoked with the single argument **-i**. For all others, it shall be invoked with the two arguments **-c** and string. If no **SHELL** environment variable is set, or it is set to a null string, the **sh** utility shall be used.

TERM is a variable that shall be interpreted as the name of the terminal type. If this variable is unset or null, an unspecified default terminal type shall be used.

EXINIT is a variable that shall be interpreted to contain a list of **ex** commands that are executed on editor startup, before reading the first file. The list can contain multiple commands by separating then using a vertical line (|) character.

HOME shall be interpreted as a pathname of a directory that shall be searched for an editor startup file name **.exrc**.

LC_COLLATE determines the collating sequence used in evaluating regular expressions and in processing the **tags** file. If it is not specified or is null, it defaults to the value of **LANG**.

LC_CTYPE determines the interpretation of text as single and/or multibyte characters, the classification of characters as uppercase or lowercase letters, the shifting of the case of letters, and the characters matched by character class expressions in regular expressions. If it is not specified or is null, it defaults to the value of **LANG**.

LANG determines the language in which messages are displayed. If it is not specified or is null, it defaults to "C" (see *lang*(5)).

LC_ALL determines the locale to be used to override any values for locale categories specified by the setting of **LANG** or any environment variable (beginning with **LC_**).

LC_MESSAGES determines the processing of affirmative responses and the language in which messages should be written.

If any internationalization variable contains an invalid setting, all internationalization variables default to "C" (see *environ*(5)).

When set, the **TMPDIR** environment variable specifies a directory to be used for temporary files, overriding the default directory **/var/tmp**.

International Code Set Support

Single- and multibyte character code sets are supported.

ASYNCHRONOUS EVENTS (XPG4 Only)

The following actions shall be taken upon receipt of signals:

SIGINT

When an interrupt occurs, **ex** shall alert the terminal and write a message. The current editor command shall be aborted, and **ex** shall return to the command level and prompt for another command. If the standard input is not a terminal device, **ex** shall exit at the interrupt and return a nonzero exit status.

SIGCONT

The screen shall be refreshed.

SHIGHUP

If the current buffer has changed since the last **e** or **w** command, **ex** shall attempt to save the current file in a state such that it can be recovered later by an **ex -r** command.

The action taken for all other signals is unspecified.

EXTENDED DESCRIPTION (XPG4 Only)

The pathname of the file being edited by **ex** is referred to as the **current** file. The text of the file shall be read into a working version of the file (called **buffer** in this clause), and all editing changes shall be performed on that version; the changes shall have no effect on the original file until an **ex** command causes the file to be written out. Lines in the buffer may be limited to { **LINE_MAX** } bytes, and an error message may be written if the limit is exceeded during editing.

The **alternate** pathname is the name of the last file mentioned in an editor command, or the previous current pathname if the last file mentioned became the current file. When the **%** appears in a pathname entered as part of a command argument, it shall be replaced by the altername pathname. Any character, including **%** and **#** shall retain its literal value when preceded by a backslash.

When an error occurs, **ex** shall alert ther terminal and write a message.

If the system crashes, **ex** shall attempt to preserve the buffer if any unwritten changes were made. The command-line option **-r** can be used to retrieve the saved changes.

During initialization (before the first file is read or any user commands from the terminal are processed), if the environment variable **EXINIT** is set, the editor shall execute **ex** commands contained in that variable. If the variable is not set, **ex** shall attempt to read commands from the **$HOME/.exrc**. If and only if **EXINIT** or **$HOME/.exrc** sets the editor option exrc, ex finally shall attempt to read commands from a file **.exrc** in the current directory. In the event that **EXINIT** is not set and the current directory is the home directory of the user, any **.exrc** file shall only be processed once. No **.exrc** shall be read unless it is owned by the same user ID as the effective user ID of the process. After any **.exrc** files are processed, any commands specified by the **-c** option shall be processed.

By default, **ex** shall start in the command mode, which shall be indicated by the ":" prompt. The input mode can be entered by **append**, **insert**, or **change** commands. There is one other mode, visual mode, in which full screen editing is available. This is described more fully under the **visual** command. The command line can consist of multiple **ex** commands separated by vertical-line characters(|). The use of commands that enter input or visual modes in this manner, unless they are the final command on the line, produces undefined results.

Command lines beginning with the double-quote character (") shall be ignored. This can be used for comments in an editor script.

WARNINGS

The **undo** command causes all marks to be lost on lines that are changed and then restored.

The **z** command prints a number of logical rather than physical lines. More than a screenful of output can result if long lines are present.

Null characters are discarded in input files and cannot appear in resultant files.

On some systems, the recovery of an edit file with the **-r** option is possible only if certain system-dependent actions are taken when the system is restarted.

Edit preserve files can only be recovered on systems running the same HP-UX release in which they were preserved. Preserve files are not recoverable across different releases.

On HP terminals, the attribute field of any function key specified by a **map** *#n* ... command should be set to **normal** rather than to the default of **transmit**.

Do not use the **-C** option to edit unencrypted files. The **-C** option is meant to be used only on files that are already encrypted. If the **-C** option is used on files which are not yet encrypted, a write in the edit session is likely to corrupt the file.

For information about line length limits, file size limits, etc., see the WARNINGS section of *vi*(1).

EXIT STATUS (XPG4 Only)

The **ex** utility shall exit with one of the following values:

0	Successful completion.
>0	An error occurred.

AUTHOR

ex was developed by the University of California, Berkeley. The 16-bit extensions to **ex** are based in part on software of the Toshiba Corporation.

FILES

$HOME/.exrc	Primary editor initialization file
./.exrc	Secondary editor initialization file
/usr/lbin/expreserve	Preserve command
/usr/lbin/exrecover	Recover command
/usr/share/lib/terminfo/*/*	Description of terminal capabilities
/var/preserve	Preservation directory
/var/tmp/Ex*nnnnn*	Editor temporary file
/var/tmp/Rx*nnnnn*	Named buffer temporary file

SEE ALSO

ctags(1), ed(1), stty(1), vi(1), write(1), terminfo(4), environ(5), lang(5), regexp(5).

The Ultimate Guide to the vi and ex Text Editors, Benjamin/Cummings Publishing Company, Inc., ISBN 0-8053-4460-8, HP part number 97005-90015.

STANDARDS COMPLIANCE

ex: SVID2, SVID3, XPG2, XPG3, XPG4

e

NAME

expand, unexpand - expand tabs to spaces, and vice versa

SYNOPSIS

expand [**-t** *tablist*] [*file* ...]

unexpand [**-a**] [**-t** *tablist*] [*file* ...]

Obsolescent:

expand [**-***tabstop*] [**-***tab1***,***tab2***,**...**,** *tabn*] [*file* ...]

DESCRIPTION

expand processes the named files or the standard input and writes to the standard output with tabs changed into spaces. Backspace characters are preserved in the output, and the column count is decreased by one column for tab calculations. For proper tab calculation, if a multi-column character is to be "backspace'd", it should be followed by multiple backspace characters which equal to it's column width. If a tab character is found after the last tab position, it is replaced by a single space. **expand** is useful for preprocessing character files that contain tabs (before sorting, looking at specific columns, etc).

expand recognizes the following command-line options and arguments:

[**-t** *tablist*] *tablist* specifies where to set the tab positions instead of the default **8**. *tablist* can take two forms. If it is a single number, tabs are set *tablist* spaces apart. *tablist* can also be a blank- or comma-separated list of increasing positions where tabs are to be set.

[**-***tabstop*] This option is obsolescent and is equivalent to using **-t** *tabstop*.

[**-***tab1***,***tab2***,**...**,***tabn*]
This option is obsolescent and is equivalent to using **-t** *tab1***,***tab2***,** ... **,***tabn*.

unexpand processes the named files or the standard input and writes to the standard output with spaces changed into tabs where possible. By default, only leading spaces and tabs are converted to maximal strings of tabs. The default tab position is every 8 characters. Backspace characters are preserved into the output, and the column count is decreased by one column for tab calculations. For proper tab calculation, if a multi-column character is to be "backspace'd", it should be followed by multiple backspace characters which equal to it's column width.

unexpand recognizes the following command-line options and arguments:

-a Tabs are inserted whenever they would compress the resultant file by replacing two or more spaces before a tab position.

-t *tablist* *tablist* specifies the tab positions. *tablist* can take two forms. If it is a single number, tabs are set every *tablist* spaces apart. If *tablist* is a blank- or comma-separated list of increasing positions, tabs are set at those locations. The **-t** option implies the **-a** option. If the **-t** option is not specified, the default is equivalent to specifying **-t 8** except that **-a** is not implied for this case.

EXTERNAL INFLUENCES

Environment Variables

LC_CTYPE determines the interpretation of text as single and/or multi-byte characters.

LC_MESSAGES determines the language in which messages are displayed.

If **LC_CTYPE** or **LC_MESSAGES** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**.

If any internationalization variable contains an invalid setting, **expand** and **unexpand** behave as if all internationalization variables are set to "C". See *environ*(5).

If **LC_ALL** is set to a non-empty string value, it overrides the values of all the other internationalization variables.

International Code Set Support

Single- and multi-byte character code sets are supported with the exception that **unexpand** do not recognize multi-byte alternative space characters.

STANDARDS CONFORMANCE

expand: XPG4, POSIX.2

unexpand: XPG4, POSIX.2

NAME

expand_alias - recursively expands the sendmail aliases

SYNOPSIS

expand_alias [**-r** *max_recursion*] [**-t**] [**-tt**] *alias*

DESCRIPTION

Expand_alias is a shell script that recursively expands the **sendmail** aliases. Through use of **telnet host 25** and the **expn** command, each alias is recursively expanded into its destination(s). Indentation is used to show each level of recursion. Because of the recursive use of **telnet**, **expand_alias** is slow. If the local **telnet** cannot directly connect to a remote system, due to a firewall configuration, **expand_alias** will not be able to succeed. If the local **telnet** is to transparently connect across the firewall, **expand_alias** will be able to contact sendmail daemons outside the firewall, allowing the alias to be more fully expanded. (For example, some local telnet clients use a **socksd** located on the firewall to permit the local **telnet** client to transparently connect to Internet hosts. If the local default **telnet** uses a **socksd** in such a manner, **expand_alias** will use that **telnet** functionality to more fully expand an alias.)

max_recursion defaults to 10. After *max_recursion* expansions, no further expansion is attempted.

If **-t** is specified, only the terminal aliases will be displayed.

-tt is similar to **-t** except that if a terminal line has a pipe, its printing is suppressed and the previous level of expansion is printed instead.

EXAMPLES

expand_alias root

expand_alias root@cat

expand_alias root@cat.cup.hp.com

expand_alias root@cup.hp.com

AUTHOR

expand_alias was developed by the Hewlett-Packard Company.

NAME

expr - evaluate arguments as an expression

SYNOPSIS

expr *arguments*

DESCRIPTION

expr takes *arguments* as an expression, evaluates, then writes the result on the standard output. Terms in the expression must be separated by blanks. Characters special to the shell must be escaped. Note that **0**, rather than the null string, is returned to indicate a zero value. Strings containing blanks or other special characters should be quoted. Integer-valued arguments can be preceded by a unary minus sign. Internally, integers are treated as 32-bit, 2's complement numbers.

The operators and keywords are listed below. Characters that need to be escaped are preceded by \. The list is in order of increasing precedence with equal-precedence operators grouped within **{}** symbols.

expr **\\|** *expr* Returns the first *expr* if it is neither null nor **0**, otherwise returns the second *expr*.

expr **\\&** *expr* Returns the first *expr* if neither *expr* is null or **0**, otherwise returns **0**.

expr { **=**, **\\>**, **\\>=**, **\\<**, **\\<=**, **!=** } *expr*
If both arguments are integers, and if the comparison is satisfied, *expr* returns **1** otherwise it returns **0**. *expr* returns the result of an integer comparison if both arguments are integers; otherwise returns the result of a lexical comparison (note that **=** and **==** are identical, in that both test for equality).

expr { **+**, **-** } *expr*
Addition or subtraction of decimal integer-valued arguments.

expr { **\\***, **/**, **%** } *expr*
Multiplication, division or remainder of decimal integer-valued arguments producing an integer result.

expr **:** *expr* The matching operator **:** compares the first argument with the second argument which must be a regular expression. *expr* supports the Basic Regular Expression syntax (see *regexp*(5)), except that all patterns are "anchored" (i.e., begin with **^**) and, therefore, **^** is not a special character, in that context. Normally, the matching operator returns the number of characters matched (**0** on failure). Alternatively, the **\\(** ... **\\)** pattern symbols can be used to return a portion of the first argument.

length *expr* The length of *expr*.

substr *expr expr expr*
Takes the substring of the first *expr*, starting at the character specified by the second *expr* for the length given by the third *expr*.

index *expr expr* Returns the position in the first *expr* which contains a character found in the second *expr*.

match Match is a prefix operator equivalent to the infix operator **:**.

\\(... **\\)** Grouping symbols. Any expression can be placed within parentheses. Parentheses can be nested to a depth of **EXPR_NEST_MAX** as specified in the header file <**limits.h**>.

EXTERNAL INFLUENCES

Environment Variables

LC_COLLATE determines the collating sequence used in evaluating regular expressions and the behavior of the relational operators when comparing string values.

LC_CTYPE determines the interpretation of text as single- and/or multi-byte characters, and the characters matched by character class expressions in regular expressions.

LANG determines the language in which messages are displayed.

If **LC_COLLATE** or **LC_CTYPE** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**. If any internationalization variable contains an invalid setting, **expr** behaves as if all internationalization variables are set to "C" (see *environ*(5)).

International Code Set Support

Single- and multi-byte character code sets are supported.

RETURN VALUE

As a side effect of expression evaluation, *expr* returns the following exit values:

0	Expression is neither null nor zero.
1	Expression is null or zero.
2	Invalid expression.
>2	An error occurred while evaluating the expression.

DIAGNOSTICS

syntax error	Operator or operand errors
non-numeric argument	Arithmetic attempted on a string

EXAMPLES

Add 1 to the shell variable **a**:

```
a=‘expr $a + 1‘
```

For **$a** equal to either **/usr/abc/file** or just **file**, return the last segment of a path name (i.e., **file**). Beware of **/** alone as an argument because *expr* interprets it as the division operator (see WARNINGS below):

```
expr $a : ’.*/\(.*\)’ \| $a
```

A better representation of the previous example. The addition of the **//** characters eliminates any ambiguity about the division operator and simplifies the whole expression:

```
expr //$a : ’.*/\(.*\)’
```

Return the number of characters in **$VAR**:

```
expr $VAR : ’.*’
```

WARNINGS

After argument processing by the shell, *expr* cannot tell the difference between an operator and an operand except by the value. If **$a** is an **=**, the command:

```
expr $a = ’=’
```

resembles:

```
expr = = =
```

as the arguments are passed to *expr* (and they will all be taken as the **=** operator). The following works:

```
expr X$a = X=
```

AUTHOR

expr was developed by OSF and HP.

SEE ALSO

sh(1), test(1), environ(5), lang(5), regexp(5).

STANDARDS CONFORMANCE

expr: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

NAME

factor, primes - factor a number, generate large primes

SYNOPSIS

factor [*number*]

primes [*start* [*stop*]]

DESCRIPTION

If no arguments are provided on the command line, **factor** waits for a number to be typed in. If a positive number is typed, it factors the number and print its prime factors; each one is printed the proper number of times. It then waits for another number. **factor** exits if it encounters a zero or any non-numeric character.

If an argument is provided on the command line, **factor** factors the number as above, then exits.

Maximum time to factor is proportional to sqrt(*n*) and occurs when *n* is prime or the square of a prime.

The largest number that can be dealt with by **factor** is 1.0e14.

primes prints prime numbers between a lower and upper bound. If no arguments are provided on the command line, **primes** waits for two numbers to be typed in. The first number is interpreted as the lower bound; the second as the upper bound. All prime numbers in the resulting inclusive range are printed.

If *start* is specified, all primes greater than or equal to *start* are printed. If both *start* and *stop* are given, all primes occurring in the inclusive range *start* through *stop* are printed.

start and *stop* values must be integers represented as long integers.

If the stop value is omitted in either case, **primes** runs either until overflow occurs or until it is stopped by typing the interrupt character.

The largest number that can be dealt with by **primes** is 2,147,483,647.

DIAGNOSTICS

Both commands print **Ouch** when the input is out of range, illegal characters are encountered, or when *start* is greater than *stop*.

EXAMPLES

Print the prime factorization for the number 12:

```
factor 12
```

Print all prime numbers between 0 and 20:

```
primes 0 20
```

NAME

fastbind - Prepare an incomplete executable for faster program start-up

SYNOPSIS

fastbind [-nu] *incomplete-executable...*

DESCRIPTION

fastbind is a tool that can improve the start-up time of programs that use shared libraries (incomplete executables) by storing information about needed shared library symbols in the executable file.

fastbind performs analysis on the symbols used to bind an executable and all of it's dependent shared libraries, and stores this information in the executable file. The next time the executable is run, the dynamic loader (**/usr/lib/dld.sl** for 32-bit PARISC or **/usr/lib/pa20_64/dld.sl** for 64-bit PARISC) will notice that this information is available, and it will use this fastbind information to bind the executable instead of the standard search method for binding the symbols.

Since **fastbind** writes the fastbind information in the executable file, you must have write permission on the executable file. Also, if the executable file being analyzed is being run as another process, the file will be locked against modifications by the kernel, and **fastbind** will fail.

If the shared libraries that an executable is dependent on are modified after the fastbind information is created, the dynamic loader will silently revert to standard search method for binding the symbols. The fastbind information can be re-created by running **fastbind** on the executable again. **fastbind** will automatically erase the old fastbind information and generate the new one.

The **ld** option **+fb** can be used to instruct the linker to run the fastbind tool on an incomplete executable it has produced.

Environment Variables

If **dld** determines that the fastbind information is out of date, it will silently revert to standard search method for binding the symbols. If the environment variable **_HP_DLDOPTS** is set to **-fbverbose** the dynamic loader will emit a warning message when the fastbind information is out of date.

The environment variable **_HP_DLDOPTS** can be set to **-nofastbind** to make the dynamic loader ignore the fastbind information and revert to the standard search method for binding the symbols.

Options

fastbind recognizes the following options:

-n Remove the fastbind information from the executable, returning it to the same state it was in before **fastbind** was originally run on it.

-u Normally, if **fastbind** detects any unsatisfied symbols while building the fastbind information, it will generate an error message and not modify the executable file. When **fastbind** is invoked with **-u** option however, unresolved symbols are allowed.

EXTERNAL INFLUENCES

Environment Variables

The following internationalization variables affect the execution of **fastbind**:

LANG
Determines the locale category for native language, local customs and coded character set in the absence of **LC_ALL** and other **LC_*** environment variables. If **LANG** is not specified or is set to the empty string, a default of **C** (see *lang*(5)) is used instead of **LANG**.

LC_ALL
Determines the values for all locale categories and has precedence over **LANG** and other **LC_*** environment variables.

LC_MESSAGES
Determines the locale that should be used to affect the format and contents of diagnostic messages written to standard error.

LC_NUMERIC
Determines the locale category for numeric formatting.

LC_CTYPE
Determines the locale category for character handling functions.

NLSPATH
Determines the location of message catalogs for the processing of **LC_MESSAGES**.

If any internationalization variable contains an invalid setting, **fastbind** behaves as if all internationalization variables are set to **C**. See *environ*(5).

In addition, the following environment variable affects **fastbind**:

TMPDIR
Specifies a directory for temporary files (see *tmpnam*(3S)).

DIAGNOSTICS

fastbind returns zero when the operation is successful. A non-zero return code indicates that an error occurred.

EXAMPLES

To run **fastbind** on the executable file **a.out**
enter:

```
fastbind a.out
```

To later remove the fastbind information from the executable file **a.out** enter:

```
fastbind -n a.out
```

WARNINGS

32-bit PARISC **fastbind** does not work with EXEC_MAGIC executables.

fastbind effectively enforces bind restricted and bind immediate. For example, consider an executable linked bind deferred, which calls a function foo() defined in an implicitly loaded library. Before the actual call is made, if it explicitly loads a shared library (using *shl_load*(3X) with **BIND_FIRST**) having a definition for foo(), when foo() is finally called, it will be resolved from the explicitly loaded library. But after running fastbind, the symbol foo() will be resolved from the implicitly loaded library.

AUTHOR

fastbind was developed by Hewlett-Packard.

FILES

a.out	output file
/usr/lib/dld.sl	32-bit PARISC dynamic loader
/usr/lib/pa20_64/dld.sl	64-bit PARISC dynamic loader
/usr/lib/nls/$LANG/fastbind.cat	message catalog
/var/tmp/__FB*	temporary files

SEE ALSO

System Tools:

ld(1)	invoke the link editor

Miscellaneous:

a.out(4)	assembler, compiler, and linker output
dld.sl(5)	dynamic loader

Texts and Tutorials:
HP-UX Linker and Libraries User's Guide

NAME

fastmail - quick batch mail interface

SYNOPSIS

fastmail [**-b** *bcc-list*] [**-c** *cc-list*] [**-C** *comments*] [**-f** *from-name*] [**-F** *from-addr*] [**-i** *in-reply-to*] [**-r** *reply-to*] [**-R** *references*] [**-s** *subject*] *filename address-list*

DESCRIPTION

The **fastmail** command is a simple interface to the mail system that allows you to send a message without the overhead of an interactive mailer. It is particularly efficient in batch-processing mail to very large groups of people.

All addresses should be full e-mail addresses, **sendmail** aliases in the **/etc/mail/aliases** file, or local login names.

Options

fastmail recognizes the following options:

-b *bcc-list* Include a **Bcc:** header entry. Send blind carbon copies to the comma-separated list of addresses in *bcc-list*.

-c *cc-list* Include a **Cc:** header entry. Send carbon copies to the comma-separated list of addresses in *cc-list*.

-C *comments* Include a **Comments:** header entry with the string value *comments*.

-d Debug. Display information on processing steps.

-f *from-name* Replace the user name in the **From:** header entry with *from-name*.

If the user is **x@y**, and the user name is **MrX**, then the default **From:** line is:

From: x@y (MrX) .

The option **-f Joe** changes it to:

From: x@y (Joe)

-F from-addr Replace the address in the **From:** header entry with *from-addr*. In the **-f** example above, **-F a@b** changes the original entry to

From: a@b (MrX)

-i *in-reply-to* Include the **In-Reply-To:** header entry with the string value *in-reply-to*. This is usually used to identify a message that you are replying to.

-r *replyto* Include the **Reply-To:** header entry with the single address given in *replyto*. This is the address where replies will usually be sent, instead of to the address given in the **From:** header entry, very common with mailing lists.

-R *references* Include a **References:** header entry containing the string value *references*.

-s *subject* Include a **Subject:** header entry containing the value *subject*. If this option is omitted, the message is sent without a subject entry.

Operands

fastmail recognizes the following operands:

address-list A list of one or more blank-separated addresses for the **To:** header line. These are the principal recipients of the message.

filename Either the name of a file containing the message, or a dash (**-**) to read from standard input.

EXAMPLES

A Fully Specified Command

This command has every option specified.

```
fastmail \
    -b "bcc1,bcc2,bcc3,bcc4" \
    -C "Just a Comment" \
    -c "cc1,cc2,cc3,cc4" \
```

f

```
        -d \
        -F me@anotherhost.com \
        -f My Name \
        -i "Your recent message" \
        -R REF:13579 \
        -r oscar \
        -s "Testing fastmail" \
        message-file \
        addr1 addr2 addr3 addr4
```

The online execution displays the following debug messages:

```
Mailing to addr1,addr2,addr3,addr4 cc1,cc2,cc3,cc4 bcc1,bcc2,bcc
3,bcc4 [via sendmail]
cat /tmp/fastmail.5578 message-file | /usr/sbin/sendmail addr1,a
ddr2,addr3,addr4 cc1,cc2,cc3,cc4 bcc1,bcc2,bcc3,bcc4
```

The received message has the following relevant header entries:

```
From realsender@mycomputer.myhost.com Tue Oct 22 21:14:04 EDT 1996
Subject: Testing fastmail
From: me@anotherhost.com (My Name)
Reply-To: oscar@mycomputer.myhost.com
To: addr1@mycomputer.myhost.com, addr2@mycomputer.myhost.com,
        addr3@mycomputer.myhost.com, addr4@mycomputer.myhost.com
Cc: cc1@mycomputer.myhost.com, cc2@mycomputer.myhost.com,
        cc3@mycomputer.myhost.com, cc4@mycomputer.myhost.com
References: REF:13579
In-Reply-To: Your recent message
Comments: Just a Comment
```

The **Bcc:** header entry is not transmitted.

A Batch Process

Suppose you are user **big** on machine **big-machine** and you have a shell script named **batch-mail** that contains the following lines:

```
#
# Batch Mail - batch mailing of a file to a LOT of users
#
# Usage: batch-mail "<from>" "<subject>" <filename>

sender_copy=$LOGIN
replyto=The-Mr-Big-list

fastmail -b $sender_copy -r $replyto -f "$1" -s "$2" $3 person1
sleep 10
fastmail -r $replyto -f "$1" -s "$2" $3 person2
sleep 10
fastmail -r $replyto -f "$1" -s "$2" $3 person3
sleep 10
fastmail -r $replyto -f "$1" -s "$2" $3 person4
```

The command:

```
batch-mail "Mr. Big" "Warning to all" warning.text
```

would mail a copy of the **warning.text** file to **person1**, **person2**, **person3**, and **person4**, staggered ten seconds apart.

$LOGIN would also silently receive a copy of the first message in the mail. Each resultant message would include the header lines:

```
From: big@big-machine (Mr. Big)
Subject: Warning to all
Reply-To: The-Mr-Big-list
```

FILES

`/etc/mail/aliases`	**`sendmail`** aliases file.
`/usr/sbin/sendmail`	Mail transport agent.
`/tmp/fastmail.`*pid*	Temporary file.

AUTHOR

`fastmail` was developed by HP.

SEE ALSO

elm(1), sendmail(1M).

RFC 822 "Standard for the Format of Internet Text Messages"

NAME

file - determine file type

SYNOPSIS

file [**-m** *mfile*] [**-c**] [**-f** *ffile*] [**-h**] *file* ...

DESCRIPTION

file performs a series of tests on each *file* in an attempt to classify it. If *file* appears to be an ASCII file, **file** examines the first 512 bytes and tries to guess its language. If *file* is an executable **a.out** file, **file** prints the version stamp, provided it is greater than 0 (see the description of the **-V** option in *ld*(1)).

file uses the file **/etc/magic** to identify files that have some sort of **magic number**, that is, any file containing a numeric or string constant that indicates its type. Commentary at the beginning of **/etc/magic** explains the format.

Options

file recognizes the following command-line options:

-m *mfile*	Use alternate magic file *mfile*.
-c	Check the magic file for format errors. This validation is not normally carried out for reasons of efficiency. No file classification is done when this option is specified.
-f *ffile*	Obtain the list of files to be examined from file *ffile*. **file** classifies each file whose name appears in *ffile*.
-h	Do not follow symbolic links.

EXTERNAL INFLUENCES

Environment Variables

LC_MESSAGES determines the language in which messages are displayed.

If **LC_MESSAGES** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**.

If any internationalization variable contains an invalid setting, **file** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported. However, all non-ASCII text files are identified as "data".

WARNINGS

The **file** command for a release interprets the core files for that particular release correctly. Using the **file** command on a core file generated on a different release will report incorrect results.

SEE ALSO

ld(1).

STANDARDS CONFORMANCE

file: SVID2, SVID3, XPG2, XPG4

NAME

find - find files

SYNOPSIS

find *pathname_list* [*expression*]

DESCRIPTION

The **find** command recursively descends the directory hierarchy for each path name in *pathname_list* (that is, one or more path names) seeking files that match a Boolean *expression* written in the primaries given below. By default, **find** does not follow symbolic links.

The Boolean expression is evaluated using short-circuit evaluation. This means that whenever the result of a Boolean operation (AND or OR) is known from evaluating the left-hand argument, the right-hand argument is not evaluated.

In the descriptions of the primaries, the argument *n* represents a decimal integer; **+***n* means more than *n*, **-***n* means less than *n*, and *n* means exactly *n*.

The following primaries are recognized:

-depth
: A position-independent term which causes descent of the directory hierarchy to be done so that all entries in a directory are acted on before the directory itself. This can be useful when **find** is used with *cpio*(1) to transfer files that are contained in directories without write permission. It is also useful when using *cpio*(1) and the modification dates of directories must be preserved. Always true.

-follow
: A position-independent term which causes **find** to follow symbolic links. When following symbolic links, **find** keeps track of the directories visited so that it can detect infinite loops; for example, such a loop would occur if a symbolic link pointed to an ancestor. This expression should not be used with the **-type l** expression. Always true.

-fsonly *FStype*
: A position-independent term which causes **find** to stop descending any directory whose file system is not of the type specified by *FStype*, where *FStype* is one of **cdfs**, **hfs**, **vxfs**, or **nfs**, representing the CDFS, HFS, JFS (VXFS) or NFS file system type, respectively.

: In this context, mount points inherit the *FStype* of their parent directory. This means that when **-fsonly hfs** has been specified and **find** encounters an NFS mount point that is mounted on an HFS file system, the mount point will be visited but entries below that mount point will not. It is important to note that when **-fsonly nfs** has been specified, any HFS file systems that are beneath the mount point of an NFS file system are not traversed. Always true.

-local
: True if the file physically resides on the local system. This does not restrict the search to only files which physically reside on the local system, it merely matches such files. See *EXAMPLES*.

-xdev
: A position-independent term that causes **find** to avoid crossing any file system mount points that exist below starting points enumerated in *pathname_list*. The mount point itself is visited, but entries below the mount point are not. Always true.

-mountstop
: Identical to **-xdev**. This primary is provided for backward compatibility only. **-xdev** is preferred over **-mountstop**.

-name *file*
: True if *file* matches the last component of the current file name. The matching is performed according to Pattern Matching Notation (see *regexp*(5)). Pattern may contain supplementary code set characters.

-path *file*
: Same as **-name** except the full path (as would be output by **-print)** is used instead of just the base name. Note that **/** characters are not treated as a special case. For example, ***/.profile** matches **./home/fred/.profile**.

-perm [**-**]*mode*
: In this primary, the argument *mode* is used to represent file mode bits. The argument is identical in format to the *mode* operand as described in *chmod*(1), with the exception that the first character must not be the **-** operator. When using the symbolic form of *mode*, the starting template is assumed to have all file mode bits cleared.

If the leading minus is omitted, this primary is true when the file permission bits exactly match the value of *mode*. Bits associated with the symbolic attributes **s** (set-user-ID, set-group-ID) and **t** (sticky bit) are ignored when the minus is omitted.

If *mode* is preceded by a minus, this primary is true if all of the bits that are set in *mode* are also set in the file permission bits. In this case, the bits associated with the symbolic attributes **s** and **t** are significant.

-fstype *FStype* — True if the file system to which the file belongs is of type *FStype*, where *FStype* is one of **cdfs**, **hfs**, or **nfs**, corresponding to the CDFS, HFS, or NFS file system type, respectively.

-type *c* — True if the type of the file is *c*, where *c* is one of:

f	Regular file
d	Directory
b	Block special file
c	Character special file
p	FIFO (named pipe)
l	Symbolic link
s	Socket
n	Network special file
M	Mount point

-links *n* — True if the file has *n* links.

-user *uname* — True if the file belongs to the user *uname*. If *uname* is numeric and does not appear as a login name in the **/etc/passwd** file, it is taken as a user ID. The *uname* operand can be preceded by a **+** or **-** to modify the comparison of the primaries. If the argument *n* represents a decimal integer; **+***n* means more than *n*, **-***n* means less than *n*, and *n* means exactly *n*.

-group *gname* — True if the file belongs to the group *gname*. If *gname* is numeric and does not appear in the **/etc/group** file, it is taken as a group ID. The *gname* operand can be preceded by a **+** or **-** to modify the comparison of the primaries. If the argument *n* represents a decimal integer; **+***n* means more than *n*, **-***n* means less than *n*, and *n* means exactly *n*.

-nouser — True if the file belongs to a user ID that is not listed in the password database. See *passwd*(4).

-nogroup — True if the file belongs to a group ID that is not listed in the group database. See *group*(4).

-size *n*[**c**] — True if the file is *n* blocks long (512 bytes per block). If *n* is followed by a **c**, the size is in bytes.

-atime *n* — True if the file access time subtracted from the initialized time is *n*-1 to *n* multiples of *24* h. The initialization time shall be a time between the invocation of the **find** utility and the first access by that invocation of the **find** utility to any file specified by its **path** operands. The access time of directories in *pathname_list* is changed by **find** itself.

-mtime *n* — True if the file modification time subtracted from the initialization time is *n*-1 to *n* multiples of *24* h. The initialization time shall be a time between the invocation of the **find** utility and the first access by that invocation of the **find** utility to any file specified in its **path** operands.

-ctime *n* — True if the time of last change of file status information subtracted from the initialization time is *n*-1 to *n* multiples of *24* h. The initialization time shall be a time between the invocation of the **find** utility and the first access by that invocation of the **find** utility to any file specified by its **path** operands.

-newer *file* — True if the current file has been modified more recently than the argument *file*.

-newer[*tv1*[*tv2*]] *file* — True if the indicated time value (*tv1*) of the current file is newer than the indicated time value (*tv2*) of *file*. The time values *tv1* and *tv2* are each selected from the set of characters:

a	The time the file was last accessed
c	The time the inode of the file was last modified
m	The time the file was last modified

If the *tv2* character is omitted, it defaults to **m**. Note that the **-newer** option is equivalent to **-newermm**.

Syntax examples;

-newera *file*
-newermc *file*

-inum *n* — True if the file serial number (inode number) is *n*. Note that file serial numbers are unique only within a given file system. Therefore, matching file serial numbers does not guarantee that the referenced files are the same unless you restrict the search to a single file system.

-linkedto *path* — True if the file is the same physical file as the file specified by *path* (i.e., linked to *path*). This primary is similar to **-inum**, but correctly detects when a file is hard-linked to *path*, even when multiple file systems are searched.

-print — Causes the current path name to be printed. Always true.

-exec *cmd* — True if the executed *cmd* returns a zero value as exit status. The end of *cmd* must be punctuated by a semicolon (**;**) or a plus sign (**+**) (semicolon and plus are special to the shell and must be escaped). When a plus sign is used, *cmd* aggregates a set of pathnames and executes on the set. The reason for preferring **+** to a semicolon is vastly improved performance. Any command argument **{}** is replaced by the current path name. *cmd* may contain supplementary code set characters.

-ok *cmd* — Same as **-exec** except that the generated command line is printed with a question mark first, and is executed only if the user responds by typing **y**. The form of the affirmative response is locale dependent: y in the C locale, see LANG on environ(5). *cmd* may contain supplementary code set characters.

-cpio *device* — Write the current file on *device* in *cpio*(4) format (5120-byte records). The use of **-cpio** implies **-depth**. Always true.

-ncpio — Same as **-cpio** but adds the **-c** option to **cpio**. The use of **-ncpio** implies **-depth**. Always true.

-prune — If the current entry is a directory, cause **find** to skip that directory. This can be useful to avoid walking certain directories, or to avoid recursive loops when using **cpio -p**. Note, however, that **-prune** is useless if the **-depth** option has also been given. See the description of **-only** and the *EXAMPLES* section, below, for more information. Always true.

-only — This is a positive-logic version of **-prune**. A **-prune** is performed after every directory, unless **-only** is successfully evaluated for that directory. As an example, the following three commands are equivalent:

```
find . -fsonly hfs -print
find . -print -fstype hfs -only
find . -print ! -fstype hfs -prune
```

Note, however, that **-only** is useless if the **-depth** option has also been given. Always true.

(*expression* **)** — True if the parenthesized expression is true. The spaces are required. Parentheses are special to the shell and must be escaped, as in **\(** and **\)**.

Primaries can be combined by using the following operators (in order of decreasing precedence):

! *expression*	Logical NOT operator. True if *expression* is not true.
expression [**-a**] *expression*	Logical AND operator. True if both of the *expression*s are true.
expression **-o** *expression*	Logical OR operator. True if either or both of the *expression*s are true.

If *expression* is omitted, or if none of **-print**, **-ok**, **-exec**, **-cpio**, or **-ncpio** is specified, **-print** is assumed. The **-user**, **-group**, and **-newer** primaries each evaluate their respective arguments once.

Access Control Lists

The **-acl** primary enables the user to search for access control list entries. It is true if the file's access control list matches an access control list pattern or contains optional access control list entries (see *acl*(5)). It has three forms:

-acl *aclpatt*	Match all files whose access control list includes all (zero or more) pattern entries specified by the *aclpatt* pattern.
-acl =*aclpatt*	Match a file only if its access control list includes all (zero or more) pattern entries specified by the *aclpatt* pattern, and every entry in its access control list is matched by at least one pattern entry specified in the *aclpatt* pattern.
-acl opt	Match all files containing optional access control list entries.

The *aclpatt* string can be given as an operator or short form pattern; see *acl*(5).

By default, **-acl** is true for files whose access control lists include all the (zero or more) access control list patterns in *aclpatt*. A file's access control list can also contain unmatched entries.

If *aclpatt* begins with **=**, the remainder of the string must match all entries in a file's access control list.

The *aclpatt* string (by default, or the part following **=**) can be either an access control list or an access control list pattern. However, if it is an access control list, *aclpatt* must include at least the three base entries ((*user*.%, mode), (%.*group*, mode), and (%.%, mode)).

As a special case, if *aclpatt* is the word **opt**, the primary is true for files with access control list entries.

EXTERNAL INFLUENCES

Environment Variables

If an internationalization variable is not specified or is null, it defaults to the value of **LANG**.

If **LANG** is not specified or is null, it defaults to **C** (see *lang*(5)).

If **LC_ALL** is set to a nonempty string value, it overrides the values of all the other internationalization variables.

If any internationalization variable contains an invalid setting, all internationalization variables default to **C** (see *environ*(5)).

LC_CTYPE determines the interpretation of text as single and/or multibyte characters, the classification of characters as printable, and the characters matched by character class expressions in regular expressions.

LC_MESSAGES determines the locale that should be used to affect the format and contents of diagnostic messages written to standard error and informative messages written to standard output.

NLSPATH determines the location of message catalogues for the processing of **LC_MESSAGES**.

International Code Set Support

Single- and multibyte character code sets are supported.

EXAMPLES

Search the two directories **/example** and **/new/example** for files containing the string **Where are you** and print the names of the files:

```
find /example /new/example -exec grep -l 'Where are you' {} \;
```

Remove all files named **a.out** or ***.o** that have not been accessed for a week:

```
find / \( -name a.out -o -name '*.o' \) -atime +7 -exec rm {} \;
```

Note that the spaces delimiting the escaped parentheses are required.

Print the names of all files on this machine. Avoid walking **nfs** directories while still printing the **nfs** mount points:

```
find / -fsonly hfs -print
```

Match only local files, and do not examine the contents of any directory found to be remotely mounted:

```
find / ! -local -prune -o -size +50 -print
```

This only works correctly if there are no local file systems mounted on top of remote directories. This example will print all local files on the system larger than 50 blocks, without wasting time accessing remote files.

To get the same effect, but to check for files in local file systems mounted on remote directories, use:

```
find / -local -size +50 -print
```

Copy the entire file system to a disk mounted on **/Disk**, avoiding the recursive copy problem. Both commands are equivalent (note the use of **-path** instead of **-name**):

```
cd /; find . ! -path ./Disk -only -print | cpio -pdxm /Disk
cd /; find . -path ./Disk -prune -o -print | cpio -pdxm /Disk
```

Copy the root disk to a disk mounted on **/Disk**, skipping all mounted file systems below **/**. Note that **-xdev** does not cause **/** to be skipped, even though it is a mount point. This is because **/** is the starting point and **-xdev** only affects entries **below** starting points.

```
cd /;  find . -xdev -print | cpio -pdm /Disk
```

Change permissions on all regular files in a directory subtree to mode 444, and permissions on all directories to 555:

find *pathname* **-type f -print | xargs chmod 444**
find *pathname* **-type d -print | xargs chmod 555**

Note that output from **find** was piped to *xargs*(1) instead of using the **-exec** primary. This is because when a large number of files or directories is to be processed by a single command, the **-exec** primary spawns a separate process for each file or directory, whereas *xargs* collects file names or directory names into multiple arguments to a single *chmod* command, resulting in fewer processes and greater system efficiency. The **+** delimiter for the **-exec** primary can be used to achieve the same efficiency.

Access Control List Examples

Find all files not owned by user **karl** that have access control lists with at least one entry associated with **karl**, and one entry for no specific user in group **bin** with the read bit on and the write bit off:

```
find  /  ! -user karl -acl 'karl.*, %.bin+r-w' -print
```

Find all files that have a read bit set in any access control list entry:

```
find  /  -acl '*.*+r' -print
```

Find all files that have the write bit unset and execute bit set in every access control list entry:

```
find  /  -acl '=*.*-w+x' -print
```

Find all files that have optional access control list entries:

```
find  /  -acl opt -print
```

DEPENDENCIES

NFS

The **-acl** primary is always false for NFS files.

WARNINGS

Because of interoperability goals, **cpio** does not support archiving files larger than 2GB or files that have user/group IDs larger than 60,000 (60K). Files with user/group IDs greater than 60K are archived and restored under the user/group ID of the current process.

AUTHOR

find was developed by AT&T and HP.

FILES

/etc/group	Group names
/etc/mnttab	Mount points
/etc/passwd	User names

SEE ALSO

chacl(1), chmod(1), cpio(1), sh(1), test(1), xargs(1), mknod(2), stat(2), cpio(4), fs(4), group(4), passwd(4), acl(5), environ(5), lang(5), regexp(5).

STANDARDS CONFORMANCE

find: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

NAME

findmsg, dumpmsg - create message catalog file for modification

SYNOPSIS

findmsg [**-aiv**] [[**-D** *sym*] [**-U** *sym*]] *file* ...

dumpmsg *file* ...

DESCRIPTION

The **findmsg** command extracts messages from a C program source *file* and writes them to the standard output in a format suitable for input to **gencat** (see *gencat*(1)). The input file will be preprocessed using **cpp** (see *cpp*(1)) in order to select print specifiers and handle **ifdef**, **ifndef**... conditional **cpp** primitives. If multiple input files are specified and the **-a** option is not used, the files are processed sequentially such that message catalog comment lines identifying the input *file* are written before the output for each input *file*.

The **findmsg** command scans the source files for uncommented lines with one of the following three formats embedded within it:

catgets(*any_var* **,NL_SETN,** *n* **,<** *message* **>)**

< *message* **>**   **/* catgets** *n* ***/**

/* catgets *n* ***/**   **<** *message* **>**

or any combination of these formats wholly contained on a single physical line. **<** *message* **>** could be a string constant or a combination of string constants and print specifiers (PRI*). Any number of spaces or tabs can separate the **catgets** comment from the *message*. The digit *n*, which can be any valid message number (see *gencat*(1)), is combined with the *message* string to produce a message catalog source line. The message source line is assigned to the set whose number is the current value of **NL_SETN** as set by the last **#define** directive encountered. If **NL_SETN** has not yet been defined when a message line is found, the message is output without a set number specification. If more than one message is found belonging to the same set and message number, the last message found is output; any others are silently discarded. Conditional compilation and **#include** instructions in the C source files are ignored.

Options

findmsg recognizes the following command-line options:

-a Merge identically numbered sets from multiple input files so that **gencat** can process the **findmsg** output.

-D*sym* Define symbol *sym* .

-U*sym* Cause symbol *sym* to be undefined.

-i Consider all #ifdefs to extract messages from the input file. Options **-D** and **-U** will be used to select print specifiers if this option is not used.

-v Outputs all error messages issued by **cpp**. By default, **findmsg** does not display the error messages issued by **cpp**.

The **dumpmsg** command extracts messages from a message catalog *file* created by **gencat**. Messages are written to standard output in a format suitable for editing and re-input to **gencat**.

EXTERNAL INFLUENCES

Environment Variables

LC_CTYPE determines the interpretation of messages as single-byte and/or multi-byte characters.

LC_MESSAGES determines the language in which messages are displayed.

If **LC_CTYPE** or **LC_MESSAGES** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of **C** (see *lang*(5)) is used instead of **LANG**. If any internationalization variable contains an invalid setting, **findmsg** and **dumpmsg** behave as if all internationalization variables are set to **C**. See *environ*(5).

International Code Set Support

Single-byte and multi-byte character code sets are supported.

f

WARNINGS

The **findmsg** and **dumpmsg** commands are HP proprietary, not portable to other vendors' systems, and will not be provided in future HP-UX releases.

AUTHOR

findmsg and **dumpmsg** were developed by HP.

SEE ALSO

findstr(1), gencat(1), insertmsg(1), catgets(3C).

NAME

findstr - find strings for inclusion in message catalogs

SYNOPSIS

findstr *file* ...

DESCRIPTION

findstr examines files of C source code for uncommented string constants which it places, along with the surrounding quotes, on the standard output, preceding each by the file name, start position, and length. This information is used by **insertmsg** (see *insertmsg*(1)). **findstr** does not output strings that are parameters of the **catgets()** routine (see *catgets*(3C)).

EXTERNAL INFLUENCES

Environment Variables

LC_CTYPE determines the interpretation of comments and string literals as single- and/or multi-byte characters.

LC_MESSAGES determines the language in which messages are displayed.

If **LC_CTYPE** or **LC_MESSAGES** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**. If any internationalization variable contains an invalid setting, **findstr** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported.

WARNINGS

findstr outputs initialization strings of static string variables. Calling **insertmsg** with these strings causes their replacement with a call to **catgets()** (see *catgets*(3C)). Since the initializer must be a string, this assignment results in an invalid C declaration. For example, the following line:

```
static char *x[] = "message"
```

is modified by **insertmsg** (see *insertmsg*(1)) to:

```
static char *x[] = (catgets(catd,NL_SETN,1,"message"))
```

These strings should be manually removed from **findstr** output before being input to **insertmsg**.

findstr will not be provided in future HP-UX releases.

SEE ALSO

insertmsg(1).

f

f

NAME

finger - user information lookup program

SYNOPSIS

finger [*options*] *user_name* ...

DESCRIPTION

By default, **finger** lists for each *user_name* on the system:

- Login name,
- Full given name,
- Terminal write status (if write permission is denied),
- Idle time,
- Login time,
- User's home directory and login shell,
- Any plan the user has placed in file **.plan** in their home directory,
- Project on which they are working from the file **.project**, also in the home directory,
- office location and phone number (if known),
- last time the user received the mail, and last time the user read the mail.

Idle time is in minutes if listed as a single integer, hours and minutes if a **:** is present, or days and hours if a **d** is present. Account names as well as first and last names of users are accepted.

finger can also be used to list users on a remote machine. The format for *user_name* is *user_name@host*. If *user_name* is not specified, the remote system (HP-UX or non-HP-UX) uses its default standard format for listing user information.

Options

finger recognizes the following options:

-b	Suppress printing the user's home directory and shell.
-f	Suppress printing the header that is normally printed in a short-format printout.
-h	Suppress printing the **.project** file in a long-format printout.
-i	Force "idle" output format. Similar to short format except that only the login name, terminal, login time, and idle time are printed.
-l	Force long output format.
-m	Match arguments only on user name.
-p	Suppress printing of the **.plan** files
-q	Force quick output format. Similar to short format except that only the login name, terminal, and login time are printed.
-R	Print the user's host name.
-s	Force short output format.
-w	Suppress printing the full name in a short-format printout.

WARNINGS

Only the first line of the **.project** file is printed.

AUTHOR

finger was developed by the University of California, Berkeley.

FILES

/etc/utmp	who file
/var/adm/wtmp	last login file
/etc/passwd	for users names, offices, ...
~/.plan	plans
~/.project	projects
/var/mail	mail directory

SEE ALSO

chfn(1), who(1).

NAME

fmt - format text

SYNOPSIS

fmt [**-cs**] [**-w** *width*] [*file*...]

DESCRIPTION

The **fmt** command is a simple text formatter that fills and joins lines to produce output lines of (up to) the number of characters specified in the **-w** *width* option. The default *width* is 72. **fmt** concatenates the **file** arguments. If none are given, **fmt** formats text from the standard input.

Blank lines are preserved in the output, as is the spacing between words. **fmt** does not fill lines beginning with a period (**.**), for compatibility with **nroff**. Nor does it fill lines starting with **From:**.

Indentation is preserved in the output and input lines with differing indentation are not joined (unless **-c** is used).

fmt can also be used as an in-line text filter for **vi**; the **vi** command:

```
!}fmt
```

reformats the text between the cursor location and the end of the paragraph.

Options

fmt recognizes the following options:

-c Crown margin mode. Preserve the indentation of the first two lines within a paragraph and align the left margin of each subsequent line with that of the second line. This is useful for tagged paragraphs.

-s Split lines only. Do not join short lines to form longer ones. This prevents sample lines of code, and other such formatted text, from being unduly combined.

-w Fill output lines to up to *width* columns.

WARNINGS

The **-w** *width* option is acceptable for BSD compatibility, but it may go away in future releases.

SEE ALSO

nroff(1), vi(1).

NAME

fold - fold long lines for finite width output device

SYNOPSIS

fold [**-b**] [**-s**] [**-w** *width*] [*file* ...]

Obsolete form:

fold [**-s**] [**-***width*] [*file* ...]

DESCRIPTION

The **fold** command is a filter that folds the contents of the specified files, breaking the lines to have a maximum of *width* column positions (or bytes, if the **-b** option is specified). The **fold** command breaks lines by inserting a newline character so that each output line is the maximum width possible that does not exceed the specified number of column positions (or bytes). A line cannot be broken in the middle of a character. If no files are specified or if a *file* name of **-** is specified, the standard input is used.

The **fold** command is often used to send text files to line printers that truncate, rather than fold, lines wider than the printer is able to print.

If the backspace, tab, or carriage-return characters are encountered in the input, and the **-b** option is not specified, they are treated specially as follows:

Backspace	The current count of line width is decremented by one, although the count never becomes negative. Thus, the character sequence *character-backspace-character* counts as using one column position, assuming both characters each occupy a single column position. **fold** does not insert a newline character immediately before or after any backspace character.
Tab	Each tab character encountered advances the column position pointer to the next tab stop. Tab stops are set 8 columns apart at column positions 1, 9, 17, 25, 33, etc.
Carriage-return	The current count of line width is set to zero. **fold** does not insert a newline character immediately before or after any carriage-return character.

Note that **fold** may affect any underlining that is present.

Options

The **fold** command recognizes the following options and command-line arguments:

-b	Count *width* in bytes rather than in column positions.
-s	Break the line on the last blank character found before the specified number of column positions (or bytes). If none are found, break the line at the specified line length.
-w *width* **-***width*	Specify the maximum line length, in column positions (or bytes if **-b** is specified). The default value is 80. *width* should be a multiple of 8 if tabs are present, or the tabs should be expanded using **expand** before processing by **fold** (see *expand*(1)). The **-***width* option is obsolescent and may be removed in a future release.

EXTERNAL INFLUENCES

Environment Variables

LC_CTYPE determines the interpretation of text as single- and/or multi-byte characters.

LC_MESSAGES determines the language in which messages are displayed.

If **LC_CTYPE** or **LC_MESSAGES** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**.

If any internationalization variable contains an invalid setting, **fold** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported.

SEE ALSO

expand(1).

STANDARDS CONFORMANCE

fold: XPG4, POSIX.2

NAME

forder - convert file data order

SYNOPSIS

forder [**-a**] [**-l**] [**-n**] [*file* ...]

DESCRIPTION

The text orientation (mode) of a file can be right-to-left (non-Latin) or left-to-right (Latin). This text orientation can affect the way data is arranged in the file. The data arrangements that result are called screen order and keyboard order. **forder** converts the order of characters in the file from screen order to keyboard order or vice versa.

forder reads the concatenation of input files (or standard input if none are given) and produces on standard output a converted version of its input. If **-** appears as an input file name, **forder** reads standard input at that point (use **--** to delimit the end of options in such instances).

forder converts input files for all languages that are read from right-to-left. Unless the **-a** option is used, the command merely copies input files to standard output for languages that are read from left-to-right.

Options

forder recognizes the following options:

-a Convert file data order for languages read from left-to-right.

-l Identify the file as having been created in Latin mode.

-n Identify the file as having been created in non-Latin mode.

EXTERNAL INFLUENCES

Environment Variables

The **LANGOPTS** environment variable determines the mode and order of the file. The syntax of **LANGOPTS** is:

[*mode*] [*_order*]

where *mode* describes the mode of a file: **l** represents Latin mode, and **n** represents non-Latin mode. Non-Latin mode is assumed for values other than **l** and **n**. The *order* describes the data order of a file: **k** is keyboard, and **s** is screen. Keyboard order is assumed for values other than **k** and **s**. Mode information in **LANGOPTS** can be overridden from the command line.

The **LC_ALL** environment variable determines the direction of a language (left-to-right or right-to-left).

The **LC_NUMERIC** environment variable determines whether a language has alternative numbers.

The **LANG** environment variable determines the language in which messages are displayed.

International Code Set Support

Single-byte character code sets are supported.

EXAMPLES

The following command begins with *file1*, which exists in screen order, converts it to keyboard order, sorts the keyboard-ordered output, converts it back to screen order, and redirects the output to *file2*. Note that **-n** is given to inform **forder** that *file1* was created in non-Latin mode.

```
forder -n file1 | sort | forder -n > file2
```

WARNINGS

It is the user's responsibility to ensure that the **LANGOPTS** environment variable accurately reflects the status of the file.

If present, alternative numbers always have a left-to-right orientation.

The **forder** command is HP proprietary, not portable to other vendors' systems, and will not be provided in future HP-UX releases.

AUTHOR

forder was developed by HP.

SEE ALSO

environ(5), hpnls(5), strord(3C), nljust(1).

NAME

from - who is my mail from?

SYNOPSIS

from [**-s** *sender*] [*user*]

DESCRIPTION

from prints the mail header lines in your mailbox file to show who sent you mail. If *user* is specified, *user*'s mailbox is examined instead of your own. If the **-s** option is given, only headers of mail from *sender* are printed.

EXAMPLES

List header lines for all current mail in your mailbox that was sent by **ken**.

```
from -s ken
```

FILES

/var/mail/*

AUTHOR

from was developed by the University of California, Berkeley.

SEE ALSO

biff(1), mail(1), prmail(1).

NAME

ftio - faster tape I/O

SYNOPSIS

ftio -o|**-O** [**achpvxAELM**] [**-B** *blksize*] [**-D** *type*] [**-e** *extarg*] [**-K** *comment*] [**-L** *filelist*] [**-N** *datefile*] [**-S** *script*] [**-T** *tty*] [**-Z** *nobufs*] *tapedev* [*pathnames*] [**-F** *ignorenames*]

ftio -i|**-I** [**cdfmptuvxAEMPR**] [**-B** *blksize*] [**-S** *script*] [**-T** *tty*] [**-Z** *nobufs*] *tapedev* [*patterns*]

ftio -g [**v**] *tapedev* [*patterns*]

DESCRIPTION

ftio is a tool designed specifically for copying files to tape drives. It performs faster than either **cpio** or **tar** in comparable situations (see *cpio*(1) and *tar*(1)). **ftio** uses multiple processes (to read/write the file system and to write/read the tape device), with large amounts of memory sharing between processes as well as a large block size for reading and writing to the tape.

f

ftio is compatible with **cpio** in that output from **cpio** is always readable by **ftio**, and output from **ftio** is readable by **cpio**, except as explained in the "cpio Compatibility" section, later in the manpage.

ftio must be invoked with exactly one of the following options: **-o**, **-O**, **-i**, **-I**, or **-g**. The **-o** and **-O** options specify that **ftio** is writing "out" from file system to tape; the **-i** and **-I** options specify that **ftio** is writing "in" from tape to file system. The **-o**, **-O**, **-i**, and **-I** options can be followed by modifiers that must appear immediately after the option with no spaces between the option and the modifier, as in **ftio -idxE** (see Modifiers section below).

tapedev specifies the name of a device special file for the tape device to which the output is written. A device on a remote machine can be specified in the form

machine:*device_special_file*

ftio creates a server process from **/usr/sbin/rmt** on the remote machine to access the tape device. If **/usr/sbin/rmt** does not exist on the remote system, **ftio** creates a server process from **/etc/rmt**, on the remote machine to access the tape device.

Options

ftio recognizes the following options:

-o Copy (out) files from the file system to *tapedev*, including path name and status information. If *pathnames* are specified, **ftio** recursively descends *pathnames* looking for files, and copies those files to *tapedev*. If *pathnames* are not specified, **ftio** reads the standard input to obtain a list of path names to copy. **ftio** can copy to multiple tapes if required. For every tape used, **ftio** generates a tape header containing the current tape volume number, machine node name and type, operating system name, release and version numbers (all from the **uname()** system call; see *uname*(2)), username of the person issuing the **ftio** command, the time and date the command was executed, the number of consecutive times the current media has been used, a comment field, and other items used internally by **ftio**. The tape header is separated from the main body of the tape archive by an end-of-file mark. The tape header can be read by invoking **cat** with the device file name as the first argument (see *cat*(1)). Note, character and block device special files written with the **-o** option are not transportable to other HP-UX implementations.

-O Copy out files in the same way as **ftio -ocva**, when no modifiers are used with the **-O**. However, if the **.ftiorc** file exists in the user's home directory, **ftio** opens this file and scans for lines preceded by **O=**. Options defined on matching lines are passed to **ftio** as if they had been specified on the command line. See EXAMPLES section.

-i Extract (copy into the file system) files from *tapedev*, which is assumed to be a tape and the product of a previous **ftio -o** operation. Only files with names that match *patterns*, according to the rules of Pattern Matching Notation (see *regexp*(5)), are selected. In addition, a leading **!** within a pattern indicates that only those names that do *not* match the remainder of the pattern should be selected. Multiple *patterns* can be specified. If no *patterns* are specified, the default for *patterns* is ***** (that is, select all files). The extracted files are conditionally created and copied into the current directory tree, based upon the options described below. The permissions of the files are those of the previous **-o** operation.

-I Extract (copy into the file system) files in the same way as for **ftio -icdmv**, when no modifiers are used with the **-I**. However, if the **.ftiorc** file exists in the user's home directory, **ftio** opens this file, and scans for lines preceded by **I=**. Options defined on matching lines are passed to **ftio** as if they had been specified on the command line. See EXAMPLES section.

-g Read the file list in *tapedev*. If *patterns* is specified, only file names that match are printed. Note that file names are always preceded by the volume that **ftio** expected the file to be on when the file list was created; thus only the last volume is valid in this respect.

-e *extarg* Specifies the handling of any extent attributes of the file[s] to be archived. Extent attributes cannot be preserved when archiving files with **ftio.** *extarg* takes one of the following values:

warn	Issue a warning message and archive the file without extent attributes.
ignore	A file with extent attributes will be archived, without preserving the extent attributes and without issuing a warning message.
force	A file with extent attributes will not be archived and a warning message will be issued.

If **-e** is not specified, the default value for *extarg* is **warn**.

-B *blksize* Specify the size (in bytes) of blocks written to tape. This number can end with **k**, which specifies multiplication by 1024. The use of larger blocks generally improves performance and tape usage. The maximum allowable block size is limited by the tape drive used. A default of 16384 bytes is set because this is the maximum block size on most Hewlett-Packard tape drives.

-D *type* Descend a directory recursively, only if the file system to which it belongs is *type*, where *type* can be **hfs**, **vxfs**, or **nfs**.

-F *ignorenames*
Arguments following **-F** specify *patterns* that should not be copied to the tape. The same rules apply to *ignorenames* as to *patterns*; see the earlier description for **ftio -i**.

-K *comment* Specify a comment to be placed in the **ftio** tape header.

-L *filelist* Create a list of the files being backed up. *filelist* specifies the output file. If *pathnames* is specified, perform the file search and generate a list of files prior to actually commencing the backup. This list is then appended to the tape header of each tape in the backup as a list of files that **ftio** attempted to fit onto this tape. The last tape in the backup contains a catalog identifying where the files are in the archive set. If *pathnames* is not also specified, the file list is taken from standard input before the backup begins. In addition to generating file lists, the **-L** option implements tape checkpointing, allowing the backup to restart from a write failure on bad media.

-M Make fully compatible with **cpio**. That is, do not generate or expect tape headers and change the default block size to 5120 bytes. (See the cpio Compatibility section below.)

-N *datefile* Only files newer than the file specified in *datefile* are copied to tape.

-R Resynchronize automatically, when **ftio** goes out of phase. This is useful when restoring from a multi-tape backup from tapes other than the first. By default, **ftio** asks the user if resynchronization is required.

-S *script* Specify a command to be invoked every time a tape is completed in a multi-tape backup. The command is invoked by the Bourne shell (see *sh-bourne*(1) with the following arguments: *script tape_no user_name*. *script* is the string argument *script* specified with the **-S** option. *tape_no* is the number of the tape required, and *user_name* is the user who invoked **ftio**. Typically, the string *script* specifies a shell script which is used to notify the user that a tape change is required.

-T *tty* Specify alternative to **/dev/tty**. Normally **/dev/tty** is opened by **ftio** when terminal interaction is required.

-Z *nobufs* Specify the number of *blksize* chunks of memory to use as buffer space between the two processes, where *blksize* is the size of blocks written to the tape. More chunks is usually better, but a point is reached where no improvement is gained, and performance might deteriorate as buffer space is swapped out of main memory. A default value of 16 is set for *nobufs*, but using 32 or 64 might improve performance if your system is not heavily loaded. Best results are obtained when backups are performed with the system in single-user mode (see *shutdown*(1M)).

Modifiers

The following modifiers can be used with certain options as indicated in the SYNOPSIS:

a After files are copied to tape, reset their access time to appear as though the files were not accessed by **ftio**.

c Write header information in ASCII character form, for portability.

d When restoring files, create directories as needed.

f Copy in all files except those that match *patterns*.

h Archive the files to which symbolic links point, as if they were normal files or directories. By default, **ftio** archives the link itself.

m Retain previous file modification time and ownership of file. Restoring modification time does not apply to directories that are being restored.

p At the end of the backup, print the number of blocks transferred, the total time taken (excluding tape rewind and reel-change time), and the effective transfer rate calculated from these figures. These values are printed at the end of each tape if **p** is specified twice.

t Print only a table of contents of the input. No files are created, read, or copied.

u Copy unconditionally (by default, **ftio** does not replace a newer file with a older file of the same name).

v Be verbose. Print a list of file names and tape headers. When used with the **t** modifier, the table of contents looks the same as the output of the **ls -l** (ell) command (see *ls*(1)).

x Save or restore device special files. **ftio** uses *mknod*(2) to recreate these files during a restore operation. Thus, this modifier is restricted to users with appropriate privileges. This is intended for intrasystem (backup) use. Restoring device files onto a different system can be very dangerous.

A If copying from tape (**-i** or **-I** option), print all file names found on the tape archive, noting which files have been restored. This is useful when the user restores selected files, but wants to know which (if any) files are on the tape.

If copying to tape (**-o** or **-O** option), the **A** modifier suppresses warning messages regarding optional access control list entries. *ftio*(1) does not back up optional access control list entries in a file's access control list (see *acl*(5)). Normally, a warning message is printed for each file that has optional access control list entries.

E When archiving, store all files having absolute path names (that is, path names beginning with **/**) with path names relative to the root directory (in other words, remove the leading **/**). On restoration, any files in the archive that had an absolute path name before archiving are restored relative to the current directory.

L Same as the **-L** option, except that the file list is left in the current directory as the file **ftio.list**, instead of the file named in *filelist*.

P On restoration, use **prealloc()** to allocate disk space beforehand for the file (see *prealloc*(2)). This vastly improves the localization of file fragments.

When end-of-tape is reached, **ftio** invokes *script* if the **-S** option was specified, rewinds the current tape, then asks the user to mount the next tape.

To pass one or more metacharacters to **ftio** without having the shell expand them, protect them either by preceding each of them with a backslash (as in **/usr*)**, or enclosing them in protective single quotes (as in **'/usr*'**).

cpio Compatibility

`ftio` uses the same archive format as `cpio`. However, by default `ftio` creates tape headers and uses a tape block size of 16KB. `cpio` by default uses 512-byte blocks. When used with the `-B` option, `cpio` uses 5120 byte blocks. To achieve full compatibility with `cpio` in either input or output mode, the user should specify the `M` modifier. `ftio -oM` creates a single- or multi-tape archive that has no tape headers, and, by default, the same block size as `cpio -`[`o`|`i`]`B`. An archive created by a `cpio -oB` command can be restored using `ftio -iM`. If the `M` modifier of `ftio` is combined with a `-B 512` block-size specification, full compatibility with `cpio -`[`o`|`i`] (no `-B`) is achieved.

EXTERNAL INFLUENCES

Environment Variables

`LC_COLLATE` determines the collating sequence used in evaluating pattern matching notation for file name generation.

`LC_CTYPE` determines the characters matched by character class expressions in pattern matching notation.

`LC_TIME` determines the format and contents of date and time strings.

`LANG` determines the language in which messages are displayed.

If `LC_COLLATE, LC_CTYPE,` or `LC_TIME` is not specified in the environment or is set to the empty string, the value of `LANG` is used as a default for each unspecified or empty variable. If `LANG` is not specified or is set to the empty string, a default of C (see *lang*(5)) is used instead of `LANG`. If any internationalization variable contains an invalid setting, `ftio` behaves as if all internationalization variables are set to C. See *environ*(5).

International Code Set Support

Single-byte character code sets are supported.

EXAMPLES

Copy the entire contents of the file system (including special files) onto tape drive `/dev/rmt/c0t0d0BEST`:

```
ftio -ox /dev/rmt/c0t0d0BEST /
```

Restore all the files on `/dev/rmt/c0t0d0BEST`, relative to the current directory:

```
ftio -idxE /dev/rmt/c0t0d0BEST
```

List the contents of a backup set created using `ftio -o`. Note that use of the `v` modifier gives a more detailed listing, and displays the contents of tape headers.

```
ftio -itv /dev/rmt/c0t0d0BEST
```

Show how to use the `.ftiorc` file:

Assume a `.ftiorc` file exists in the user's home directory and contains the following:

```
# Sample .ftiorc file.
I= cdmuvEpp -B 16k -S /usr/local/bin/ftio.change
O= cavEpp -Z 8 -B 16k -S /usr/local/bin/ftio.change
```

Invoke `ftio` with the following command line to back up the user's home directory and the operating system commands directory:

```
ftio -O /dev/rmt/c0t0d0BEST /home/my_home /usr/sbin
```

Specifying the `-O` option causes `ftio` to check the `.ftiorc` file for additional options. In this case, character headers are generated, access times are reset, a listing of the files copied are printed to standard output, all file names are copied to `/dev/rmt/c0t0d0BEST` with path names relative to `/`, performance data is printed when the backup is complete (and at every tape change), and, if the backup goes beyond one media the script, `/usr/local/bin/ftio.change` is invoked by `ftio` after each media is completed.

WARNINGS

Because of industry standards and interoperability goals, `ftio` does not support the archival of files larger than 2GB or files that have user/group IDs greater than 60K. Files with user/group IDs greater than 60K are archived and restored under the user/group ID of the current process.

ftio operates using System V shared memory and semaphores. The resources committed to these functions are not freed automatically by the system when the process terminates. **ftio** does this only when it terminates normally, or when it terminates after receiving one the following signals: **SIGHUP**, **SIGINT**, **SIGTERM**. Any other signal is handled in the default manner described by *signal*(2). Note that the behavior for **SIGKILL** is to terminate the process without delay. Thus, if **ftio** receives a **SIGKILL** signal (as might be produced by the indiscriminate use of **kill -9** (see *kill*(1)), system resources used for shared memory and semaphores are not returned to the system. If it becomes necessary to terminate an invocation of **ftio**, use **kill -15** instead. Current system usage of shared memory and semaphores can be checked using the **ipcs** command (see *ipcs*(1)). Committed resources can be removed using **ipcrm** (see *ipcrm*(1)).

AUTHOR

ftio was developed by HP.

SEE ALSO

cpio(1), find(1), ipcs(1), ipcrm(1), kill(1), ls(1), rmt(1M), mknod(2), prealloc(2), signal(2), uname(2), acl(5), environ(5), lang(5), regexp(5), mt(7).

NAME

ftp - file transfer program

SYNOPSIS

ftp [**-g**] [**-i**] [**-n**] [**-v**] [**-B** *size*] [*server-host*]

DESCRIPTION

ftp is a user interface to the File Transfer Protocol. **ftp** copies files over a network connection between the local "client" host and a remote "server" host. **ftp** runs on the client host.

Options

The **ftp** command supports the following options:

-g Disable file name "globbing"; see the **glob** command, below. By default, when this option is not specified, globbing is enabled.

-i Disable interactive prompting by multiple-file commands; see the **prompt** command, below. By default, when this option is not specified, prompting is enabled.

-n Disable "auto-login"; see the **open** command, below. By default, when this option is not specified, auto-login is enabled.

-v Enable verbose output; see the **verbose** command, below. If this option is not specified, **ftp** displays verbose output only if the standard input is associated with a terminal.

-B Set the buffer size of the data socket to *size* blocks of 1024 bytes. The valid range for *size* is an integer from 1 to 64 (default is 56).

Note: A large buffer size will improve the performance of **ftp** on fast links (e.g., FDDI), but may cause long connection times on slow links (e.g., X.25).

The name of the server host that **ftp** communicates with can be specified on the command line. If the server host is specified, **ftp** immediately opens a connection to the server host; see the **open** command, below. Otherwise, **ftp** waits for commands from the user.

File Transfer Protocol specifies file transfer parameters for *type*, *mode*, *form*, and *struct*. **ftp** supports the **ASCII**, **binary**, and **tenex** File Transfer Protocol *types*. **ASCII** is the default FTP *type*. (It should be noted though that, whenever **ftp** establishes a connection between two similar systems, it switches automatically to the more efficient **binary** type.) **ftp** supports only the default values for the file transfer parameters *mode* which defaults to **stream**, *form* which defaults to **non-print**, and *struct* which defaults to **file**.

COMMANDS

ftp supports the following commands. Command arguments with embedded spaces must be enclosed in quotes (for example, "argument with embedded spaces").

![*command* [*args*]]
Invoke a shell on the local host. The **SHELL** environment variable specifies which shell program to invoke. **ftp** invokes **/usr/bin/sh** if **SHELL** is undefined. If *command* is specified, the shell executes it and returns to **ftp**. Otherwise, an interactive shell is invoked. When the shell terminates, it returns to **ftp**.

$ *macro-name* [*args*]
Execute the macro *macro-name* that was defined with the **macdef** command. Arguments are passed to the macro unglobbed.

account [*passwd*]
Supply a supplemental password required by a remote system for access to resources once a login has been successfully completed. If no argument is included, the user is prompted for an account password in a non-echoing input mode.

append *local-file* [*remote-file*]
Copy *local-file* to the end of *remote-file*. If *remote-file* is left unspecified, the local file name is used in naming the remote file after being altered by any *ntrans* or *nmap* setting.

ascii
Set the file transfer *type* to network ASCII. This is the default type.

bell
Sound a bell after each file transfer completes.

binary
Set the file transfer *type* to **binary**.

bye Close the connection to the server host if a connection was open, and exit. Typing an end-of-file (EOF) character also terminates and exits the session.

case
Toggle remote computer file name case mapping during **mget** commands. When **case** is on (the default is off), remote computer file names with all letters in uppercase are written in the local directory with the letters mapped to lowercase.

cd *remote-directory*
Set the working directory on the server host to *remote-directory*.

cdup
Set the working directory on the server host to the parent of the current remote working directory.

chmod *mode file-name*
Change the permission modes of the file *file-name* on the remote system to *mode*.

close
Terminate the connection to the server host. The **close** command does not exit **ftp**. Any defined macros are erased.

cr Toggle carriage return stripping during **ascii** type file retrieval. Records are denoted by a carriage-return/line-feed sequence during **ascii** type file transfer. When **cr** is on (the default), carriage returns are stripped from this sequence to conform with the UNIX single line-feed record delimiter. Records on non-UNIX remote systems may contain single line-feeds; when an **ascii** type transfer is made, these line-feeds can be distinguished from a record delimiter only when **cr** is off.

delete *remote-file*
Delete *remote-file*. The *remote-file* can be an empty directory. No globbing is done.

dir [*remote-directory*] [*local-file*]
Write a *remote-directory* listing to standard output or optionally to *local-file*. If neither *remote-directory* nor *local-file* is specified, list the remote working directory to standard output. If interactive prompting is on, **ftp** prompts the user to verify that the last argument is indeed the target file for **dir** output. Globbing characters are always expanded.

disconnect
A synonym for **close**.

form *format*
Set the file transfer *form* to *format*. The only supported format is **non-print**

get *remote-file* [*local-file*]
Copy *remote-file* to *local-file*. If *local-file* is unspecified, **ftp** uses the specified *remote-file* name as the *local-file* name, subject to alteration by the current *case*, *ntrans*, and *nmap* settings.

glob
Toggle file name globbing. When file name globbing is enabled, **ftp** expands *csh*(1) metacharacters in file and directory names. These characters are *****, **?**, **[**, **]**, **~**, **{**, and **}**. The server host expands remote file and directory names. Globbing metacharacters are always expanded for the **ls** and **dir** commands. If globbing is enabled, metacharacters are also expanded for the multiple-file commands **mdelete**, **mdir**, **mget**, **mls**, and **mput.**

hash
Toggle printing of a hash-sign (**#**) for each 1024 bytes transferred.

help [*command*]
Print an informative message about the **ftp** command called *ftp-command*. If *ftp-command* is unspecified, print a list of all **ftp** commands.

idle [*seconds*]
Set the inactivity timer on the remote server to *seconds* seconds. If *seconds* is omitted, **ftp** prints the current inactivity timer.

lcd [*local-directory*]
Set the local working directory to *local-directory*. If *local-directory* is unspecified, set the local working directory to the user's local home directory.

ls [*remote-directory*] [*local-file*]
Write a listing of *remote-directory* to *local-file*. The listing includes any system-dependent information that the server chooses to include; for example, most UNIX systems produce output from the command **ls -l** (see also **nlist**). If neither *remote-directory* nor *local-file* is specified, list the remote working directory. If globbing is enabled, globbing metacharacters are expanded.

macdef *macro-name*
Define a macro. Subsequent lines are stored as the macro *macro-name*; an empty input line terminates macro input mode. There is a limit of 16 macros and 4096 total characters in all defined macros. Macros remain defined until a **close** command is executed. The macro processor interprets **$** and **\\** as special characters. A **$** followed by a number (or numbers) is replaced by the corresponding argument on the macro invocation command line. A **$** followed by an *i* signals to the macro processor that the executing macro is to be looped. On the first pass **$***i* is replaced by the first argument on the macro invocation command line, on the second pass it is replaced by the second argument, and so on. A \ followed by any character is replaced by that character. Use the \ to prevent special treatment of the **$**.

mdelete [*remote-files*]
Delete *remote-files*. If globbing is enabled, globbing metacharacters are expanded.

mdir *remote-files local-file*
Write a listing of *remote-files* to *local-file*. If globbing is enabled, globbing metacharacters are expanded. If interactive prompting is on, **ftp** prompts the user to verify that the last argument is indeed the target local file for **mdir** output.

mget *remote-files*
Copy *remote-files* to the local system. If globbing is enabled, globbing metacharacters are expanded. The resulting local file names are processed according to *case*, *ntrans*, and *nmap* settings.

mkdir *directory-name*
Create remote *directory-name*.

mls *remote-files local-file*
Write an abbreviated listing of *remote-files* to *local-file*. If globbing is enabled, globbing metacharacters are expanded. If interactive prompting is *on*, **ftp** prompts the user to verify that the last argument is indeed the target local file for **mls** output.

mode [*mode-name*]
Set the FTP file transfer *mode* to *mode-name*. The only supported mode is **stream**.

modtime *remote-file*
Show the last modification time of *remote-file*.

mput *local-files*
Copy *local-files* from the local system to the remote system. The remote files have the same name as the local files processed according to *ntrans* and *nmap* settings. If globbing is enabled, globbing characters are expanded.

newer *file-name*
Get the file only if the modification time of the remote file is more recent that the file on the current system. If the file does not exist on the current system, the remote file is considered *newer*. Otherwise, this command is identical to **get**.

nlist [*remote-directory*] [*local-file*]
Write an abbreviated listing of *remote-directory* to *local-file*. If *remote-directory* is left unspecified, the current working directory is used. If interactive prompting is on, **ftp** prompts the user to verify that the last argument is indeed the target local file for **nlist** output.

nmap [*inpattern outpattern*]
Set or unset the filename mapping mechanism. If no arguments are specified, the filename mapping mechanism is unset. If arguments are specified, remote filenames are mapped during **mput** commands and **put** commands issued without a specified remote target filename. If arguments are specified, local filenames are mapped during **mget** commands and **get** commands issued without a specified local target filename. This command is useful when connecting to a non-UNIX remote computer with different file naming conventions or practices. The mapping follows the pattern set by

inpattern and *outpattern*. *inpattern* is a template for incoming filenames (which may have already been processed according to the **ntrans** and **case** settings). Variable templating is accomplished by including the sequences **$1**, **$2**, ..., **$9** in *inpattern*. Use **\\** to prevent this special treatment of the **$** character. All other characters are treated literally, and are used to determine the **nmap** *inpattern* variable values. For example, given *inpattern* **$1.$2** and the remote file name **mydata.data**, **$1** would have the value **mydata**, and **$2** would have the value **data**. The *outpattern* determines the resulting mapped filename. The sequences **$1**, **$2**, ..., **$9** are replaced by any value resulting from the *inpattern* template. The sequence **$0** is replaced by the original filename. Additionally, the sequence **[***seq1***,***seq2***]** is replaced by *seq1* if *seq1* is not a null string; otherwise it is replaced by *seq2*. For example, the command **nmap $1.$2.$3 [$1,$2].[$2,file]** would yield the output filename **myfile.data** for input filenames **myfile.data** and **myfile.data.old**, **myfile.file** for the input filename **myfile**, and **myfile.myfile** for the input filename **.myfile**. Spaces can be included in *outpattern*, as in the example: **nmap $1 | sed "s/ *$//" > $1 .** Use the **\\** character to prevent special treatment of the **$**, **[**, **]**, and **,** characters.

f

ntrans [*inchars* [*outchars*]]
Set or unset the filename character translation mechanism. If no arguments are specified, the filename character translation mechanism is unset. If arguments are specified, characters in remote filenames are translated during **mput** commands and **put** commands issued without a specified remote target filename. If arguments are specified, characters in local filenames are translated during **mget** commands and **get** commands issued without a specified local target filename. This command is useful when connecting to a non-UNIX remote computer with different file naming conventions or practices. Characters in a filename matching a character in *inchars* are replaced with the corresponding character in *outchars*. If the character's position in *inchars* is longer than the length of *outchars*, the character is deleted from the file name.

open *server-host* [*port-number*]
Establish a connection to *server-host*, using *port-number* (if specified). If *auto-login* is enabled, **ftp** attempts to log into the server host.

passive
Toggle passive mode of transfer. By default, the passive mode of transfer is disabled. This command enables the server to specify the data port for the ftp transfer.

prompt
Toggle interactive prompting. By default, **ftp** prompts the user for a yes or no response for each output file during multiple-file commands. If interactive prompting is disabled, **ftp** performs the command for all specified files.

proxy *ftp-command*
Execute an **ftp** command on a secondary control connection. This command allows simultaneous connection to two remote FTP servers for transferring files between the two servers. The first **proxy** command should be an **open**, to establish the secondary control connection. Enter the command **proxy ?** to see other FTP commands executable on the secondary connection. The following commands behave differently when prefaced by **proxy**: **open** does not define new macros during the auto-login process, **close** does not erase existing macro definitions, **get** and **mget** transfer files from the host on the primary control connection to the host on the secondary control connection, and **put**, **mput**, and **append** transfer files from the host on the secondary control connection to the host on the primary control connection. Third party file transfers depend upon support of the FTP protocol **PASV** command by the server on the secondary control connection.

put *local-file* [*remote-file*]
Copy *local-file* to *remote-file*. If *remote-file* is unspecified, **ftp** assigns the *local-file* name, processed according to any *ntrans* or *nmap* settings, to the *remote-file* name.

pwd Write the name of the remote working directory to *stdout*.

quit
A synonym for **bye**.

quote *arguments*
Send *arguments*, verbatim, to the server host. See *ftpd*(1M).

recv *remote-file* [*local-file*]
A synonym for **get**.

reget *remote-file* [*local-file*]
: **reget** acts like **get**, except that if *local-file* exists and is smaller than *remote-file*, *local-file* is presumed to be a partially transferred copy of *remote-file* and the transfer is continued from the apparent point of failure. This command is useful when transferring very large files over networks that tend to drop connections.

rhelp [*command-name*]
: Request help from the server host. If *command-name* is specified, supply it to the server. See *ftpd*(1M).

rstatus [*file-name*]
: With no arguments, show status of remote machine. If *file-name* is specified, show status of *file-name* on remote machine.

rename *remote-from remote-to*
: Rename *remote-from*, which can be either a file or a directory, to *remote-to*.

reset
: Clear reply queue. This command re-synchronizes command/reply sequencing with the remote FTP server. Resynchronization may be necessary following a violation of the FTP protocol by the remote server.

restart *marker*
: Restart the immediately following **get** or **put** at the indicated *marker*. On UNIX systems, marker is usually a byte offset into the file.

rmdir *remote-directory*
: Delete *remote-directory*. *remote-directory* must be an empty directory.

runique
: Toggle storing of files on the local system with unique filenames. If a file already exists with a name equal to the target local filename for a **get** or **mget** command, a **.1** is appended to the name. If the resulting name matches another existing file, a **.2** is appended to the original name. If this process continues up to **.99**, an error message is printed, and the transfer does not take place. **ftp** reports the unique filename. Note that **runique** does not affect local files generated from a shell command (see below). The default value is **off**.

send *local-file* [*remote-file*]
: A synonym for **put**.

sendport
: Toggle the use of **PORT** commands. By default, **ftp** attempts to use a **PORT** command when establishing a connection for each data transfer. If the **PORT** command fails, **ftp** uses the default data port. When the use of **PORT** commands is disabled, **ftp** makes no attempt to use **PORT** commands for each data transfer. This is useful for certain FTP implementations that ignore **PORT** commands but (incorrectly) indicate that they've been accepted. See *ftpd*(1M). Turning **sendport** off may cause delays in the execution of commands.

site *arguments*
: Send *arguments*, verbatim, to the server host as a **SITE** command. See *ftpd*(1M).

size *remote-file*
: Show the size of *remote-file*.

status
: Show the current status of **ftp**.

struct [*struct-name*]
: Set the FTP file transfer *struct* to *struct-name*. The only supported *struct* is **file**.

sunique
: Toggle storing of files on remote machine under unique file names. The remote server reports the unique name. By default, **sunique** is **off**.

system
: Show the type of operating system running on the remote machine.

tenex
: Set the FTP file transfer *type* to **tenex**.

f

type [*type-name*]
Set the FTP file transfer *type* to *type-name*. If *type-name* is unspecified, write the current *type* to *stdout*. **Ascii**, **binary**, and **tenex** are the *type*s currently supported.

umask [*newmask*]
Set the default umask on the remote server to *newmask*. If *newmask* is omitted, the current umask is printed.

user *user-name* [*password*] [*account*]
Log into the server host on the current connection, which must already be open. A **.netrc** file in the user's local home directory can provide the *user-name*, *password*, and optionally the *account*; see *netrc*(4). Otherwise **ftp** prompts the user for this information. The HP-UX FTP server does not require an *account*. For security reasons, **ftp** always requires a password. It does not log into remote accounts that do not have a password.

verbose
Toggle verbose output. If verbose output is enabled, **ftp** displays responses from the server host, and when a file transfer completes it reports statistics regarding the efficiency of the transfer.

? [*command*]
A synonym for the **help** command. Prints the **help** information for the specified *command*.

Aborting A File Transfer

To abort a file transfer, use the terminal interrupt key (usually **Ctrl-C**). Sending transfers are halted immediately. **ftp** halts incoming (receive) transfers by first sending a FTP protocol **ABOR** command to the remote server, then discarding any further received data. The speed at which this is accomplished depends upon the remote server's support for **ABOR** processing. If the remote server does not support the **ABOR** command, an **ftp>** prompt does not appear until the remote server completes sending the requested file.

The terminal interrupt key sequence is ignored while **ftp** awaits a reply from the remote server. A long delay in this mode may result from the **ABOR** processing described above, or from unexpected behavior by the remote server, including violations of the FTP protocol. If the delay results from unexpected remote server behavior, the local **ftp** program must be killed manually.

File Naming Conventions

Files specified as arguments to **ftp** commands are processed according to the following rules.

- If the file name **-** is specified, **ftp** uses the standard input (for reading) or standard output (for writing).
- If the first character of the file name is **|**, **ftp** interprets the remainder of the argument as a shell command. **ftp** forks a shell, using **popen()** (see *popen*(3S)) with the supplied argument, and reads (writes) from standard output (standard input). If the shell command includes spaces, the argument must be quoted, as in:

  ```
  "| ls -lt".
  ```

  Some useful examples of this mechanism are:

  ```
  ls . "| more".
  ```

  The above command lists the files in the current directory page by page.

  ```
  put  "| tail -20 loc_file" rem_file.
  ```

  This command copies the last twenty lines of the local file "loc_file" to the remote system as "rem_file".
- Otherwise, if globbing is enabled, **ftp** expands local file names according to the rules used by the C shell (see *csh*(1)); see the **glob** command, below. If the **ftp** command expects a single local file (e.g., **put**), only the first filename generated by the globbing operation is used.
- For **mget** commands and **get** commands with unspecified local file names, the local filename is named the same as the remote filename, which may be altered by a **case**, **ntrans**, or **nmap** setting. The resulting filename may then be altered if **runique** is on.
- For **mput** commands and **put** commands with unspecified remote file names, the remote filename is named the same as the local filename, which may be altered by a **ntrans** or **nmap** setting. The resulting filename may then be altered by the remote server if **sunique** is on.

WARNINGS

Correct execution of many commands depends upon proper behavior by the remote server.

DIAGNOSTICS

`Error! could not retrieve authentication type.`

`Please notify sys admin.`

There are two authentication mechanisms used by **`ftp`**. One authentication mechanism is based on Kerberos and the other is not. The type of authentication mechanism is obtained from a system file which is updated by **`inetsvcs_sec`** (see *inetsvcs_sec*(1M)). If the system file does not contain known authentication types, the above error is displayed.

AUTHOR

`ftp` was developed by the University of California, Berkeley.

SEE ALSO

csh(1), rcp(1), ftpd(1M), inetsvcs_sec(1M), netrc(4), ftpusers(4), hosts(4).

NAME

ftp - file transfer program

SYNOPSIS

ftp [**-g**] [**-i**] [**-n**] [**-P**] [**-v**] [**-B** *size*] [*server-host*]

DESCRIPTION

ftp is a user interface to the File Transfer Protocol. **ftp** copies files over a network connection between the local “client” host and a remote “server” host. **ftp** runs on the client host.

Options

The **ftp** command supports the following options:

-g Disable file name “globbing”; see the **glob** command, below. By default, when this option is not specified, globbing is enabled.

-i Disable interactive prompting by multiple-file commands; see the **prompt** command, below. By default, when this option is not specified, prompting is enabled.

-P Disables Kerberos authentication and authorization. Only applicable in a secure environment based on Kerberos V5. When this option is specified, a password is required and the password is sent across the network in a readable form. By default, if this option is not specified, a password is not required and Kerberos authentication and authorization takes place instead. See *sis*(5).

-n Disable “auto-login”; see the **open** command, below. By default, when this option is not specified, auto-login is enabled.

-v Enable verbose output; see the **verbose** command, below. If this option is not specified, **ftp** displays verbose output only if the standard input is associated with a terminal.

-B Set the buffer size of the data socket to *size* blocks of 1024 bytes. The valid range for *size* is an integer from 1 to 64 (default is 56).

Note: A large buffer size will improve the performance of **ftp** on fast links (e.g., FDDI), but may cause long connection times on slow links (e.g., X.25).

The name of the server host that **ftp** communicates with can be specified on the command line. If the server host is specified, **ftp** immediately opens a connection to the server host; see the **open** command, below. Otherwise, **ftp** waits for commands from the user.

File Transfer Protocol specifies file transfer parameters for *type*, *mode*, *form*, and *struct*. **ftp** supports the **ASCII**, **binary**, and **tenex** File Transfer Protocol *types*. **ASCII** is the default FTP *type*. (It should be noted though that, whenever **ftp** establishes a connection between two similar systems, it switches automatically to the more efficient **binary** type.) **ftp** supports only the default values for the file transfer parameters *mode* which defaults to **stream**, *form* which defaults to **non-print**, and *struct* which defaults to **file**.

COMMANDS

ftp supports the following commands. Command arguments with embedded spaces must be enclosed in quotes (for example, "argument with embedded spaces").

![*command* [*args*]]
Invoke a shell on the local host. The **SHELL** environment variable specifies which shell program to invoke. **ftp** invokes **/usr/bin/sh** if **SHELL** is undefined. If *command* is specified, the shell executes it and returns to **ftp**. Otherwise, an interactive shell is invoked. When the shell terminates, it returns to **ftp**.

$ *macro-name* [*args*]
Execute the macro *macro-name* that was defined with the **macdef** command. Arguments are passed to the macro unglobbed.

account [*passwd*]
Supply a supplemental password required by a remote system for access to resources once a login has been successfully completed. If no argument is included, the user is prompted for an account password in a non-echoing input mode.

append *local-file* [*remote-file*]
Copy *local-file* to the end of *remote-file*. If *remote-file* is left unspecified, the local file name is used in naming the remote file after being altered by any *ntrans* or *nmap* setting.

ascii
Set the file transfer *type* to network ASCII. This is the default type.

bell
Sound a bell after each file transfer completes.

binary
Set the file transfer *type* to **binary**.

bye Close the connection to the server host if a connection was open, and exit. Typing an end-of-file (EOF) character also terminates and exits the session.

case
Toggle remote computer file name case mapping during **mget** commands. When **case** is on (the default is off), remote computer file names with all letters in uppercase are written in the local directory with the letters mapped to lowercase.

cd *remote-directory*
Set the working directory on the server host to *remote-directory*.

cdup
Set the working directory on the server host to the parent of the current remote working directory.

chmod *mode file-name*
Change the permission modes of the file *file-name* on the remote system to *mode*.

close
Terminate the connection to the server host. The **close** command does not exit **ftp**. Any defined macros are erased.

cr Toggle carriage return stripping during **ascii** type file retrieval. Records are denoted by a carriage-return/line-feed sequence during **ascii** type file transfer. When **cr** is on (the default), carriage returns are stripped from this sequence to conform with the UNIX single line-feed record delimiter. Records on non-UNIX remote systems may contain single line-feeds; when an **ascii** type transfer is made, these line-feeds can be distinguished from a record delimiter only when **cr** is off.

delete *remote-file*
Delete *remote-file*. The *remote-file* can be an empty directory. No globbing is done.

dir [*remote-directory*] [*local-file*]
Write a *remote-directory* listing to standard output or optionally to *local-file*. If neither *remote-directory* nor *local-file* is specified, list the remote working directory to standard output. If interactive prompting is on, **ftp** prompts the user to verify that the last argument is indeed the target file for **dir** output. Globbing characters are always expanded.

disconnect
A synonym for **close**.

form *format*
Set the file transfer *form* to *format*. The only supported format is **non-print**

get *remote-file* [*local-file*]
Copy *remote-file* to *local-file*. If *local-file* is unspecified, **ftp** uses the specified *remote-file* name as the *local-file* name, subject to alteration by the current *case*, *ntrans*, and *nmap* settings.

glob
Toggle file name globbing. When file name globbing is enabled, **ftp** expands *csh*(1) metacharacters in file and directory names. These characters are *****, **?**, **[**, **]**, **~**, **{**, and **}**. The server host expands remote file and directory names. Globbing metacharacters are always expanded for the **ls** and **dir** commands. If globbing is enabled, metacharacters are also expanded for the multiple-file commands **mdelete**, **mdir**, **mget**, **mls**, and **mput.**

hash
Toggle printing of a hash-sign (**#**) for each 1024 bytes transferred.

help [*command*]
Print an informative message about the **ftp** command called *ftp-command*. If *ftp-command* is unspecified, print a list of all **ftp** commands.

idle [*seconds*]
Set the inactivity timer on the remote server to *seconds* seconds. If *seconds* is omitted, **ftp** prints

the current inactivity timer.

lcd [*local-directory*]
Set the local working directory to *local-directory*. If *local-directory* is unspecified, set the local working directory to the user's local home directory.

ls [*remote-directory*] [*local-file*]
Write a listing of *remote-directory* to *local-file*. The listing includes any system-dependent information that the server chooses to include; for example, most UNIX systems produce output from the command **ls -l** (see also **nlist**). If neither *remote-directory* nor *local-file* is specified, list the remote working directory. If globbing is enabled, globbing metacharacters are expanded.

macdef *macro-name*
Define a macro. Subsequent lines are stored as the macro *macro-name*; an empty input line terminates macro input mode. There is a limit of 16 macros and 4096 total characters in all defined macros. Macros remain defined until a **close** command is executed. The macro processor interprets **$** and **\\** as special characters. A **$** followed by a number (or numbers) is replaced by the corresponding argument on the macro invocation command line. A **$** followed by an *i* signals to the macro processor that the executing macro is to be looped. On the first pass **$***i* is replaced by the first argument on the macro invocation command line, on the second pass it is replaced by the second argument, and so on. A \ followed by any character is replaced by that character. Use the \ to prevent special treatment of the **$**.

f

mdelete [*remote-files*]
Delete *remote-files*. If globbing is enabled, globbing metacharacters are expanded.

mdir *remote-files local-file*
Write a listing of *remote-files* to *local-file*. If globbing is enabled, globbing metacharacters are expanded. If interactive prompting is on, **ftp** prompts the user to verify that the last argument is indeed the target local file for **mdir** output.

mget *remote-files*
Copy *remote-files* to the local system. If globbing is enabled, globbing metacharacters are expanded. The resulting local file names are processed according to *case*, *ntrans*, and *nmap* settings.

mkdir *directory-name*
Create remote *directory-name*.

mls *remote-files local-file*
Write an abbreviated listing of *remote-files* to *local-file*. If globbing is enabled, globbing metacharacters are expanded. If interactive prompting is *on*, **ftp** prompts the user to verify that the last argument is indeed the target local file for **mls** output.

mode [*mode-name*]
Set the FTP file transfer *mode* to *mode-name*. The only supported mode is **stream**.

modtime *remote-file*
Show the last modification time of *remote-file*.

mput *local-files*
Copy *local-files* from the local system to the remote system. The remote files have the same name as the local files processed according to *ntrans* and *nmap* settings. If globbing is enabled, globbing characters are expanded.

newer *file-name*
Get the file only if the modification time of the remote file is more recent that the file on the current system. If the file does not exist on the current system, the remote file is considered *newer*. Otherwise, this command is identical to **get**.

nlist [*remote-directory*] [*local-file*]
Write an abbreviated listing of *remote-directory* to *local-file*. If *remote-directory* is left unspecified, the current working directory is used. If interactive prompting is on, **ftp** prompts the user to verify that the last argument is indeed the target local file for **nlist** output.

nmap [*inpattern outpattern*]
Set or unset the filename mapping mechanism. If no arguments are specified, the filename mapping mechanism is unset. If arguments are specified, remote filenames are mapped during **mput** commands and **put** commands issued without a specified remote target filename. If arguments are specified, local filenames are mapped during **mget** commands and **get** commands issued without a

specified local target filename. This command is useful when connecting to a non-UNIX remote computer with different file naming conventions or practices. The mapping follows the pattern set by *inpattern* and *outpattern*. *inpattern* is a template for incoming filenames (which may have already been processed according to the **ntrans** and **case** settings). Variable templating is accomplished by including the sequences **$1**, **$2**, ..., **$9** in *inpattern*. Use **** to prevent this special treatment of the **$** character. All other characters are treated literally, and are used to determine the **nmap** *inpattern* variable values. For example, given *inpattern* **$1.$2** and the remote file name **mydata.data**, **$1** would have the value **mydata**, and **$2** would have the value **data**. The *outpattern* determines the resulting mapped filename. The sequences **$1**, **$2**, ..., **$9** are replaced by any value resulting from the *inpattern* template. The sequence **$0** is replaced by the original filename. Additionally, the sequence **[***seq1***,***seq2***]** is replaced by *seq1* if *seq1* is not a null string; otherwise it is replaced by *seq2*. For example, the command **nmap $1.$2.$3 [$1,$2].[$2,file]** would yield the output filename **myfile.data** for input filenames **myfile.data** and **myfile.data.old**, **myfile.file** for the input filename **myfile**, and **myfile.myfile** for the input filename **.myfile**. Spaces can be included in *outpattern*, as in the example: **nmap $1 | sed "s/ *$//" > $1 .** Use the **** character to prevent special treatment of the **$**, **[**, **]**, and **,** characters.

ntrans [*inchars* [*outchars*]]
Set or unset the filename character translation mechanism. If no arguments are specified, the filename character translation mechanism is unset. If arguments are specified, characters in remote filenames are translated during **mput** commands and **put** commands issued without a specified remote target filename. If arguments are specified, characters in local filenames are translated during **mget** commands and **get** commands issued without a specified local target filename. This command is useful when connecting to a non-UNIX remote computer with different file naming conventions or practices. Characters in a filename matching a character in *inchars* are replaced with the corresponding character in *outchars*. If the character's position in *inchars* is longer than the length of *outchars*, the character is deleted from the file name.

open *server-host* [*port-number*]
Establish a connection to *server-host*, using *port-number* (if specified). If *auto-login* is enabled, **ftp** attempts to log into the server host.

prompt
Toggle interactive prompting. By default, **ftp** prompts the user for a yes or no response for each output file during multiple-file commands. If interactive prompting is disabled, **ftp** performs the command for all specified files.

put *local-file* [*remote-file*]
Copy *local-file* to *remote-file*. If *remote-file* is unspecified, **ftp** assigns the *local-file* name, processed according to any *ntrans* or *nmap* settings, to the *remote-file* name.

pwd Write the name of the remote working directory to *stdout*.

quit
A synonym for **bye**.

quote *arguments*
Send *arguments*, verbatim, to the server host. See *ftpd*(1M).

recv *remote-file* [*local-file*]
A synonym for **get**.

reget *remote-file* [*local-file*]
reget acts like **get**, except that if *local-file* exists and is smaller than *remote-file*, *local-file* is presumed to be a partially transferred copy of *remote-file* and the transfer is continued from the apparent point of failure. This command is useful when transferring very large files over networks that tend to drop connections.

rhelp [*command-name*]
Request help from the server host. If *command-name* is specified, supply it to the server. See *ftpd*(1M).

rstatus [*file-name*]
With no arguments, show status of remote machine. If *file-name* is specified, show status of *file-name* on remote machine.

rename *remote-from remote-to*
Rename *remote-from*, which can be either a file or a directory, to *remote-to*.

reset
Clear reply queue. This command re-synchronizes command/reply sequencing with the remote FTP server. Resynchronization may be necessary following a violation of the FTP protocol by the remote server.

restart *marker*
Restart the immediately following **get** or **put** at the indicated *marker*. On UNIX systems, marker is usually a byte offset into the file.

rmdir *remote-directory*
Delete *remote-directory*. *remote-directory* must be an empty directory.

runique
Toggle storing of files on the local system with unique filenames. If a file already exists with a name equal to the target local filename for a **get** or **mget** command, a **.1** is appended to the name. If the resulting name matches another existing file, a **.2** is appended to the original name. If this process continues up to **.99**, an error message is printed, and the transfer does not take place. **ftp** reports the unique filename. Note that **runique** does not affect local files generated from a shell command (see below). The default value is **off**.

send *local-file* [*remote-file*]
A synonym for **put**.

sendport
Toggle the use of **PORT** commands. By default, **ftp** attempts to use a **PORT** command when establishing a connection for each data transfer. If the **PORT** command fails, **ftp** uses the default data port. When the use of **PORT** commands is disabled, **ftp** makes no attempt to use **PORT** commands for each data transfer. This is useful for certain FTP implementations that ignore **PORT** commands but (incorrectly) indicate that they've been accepted. See *ftpd*(1M). Turning **sendport** off may cause delays in the execution of commands.

site *arguments*
Send *arguments*, verbatim, to the server host as a **SITE** command. See *ftpd*(1M).

size *remote-file*
Show the size of *remote-file*.

status
Show the current status of **ftp**.

struct [*struct-name*]
Set the FTP file transfer *struct* to *struct-name*. The only supported *struct* is **file**.

sunique
Toggle storing of files on remote machine under unique file names. The remote server reports the unique name. By default, **sunique** is **off**.

system
Show the type of operating system running on the remote machine.

tenex
Set the FTP file transfer *type* to **tenex**.

type [*type-name*]
Set the FTP file transfer *type* to *type-name*. If *type-name* is unspecified, write the current *type* to *stdout*. **Ascii**, **binary**, and **tenex** are the *type*s currently supported.

umask [*newmask*]
Set the default umask on the remote server to *newmask*. If *newmask* is omitted, the current umask is printed.

user *user-name* [*password*] [*account*]
Log into the server host on the current connection, which must already be open. A **.netrc** file in the user's local home directory can provide the *user-name*, *password*, and optionally the *account*; see *netrc*(4). Otherwise **ftp** prompts the user for this information. In a secure environment based on Kerberos V5, **ftp** will not require a password. Instead, Kerberos authentication and authorization will be performed as described in *sis*(5). In all other environments, users are considered

authenticated if they have a password and that password is correct, and authorized if an *account* exists for them on the remote system.

verbose
Toggle verbose output. If verbose output is enabled, **ftp** displays responses from the server host, and when a file transfer completes it reports statistics regarding the efficiency of the transfer.

? [*command*]
A synonym for the **help** command. Prints the **help** information for the specified *command*.

Aborting A File Transfer

To abort a file transfer, use the terminal interrupt key (usually **Ctrl-C**). Sending transfers are halted immediately. **ftp** halts incoming (receive) transfers by first sending a FTP protocol **ABOR** command to the remote server, then discarding any further received data. The speed at which this is accomplished depends upon the remote server's support for **ABOR** processing. If the remote server does not support the **ABOR** command, an **ftp>** prompt does not appear until the remote server completes sending the requested file.

The terminal interrupt key sequence is ignored while **ftp** awaits a reply from the remote server. A long delay in this mode may result from the **ABOR** processing described above, or from unexpected behavior by the remote server, including violations of the FTP protocol. If the delay results from unexpected remote server behavior, the local **ftp** program must be killed manually.

File Naming Conventions

Files specified as arguments to **ftp** commands are processed according to the following rules.

- If the file name **-** is specified, **ftp** uses the standard input (for reading) or standard output (for writing).
- If the first character of the file name is **|**, **ftp** interprets the remainder of the argument as a shell command. **ftp** forks a shell, using **popen()** (see *popen*(3S)) with the supplied argument, and reads (writes) from standard output (standard input). If the shell command includes spaces, the argument must be quoted, as in:

  ```
  "| ls -lt"
  ```

  Some useful examples of this mechanism are:

  ```
  ls . "| more".
  ```

  The above command lists the files in the current directory page by page.

  ```
  put  "| tail -20 loc_file" rem_file.
  ```

  This command copies the last twenty lines of the local file "loc_file" to the remote system as "rem_file".
- Otherwise, if globbing is enabled, **ftp** expands local file names according to the rules used by the C shell (see *csh*(1)); see the **glob** command, below. If the **ftp** command expects a single local file (e.g. **put**), only the first filename generated by the globbing operation is used.
- For **mget** commands and **get** commands with unspecified local file names, the local filename is named the same as the remote filename, which may be altered by a **case**, **ntrans**, or **nmap** setting. The resulting filename may then be altered if **runique** is on.
- For **mput** commands and **put** commands with unspecified remote file names, the remote filename is named the same as the local filename, which may be altered by a **ntrans** or **nmap** setting. The resulting filename may then be altered by the remote server if **sunique** is on.

WARNINGS

Correct execution of many commands depends upon proper behavior by the remote server.

DIAGNOSTICS

Error! could not retrieve authentication type.

Please notify sys admin.
There are two authentication mechanisms used by **ftp**. One authentication mechanism is based on Kerberos and the other is not. The type of authentication mechanism is obtained from a system file which is updated by inetsvcs_sec(1M). If the system file does not contain known authentication types, the above error is displayed.

f

AUTHOR

ftp was developed by the University of California, Berkeley.

SEE ALSO

csh(1), rcp(1), ftpd(1M), inetsvcs_sec(1M), netrc(4), ftpusers(4), hosts(4), sis(5).

NAME

gencat - generate a formatted message catalog file

SYNOPSIS

gencat [**-l**] *catfile msgfile* ...

DESCRIPTION

Message catalogs allow a program to process input and produce output according to local customs and languages. For details, see *Native Language Support Users Guide*.

The **gencat** command merges each message source *msgfile* into a formatted message catalog *catfile* that can be accessed by **catgets()** (see *catgets*(3C)). If *catfile* does not exist, it is created. If *catfile* exists, its messages are included in the new *catfile*. If set and message numbers collide, the new message text in *file* replaces the old message text in *catfile*. A *msgfile* consists of message, directive, and comment lines (all without leading spaces or tabs) described below. Except as noted, fields are separated by one or more space or tab characters.

If **-** is specified as catalog file, standard output is used.

If **-** is specified for an instance of message file, standard input is used.

$set *s* [*comment*]
: A **$set** directive specifies the set *s*, of the messages that follow until the next **$set** or end-of-file appears. The set number *s* is an unsigned integer in the range 1 through **NL_SETMAX**. Any string following the set number is treated as a comment. If a **$set** directive is not specified, messages are put in the default set **NL_SETD**.

  Set numbers must be in ascending order within a *msgfile* but need not be contiguous.

$delset *s* [*comment*]
: A **$delset** directive deletes the message set identified by the set number *s*, from an existing message catalog. Any string following the set number is treated as a comment.

m message_text
: A message line specifies a message number *m*, and associated message text. The message number *m* is an unsigned integer in the range 1 through **NL_MSGMAX**. The *message_text* is a C string, including spaces, tabs and \ (backslash) escapes, but by default without surrounding quotes (see **$quote** directive below). The message number *m* is separated from the *message_text* by a single space or tab character. The *message_text* begins with the first character following the separator and ends at new-line. Extra spaces or tabs (including any trailing spaces or tabs) are considered part of the *message_text*.

  The *message_text* of a message line is stored in *catfile* with message number *m* and set number *s* specified by the most recent **$set** directive.

  Message numbers must be in ascending order within a set, but need not be contiguous.

  Note that the space or tab separator distinguishes insertion of a null message from deletion of a message. If a message line has a number and separator but no text, the message number and an associated null message string are stored in *catfile*. If a message line has a number but neither separator nor text, the message number and its associated message text are deleted from *catfile*.

-l
: If the **-l** option is specified, the length of *message_text* must be no more than **MAX_BUFLEN** – 1 bytes. If the **-l** option is not specified, the length of *message_text* must be no more than **NL_TEXTMAX** bytes. See *catgets*(3C), for message length limits imposed by these routines.

$quote [*q comment*]
: A **$quote** directive specifies a quote character *q*, used to surround *message_text* and make leading and trailing space visible in a message line. Any string following the specified quote character *q* is treated as a comment. By default, or if a quote character *q* not is supplied, quoting of *message_text* is not recognized.

$ *comment*	A **$** followed by a space or tab is treated as a comment and can appear anywhere in a file. A line consisting of zero or more spaces or tabs is treated as a comment line.

NL_TEXTMAX, **NL_SETMAX**, and **NL_MSGMAX** are defined in **<limits.h>**. **NL_SETD** is defined in **<nl_types.h>**. **MAX_BUFLEN** is defined in **<msgcat.h>**.

EXTERNAL INFLUENCES

Environment Variables

LANG provides a default value for the internationalization variables that are unset or null. If **LANG** is unset or null, the default value of "C" (see *lang*(5)) is used. If any of the internationalization variables contains an invalid setting, **gencat** will behave as if all internationalization variables are set to "C". See *environ*(5).

LC_ALL, if set to a non-empty string value, overrides the values of all of the other internationalization variables.

LC_CTYPE determines the interpretation of text as single and/or multi-byte characters, the classification of characters as printable, and the characters matched by character class expressions in regular expressions.

LC_MESSAGES determines the locale that should be used to affect the format and contents of diagnostic messages written to standard error and informative messages written to standard output.

NLSPATH determines the location of message catalogs for the processing of **LC_MESSAGES**.

International Code Set Support

Single- and multi-byte character code sets are supported.

WARNINGS

The **$quote** directive is not currently supported on the HP MPE and RTE operating systems.

AUTHOR

gencat was developed by HP and the X/Open Company, Ltd.

SEE ALSO

dumpmsg(1), findmsg(1), insertmsg(1), catgets(3C), catopen(3C).

Native Language Support Users Guide.

STANDARDS CONFORMANCE

gencat: XPG2, XPG3

NAME

genxlt - generate iconv translation tables

SYNOPSIS

genxlt [**-f** *output_filename*] [*input_filename*]

DESCRIPTION

genxlt generates a compiled, non-readable binary version of the iconv table that is suitable for use by *iconv*(1) and *iconv*(3C). If *input_filename* or *output_filename* is not supplied, standard input and/or standard output will be used.

Since the output of **genxlt** is a binary, non-readable file, if the **-f** option is not used, the redirection symbol **>** maybe used to redirect the standard output to a file.

Options

genxlt recognizes the following options:

-f *output_filename*
: If this option is not selected, the data will be sent to standard output, from where it could be redirected to a file.

genxlt creates tables that are in a prescribed format and which can be interpreted by the default conversion routines of *iconv*(3C). The input file has two columns, giving the filecode mapping between the two code sets. The entries are in hexadecimal.

The input file must be formatted as two columns of hexadecimal digits. Characters in the first column are translated into the characters in the second column. Lines preceded with **#** in the first column are ignored as comments on all lines except in the case of the following keywords: *#Galley:* and *#What:*

In addition to the data, which defines the filecode mapping, a *Galley* character (see *iconv*(3C)) may also be defined for that particular conversion. This is done by adding the line **#Galley: 0xnnnn**, to the beginning of the input file. The **nnnn** is any multi-byte character (see *EXAMPLES*). A *What* string (see *what*(1)), may also be defined in the input file using the entry **#What:** <*any_string*>. This string may contain information like version number, type of conversion, etc., which are not used in any way for the conversions. Note that if the *What* string is defined, it should appear before the *Galley* definition.

EXTERNAL INFLUENCES

Environment Variables

LANG provides a default value for the internationalization variables that are unset or null. If **LANG** is unset or null, the default value of "C" (see *lang*(5)) is used. If any of the internationalization variables contains an invalid setting, **genxlt** will behave as if all internationalization variables are set to "C" (see *environ*(5)).

If **LC_ALL** is set to a non-empty string value, it overrides the values of all the other internationalization variables.

LC_MESSAGES determines the locale that should be used to affect the format and contents of diagnostic messages written to standard error and informative messages written to standard output.

NLSPATH determines the location of message catalogues for the processing of **LC_MESSAGES**.

International Code Set Support

Single and multi-byte character code sets are supported.

RETURN VALUE

The exit values are:

0	Successful completion.
>0	Error condition occurred.

EXAMPLES

This example compiles the *iconv_input* and puts the output binary in **/usr/lib/nls/iconv/tables/roma8=iso81.** The following iconv statement uses the *roma8=iso81* table to convert the *data_file* from code set **roman8** to code set **iso8859-1**.

```
% genxlt iconv_input > /usr/lib/nls/iconv/tables/roma8=iso81
% iconv -f roma8 -t iso81 data_file
```

This is an example of the input_file:

```
#What: CodesetA to CodesetB: version 1.0
#Galley:   0xffff
# the conversion data is as follows:
0x01     0x01
0x02     0x42
...
0xff87     0x4567
...
etc.
```

WARNINGS

Because **genxlt** will write over the existing table, it is wise to save the existing table into another file before using **genxlt**.

Warnings are not given for incorrect data in the input_file.

You must have super-user privileges to install files in **/usr/lib/nls/iconv/tables**.

FILES

/usr/lib/nls/iconv/tables All tables must be installed in this directory.

SEE ALSO

dmpxlt(1), iconv(1), iconv(3C).

STANDARDS COMPLIANCE

genxlt: XPG4 tables

NAME

get - get a version of an SCCS file

SYNOPSIS

get [**-r** *SID*] [**-c** *cutoff*] [**-e**] [**-b**] [**-i** *list*] [**-x** *list*] [**-k**] [**-l**[**p**]] [**-p**] [**-s**] [**-m**] [**-n**] [**-g**] [**-t**] [**-w** *string*] [**-a** *seq-number*] *file* ...

DESCRIPTION

The **get** command generates an ASCII text file from each named SCCS file according to the specifications given by its option arguments, which begin with **-**. The arguments can be specified in any order, but all option arguments apply to all named SCCS files. If a directory is named, **get** behaves as if each file in the directory was specified as a named file, except that non-SCCS files (last component of the path name does not begin with **s.**) and unreadable files are silently ignored. If a file name of **-** is given, the standard input is read and each line of the standard input is assumed to be the name of an SCCS file to be processed. Again, non-SCCS files and unreadable files are silently ignored.

The generated text is normally written into a file called the *g-file* whose name is derived from the SCCS file name by simply removing the **s.** prefix (see FILES below).

Options

Explanation of the option arguments below is based on processing only one SCCS file. When processing multiple SCCS files, the effects of any option argument applies independently to each named file.

-r*SID* The *S*CCS *ID*entification string (SID) of the version (delta) of an SCCS file to be retrieved. Table 1 shows, for the most useful cases, which version of an SCCS file is retrieved (as well as the SID of the version to be eventually created by **delta** if the **-e** option is also used), as a function of the SID specified (see *delta*(1)).

-c*cutoff* *cutoff* date-time, in the form:

YY [*MM* [*DD* [*HH* [*MM* [*SS*]]]]]

No changes (deltas) to the SCCS file which were created after the specified *cutoff* date-time are included in the generated ASCII text file. Units omitted from the date-time default to their maximum possible values; that is, **-c7502** is equivalent to **-c750228235959**. Any number of non-numeric characters can separate the various 2-digit pieces of the *cutoff* date-time. This feature allows one to specify a *cutoff* date in the form: **-c77/2/2 9:22:25**. Note that this implies that one can use the **%E%** and **%U%** identification keywords (see below) for nested **get**s within, for example, a **send** command (see *send*(1)):

```
~!get "-c%E% %U%" s.file
```

-e Indicates that the **get** is for the purpose of editing or making a change (delta) to the SCCS file via a subsequent use of **delta**. The **-e** option used in a **get** for a particular version (SID) of the SCCS file prevents further **get**s for editing on the same SID until **delta** is executed or the **j** (joint edit) flag is set in the SCCS file (see *admin*(1)). Concurrent use of **get -e** for different SIDs is always allowed. Note, however, that only one user is permitted to do a concurrent **get -e** (see *admin*(1)).

If the *g-file* generated by **get** with an **-e** option is accidentally ruined in the process of editing it, it can be regenerated by re-executing the **get** command with the **-k** option in place of the **-e** option.

SCCS file protection specified via the ceiling, floor, and authorized user list stored in the SCCS file (see *admin*(1)) are enforced when the **-e** option is used.

-b Used with the **-e** option to indicate that the new delta should have an SID in a new branch as shown in Table 1. This option is ignored if the **b** flag is not present in the file (see *admin*(1)) or if the retrieved *delta* is not a leaf *delta*. (A leaf *delta* is one that has no successors on the SCCS file tree.)

Note: A branch *delta* can always be created from a non-leaf *delta*.

-i*list* A *list* of deltas to be included (forced to be applied) in the creation of the generated file. The *list* has the following syntax:

list ::= *range* | *list*, *range*
range ::= *SID* | *SID* - *SID*

SID, the SCCS Identification of a delta, can be in any form shown in the "SID Specified" column of Table 1. Partial SIDs are interpreted as shown in the "SID Retrieved" column of Table 1. See WARNINGS.

-x*list* A *list* of deltas to be excluded (forced not to be applied) in the creation of the generated file. See the **-i** option for the *list* format.

-k Suppresses replacement of identification keywords (see below) in the retrieved text by their value. The **-k** option is implied by the **-e** option.

-l[**p**] Causes a delta summary to be written into an *l-file*. If **-lp** is used, an *l-file* is not created; the delta summary is written on the standard output instead. See FILES for the format of the *l-file*. The user must have *s-file* read permission in order to use the **-l** option.

-p Causes the text retrieved from the SCCS file to be written on the standard output. No *g-file* is created. All output that normally goes to the standard output goes to file descriptor 2 (standard error) instead, unless the **-s** option is used, in which case it disappears.

-s Suppresses all output normally written on the standard output. However, fatal error messages (which always go to file descriptor 2) remain unaffected.

-m Causes each text line retrieved from the SCCS file to be preceded by the SID of the delta that inserted the text line in the SCCS file. The format is: SID, followed by a horizontal tab, followed by the text line.

-n Causes each generated text line to be preceded with the **%M%** identification keyword value (see below). The format is: **%M%** value, followed by a horizontal tab, followed by the text line. When both the **-m** and **-n** options are used, the format is: **%M%** value, followed by a horizontal tab, followed by the **-m** option-generated format.

-g Suppresses the actual retrieval of text from the SCCS file. It is primarily used to generate an *l-file*, or to verify the existence of a particular SID.

-t Used to access the most recently created ("top") delta in a given release (e.g., **-r1**), or release and level (e.g., **-r1.2**).

-w *string* Substitute *string* for all occurrences of **@%M%** when **get**ting the file.

-a*seq-number* The delta sequence number of the SCCS file delta (version) to be retrieved (see *sccsfile*(4)). This option is used by the **comb** command (see *comb*(1)); it is not a generally useful option, and should be avoided. If both the **-r** and **-a** options are specified, the **-a** option is used. Care should be taken when using the **-a** option in conjunction with the **-e** option, because the SID of the delta to be created may not be what one expects. The **-r** option can be used with the **-a** and **-e** options to control the naming of the SID of the delta to be created.

For each file processed, **get** responds (on the standard output) with the SID being accessed and with the number of lines retrieved from the SCCS file.

If the **-e** option is used, the SID of the delta to be made appears after the SID accessed and before the number of lines generated. If there is more than one named file, or if a directory or standard input is named, each file name is printed (preceded by a new-line) before it is processed. If the **-i** option is used included deltas are listed following the notation "Included". If the **-x** option is used, excluded deltas are listed following the notation "Excluded".

TABLE 1. Determination of SCCS Identification String

SID* Specified	−**b** Option Used %	Other Conditions	SID Retrieved	SID of Delta to be Created
none %%	no	R defaults to mR	mR.mL	mR.(mL+1)
none %%	yes	R defaults to mR	mR.mL	mR.mL.(mB+1).1
R	no	R > mR	mR.mL	R.1***
R	no	R = mR	mR.mL	mR.(mL+1)
R	yes	R > mR	mR.mL	mR.mL.(mB+1).1
R	yes	R = mR	mR.mL	mR.mL.(mB+1).1
R	-	R < mR and R does *not* exist	hR.mL**	hR.mL.(mB+1).1
R	-	Trunk succ.# in release > R and R exists	R.mL	R.mL.(mB+1).1
R.L	no	No trunk succ.	R.L	R.(L+1)
R.L	yes	No trunk succ.	R.L	R.L.(mB+1).1
R.L	-	Trunk succ. in release ≥ R	R.L	R.L.(mB+1).1
R.L.B	no	No branch succ.	R.L.B.mS	R.L.B.(mS+1)
R.L.B	yes	No branch succ.	R.L.B.mS	R.L.(mB+1).1
R.L.B.S	no	No branch succ.	R.L.B.S	R.L.B.(S+1)
R.L.B.S	yes	No branch succ.	R.L.B.S	R.L.(mB+1).1
R.L.B.S	-	Branch succ.	R.L.B.S	R.L.(mB+1).1

*	"R", "L", "B", and "S" are the "release", "level", "branch", and "sequence" components of the SID, respectively; "m" means "maximum". Thus, for example, "R.mL" means "the maximum level number within release R"; "R.L.(mB+1).1" means "the first sequence number on the *new* branch (i.e., maximum branch number plus one) of level L within release R". Note that if the SID specified is of the form "R.L", "R.L.B", or "R.L.B.S", each of the specified components *must* exist.
**	"hR" is the highest *existing* release that is lower than the specified, *nonexistent*, release R.
***	This is used to force creation of the *first* delta in a *new* release.
#	Successor.
%	The **-b** option is effective only if the **b** flag (see *admin*(1)) is present in the file. An entry of **-** means "irrelevant".
%%	This case applies if the **d** (default SID) flag is *not* present in the file. If the **d** flag *is* present in the file, then the SID obtained from the **d** flag is interpreted as if it had been specified on the command line. Thus, one of the other cases in this table applies.

Identification Keywords

Identifying information is inserted into the text retrieved from the SCCS file by replacing *identification keywords* with their value wherever they occur. The following keywords can be used in the text stored in an SCCS file:

Keyword	**Value**
%M%	Module name: either the value of the **m** flag in the file (see *admin*(1)), or if absent, the name of the SCCS file with the leading **s.** removed.
%I%	SCCS identification (SID) (**%R%.%L%.%B%.%S%**) of the retrieved text.
%R%	Release.
%L%	Level.
%B%	Branch.
%S%	Sequence.
%D%	Current date (YY/MM/DD).
%H%	Current date (MM/DD/YY).
%T%	Current time (HH:MM:SS).
%E%	Date newest applied delta was created (YY/MM/DD).
%G%	Date newest applied delta was created (MM/DD/YY).
%U%	Time newest applied delta was created (HH:MM:SS).
%Y%	Module type: value of the **t** flag in the SCCS file (see *admin*(1)).
%F%	SCCS file name.

%P%	Fully qualified SCCS file name.
%Q%	The value of the **q** flag in the file (see *admin*(1)).
%C%	Current line number. This keyword is intended for identifying messages output by the program such as "this should not have happened" type errors. It is *not* intended to be used on every line to provide sequence numbers.
%Z%	The 4-character string recognizable by **what** (see *what*(1)).
%W%	A shorthand notation for constructing *what*(1) strings for HP-UX system program files. **%W%**=**%Z%%M%***horizontal-tab***%I%**
%A%	Another shorthand notation for constructing *what*(1) strings for non-HP-UX system program files. **%A% = %Z%%Y% %M% %I%%Z%**

EXTERNAL INFLUENCES

Environment Variables

LC_CTYPE determines the interpretation of text as single- and/or multi-byte characters.

LC_MESSAGES determines the language in which messages are displayed.

If **LC_CTYPE** or **LC_MESSAGES** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**. If any internationalization variable contains an invalid setting, **get** behaves as if all internationalization variables are set to "C". See *environ*(5).

g

International Code Set Support

Single- and multi-byte character code sets are supported.

DIAGNOSTICS

Use *sccshelp*(1) for explanations.

WARNINGS

If the effective user has write permission (either explicitly or implicitly) in the directory containing the SCCS files, but the real user does not, then only one file can be named when the **-e** option is used.

Unexpected results occur when using the **-i** option to merge changes into sections of a file that have been (perhaps inadvertently) deleted and subsequently re-inserted into a file.

An *l-file* cannot be generated when **-g** is used. In other words, **-g -l** does not work.

FILES

Several auxiliary files can be created by **get**. These files are known generically as the *g-file*, *l-file*, *p-file*, and *z-file*. The letter before the hyphen is called the tag. An auxiliary file name is formed from the SCCS file name: the last component of all SCCS file names must be of the form **s.***module-name*, the auxiliary files are named by replacing the leading **s** with the tag. The *g-file* is an exception to this scheme: the *g-file* is named by removing the **s.** prefix. For example, **s.xyz.c**, the auxiliary file names would be **xyz.c**, **l.xyz.c**, **p.xyz.c**, and **z.xyz.c**, respectively.

The *g-file*, which contains the generated text, is created in the current directory (unless the **-p** option is used). A *g-file* is created in all cases, whether or not any lines of text were generated by the **get**. It is owned by the real user. If the **-k** option is used or implied its mode is 644; otherwise its mode is 444. Only the real user need have write permission in the current directory.

The *l-file* contains a table showing which deltas were applied in generating the retrieved text. The *l-file* is created in the current directory if the **-l** option is used; its mode is 444 and it is owned by the real user. Only the real user need have write permission in the current directory.

Lines in the *l-file* have the following format:

1. A blank character if the delta was applied;
   ***** otherwise.
2. A blank character if the delta was applied or was not applied and ignored;
   ***** if the delta was not applied and was not ignored.
3. A code indicating a "special" reason why the delta was or was not applied:

   I: Included.

X: Excluded.
C: Cut off (by a **-c** option).

4. Blank.
5 SCCS identification (SID).
6. Tab character.
7. Creation date and time (in the form YY/MM/DD HH:MM:SS).
8. Blank.
9. Login name of person who created *delta*.

The comments and MR data follow on subsequent lines, indented one horizontal tab character. A blank line terminates each entry.

The *p-file* is used to pass information resulting from a **get** with an **-e** option along to *delta*. Its contents are also used to prevent a subsequent execution of **get** with an **-e** option for the same SID until *delta* is executed or the joint edit flag, **j**, (see *admin*(1)) is set in the SCCS file. The *p-file* is created in the directory containing the SCCS file and the effective user must have write permission in that directory. Its mode is 644 and it is owned by the effective user. The format of the *p-file* is: the gotten SID, followed by a blank, followed by the SID that the new delta will have when it is made, followed by a blank, followed by the login name of the real user, followed by a blank, followed by the date-time the **get** was executed, followed by a blank and the **-i** option argument if it was present, followed by a blank and the **-x** option argument if it was present, followed by a new-line. There can be an arbitrary number of lines in the *p-file* at any time; no two lines can have the same new delta SID.

The *z-file* serves as a *lock-out* mechanism against simultaneous updates. Its contents are the binary (2 bytes) process ID of the command (i.e., **get**) that created it. The *z-file* is created in the directory containing the SCCS file for the duration of **get**. The same protection restrictions as those for the *p-file* apply for the *z-file*. The *z-file* is created mode 444.

SEE ALSO

admin(1), delta(1), sccshelp(1), prs(1), what(1), sccsfile(4).

STANDARDS CONFORMANCE

get: SVID2, SVID3, XPG2, XPG3, XPG4

NAME

getaccess - list access rights to file(s)

SYNOPSIS

getaccess [**-u** *user*] [**-g** *user*] *group*[**,***group*]...] [**-n**] *file* ...

getaccess -r [**-n**] *file* ...

DESCRIPTION

getaccess lists for the specified files the effective access rights of the caller (that is, for their effective user ID, effective group ID, and supplementary groups list). By default, the command prints a symbolic representation of the user's access rights to the named file: **r** or **-** for read/no read, **w** or **-** for write/no write, and **x** or **-** for execute/no execute (for directories, search/no search), followed by the file name.

Options

getaccess recognizes the following options and command-line arguments:

-u *user* List access for the given user instead of the caller. A *user* can be a known user name, a valid ID number, or @, representing the file's owner ID. If information about more than one file is requested, the value of @ can differ for each.

This option sets the user ID only. The access check is made with the caller's effective group ID and supplementary group IDs unless **-g** is also specified.

-g *group*[, *group*]...]
List access for the given group(s) instead of the caller's effective group ID and supplementary groups list. A *group* can be a known group name, a valid ID number, or @, representing the file's group ID. If information about more than one file is requested, the value of @ can differ for each.

-r List access using the caller's real user ID, group ID, and supplementary groups list, instead of effective ID values.

-n List access rights numerically (octal digits **0**..**7** instead of **rwx**) for each file requested. The bit values **R_OK**, **W_OK**, and **X_OK** are defined in the file <**unistd.h**>.

Checking access using access control lists is described in *acl*(5).

In addition, the write bit is cleared for files on read-only file systems or shared-text programs being executed. The execute bit is not turned off for shared-text programs open for writing because it is not possible to ascertain whether a file open for writing is a shared-text program.

Processes with appropriate privileges have read and write access to all files. However, write access is denied for files on read-only file systems or shared-text programs being executed. Execute access is allowed if and only if the file is not a regular file or the execute bit is set in any of the file's ACL entries.

To use **getaccess** successfully, the caller must have search access in every directory component of the path name of the *file*. **getaccess** verifies search access first by using the caller's effective IDs, regardless of the user and group IDs specified. This is distinct from the case in which the caller can search the path but the user for whom access is being checked does not have access to the file.

Note: a file name argument of **-** has no special meaning (such as standard input) to **getaccess**.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages are displayed.

If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**. If any internationalization variable contains an invalid setting, **getaccess** behaves as if all internationalization variables are set to "C". See *environ*(5).

RETURN VALUE

getaccess returns one of the following values:

0 Successful completion.

1 **getaccess** was invoked incorrectly or encountered an unknown user or group name. An appropriate message is printed to standard error.

2 A file is nonexistent or unreachable (by the caller). **getaccess** prints an appropriate message to standard error, continues, then returns a value of 2 upon completion.

EXAMPLES

The following command prints the caller's access rights to *file1* using the file's group ID instead of the caller's effective group ID and groups list.

```
getaccess -g@ file1
```

Here's how to check access by user **ggd** in groups **red** and **19** to all files in the current directory, with access rights expressed as octal values.

```
getaccess -u ggd -g red,19 -n .* *
```

Here's how to list access rights for all files under **mydir**.

```
find mydir -print | sort | xargs getaccess
```

AUTHOR

getaccess was developed by HP.

FILES

/etc/passwd
/etc/group

SEE ALSO

chacl(1), lsacl(1), getaccess(2), glossary(9).

NAME

getconf - get system configuration values

SYNOPSIS

getconf [**-v** *specification*] [*system_var*]

getconf [**-v** *specification*] [*path_var*] [*pathname*]

DESCRIPTION

The **getconf** command provides an interface to the *confstr*(3C), *pathconf*(2), and *sysconf*(2) library routines and system calls.

Use the first synopsis form, for inquiries involving **confstr()**, or **sysconf()** (in the first table below). Use the second synopsis form, for inquiries involving **pathconf()** (in the second table below).

Options

getconf recognizes the following option:

-v *specification* Return configuration variables corresponding to a particular compilation environment supported by HP-UX. If the **-v** option is not specified, *specification* defaults to XBS5_ILP32_OFF32. See table below for possible specifications and meanings.

Specification	int	long	pointer	off_t
XBS5_ILP32_OFF32	32	32	32	32
XBS5_ILP32_OFFBIG	32	32	32	>=64
XBS5_LP64_OFF64	32	64	64	64
XBS5_LPBIG_OFFBIG	>=32	>=64	>=64	>=64

g

EXTERNAL INFLUENCES

Environment Variables

LC_MESSAGES determines the language in which messages are displayed.

If **LC_MESSAGES** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**.

If any internationalization variable contains an invalid setting, **getconf** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single-byte and multi-byte character code sets are supported.

RETURN VALUE

The error codes returned by **getconf** are:

0 Success. A value corresponding to the operand was returned.
1 One or more missing or extra operands.
2 Operand was not recognized.
3 *pathname* could not be accessed.

EXAMPLES

Request the number of intervals per second:

```
getconf CLK_TCK
```

Request the maximum value of a file's link count:

```
getconf LINK_MAX /etc/passwd
```

Other supported inquiries include the following:

```
ARG_MAX             _POSIX_CHILD_MAX        POSIX2_C_DEV
BC_BASE_MAX         _POSIX_JOB_CONTROL      POSIX2_EXPR_NEST_MAX
BC_DIM_MAX          _POSIX_NGROUPS_MAX      POSIX2_FORT_DEV
BC_SCALE_MAX        _POSIX_OPEN_MAX         POSIX2_FORT_RUN
BC_STRING_MAX       _POSIX_SAVED_IDS        POSIX2_LINE_MAX
CHILD_MAX           _POSIX_SSIZE_MAX        POSIX2_LOCALEDEF
CLK_TCK             _POSIX_STREAM_MAX       POSIX2_RE_DUP_MAX
COLL_WEIGHTS_MAX    _POSIX_TZNAME_MAX       POSIX2_SW_DEV
```

CS_PATH	**_POSIX_VERSION**	**POSIX2_VERSION**
EXPR_NEST_MAX	**POSIX2_BC_BASE_MAX**	**RE_DUP_MAX**
LINE_MAX	**POSIX2_BC_DIM_MAX**	**SC_PASS_MAX**
NGROUPS_MAX	**POSIX2_BC_SCALE_MAX**	**SC_XOPEN_VERSION**
OPEN_MAX	**POSIX2_BC_STRING_MAX**	**STREAM_MAX**
PATH	**POSIX2_COLL_WEIGHTS_MAX**	**TZNAME_MAX**
_POSIX_ARG_MAX	**POSIX2_C_BIND**	
CHARCLASS_NAME_MAX	**CHAR_BIT**	**CHAR_MAX**
CHAR_MIN	**NZERO**	**POSIX2_CHAR_TERM**
POSIX2_C_VERSION	**POSIX_OPEN_MAX**	**POSIX_PATH_MAX**
POSIX_PIPE_BUF	**POSIX_SAVED_IDS**	**POSIX_SSIZE_MAX**
POSIX_STREAM_MAX	**POSIX_TZNAME_MAX**	**POSIX_VERSION**
SCHAR_MAX	**SCHAR_MIN**	**INT_MAX**
INT_MIN	**LONG_BIT**	**LONG_MAX**
LONG_MIN	**MB_LEN_MAX**	**NL_NMAX**
NL_ARGMAX	**NL_LANGMAX**	**NL_MSGMAX**
NL_SETMAX	**NL_TEXTMAX**	**POSIX2_UPE**
POSIX_ARG_MAX	**POSIX_CHILD_MAX**	**POSIX_JOB_CONTROL**
POSIX_LINK_MAX	**POSIX_MAX_CANON**	**POSIX_MAX_INPUT**
POSIX_NAME_MAX	**POSIX_NGROUPS_MAX**	**SHRT_MAX**
SHRT_MIN	**SSIZE_MAX**	**TMP_MAX**
UCHAR_MAX	**UINT_MAX**	**ULONG_MAX**
USHRT_MAX	**WORD_BIT**	**XOPEN_VERSION**
XOPEN_XCU_VERSION	**XOPEN_XPG2**	**XOPEN_XPG3**
XOPEN_XPG4	**CPU_CHIP_TYPE**	**KERNEL_BITS**
HW_CPU_SUPP_BITS	**MACHINE_MODEL**	**HW_32_64_CAPABLE**

XBS5_ILP32_OFF32_CFLAGS	**XBS5_ILP32_OFF32_LDFLAGS**
XBS5_ILP32_OFF32_LIBS	**XBS5_ILP32_OFF32_LINTFLAGS**
XBS5_ILP32_OFFBIG_CFLAGS	**XBS5_ILP32_OFFBIG_LDFLAGS**
XBS5_ILP32_OFFBIG_LIBS	**XBS5_ILP32_OFFBIG_LINTFLAGS**
XBS5_LP64_OFF64_CFLAGS	**XBS5_LP64_OFF64_LDFLAGS**
XBS5_LP64_OFF64_LIBS	**XBS5_LP64_OFF64_LINTFLAGS**

Supported inquiries requiring the second parameter include:

LINK_MAX	**PIPE_BUF**	**_POSIX_NAME_MAX**
MAX_CANON	**_POSIX_CHOWN_RESTRICTED**	**_POSIX_NO_TRUNC**
MAX_INPUT	**_POSIX_LINK_MAX**	**_POSIX_PATH_MAX**
NAME_MAX	**_POSIX_MAX_CANON**	**_POSIX_PIPE_BUF**
PATH_MAX	**_POSIX_MAX_INPUT**	**_POSIX_VDISABLE**
POSIX_NO_TRUNC	**POSIX_CHOWN_RESTRICTED**	**POSIX_VDISABLE**

AUTHOR

getconf was developed by HP and POSIX.

SEE ALSO

pathconf(2), sysconf(2), confstr(3C).

STANDARDS CONFORMANCE

getconf: POSIX.2, XPG4

NAME

getopt - parse command options

SYNOPSIS

getopt *optstring args*

DESCRIPTION

getopt is used to break up options in command lines for easy parsing by shell procedures and to check for legal options. *optstring* is a string of recognized option letters (see *getopt*(3C)). If a letter is followed by a colon, the option is expected to have an argument which may or may not be separated from it by white space.

The positional parameters ($1 $2 ...) of the shell are reset so that each option is preceded by a **-** and is in its own positional parameter; each option argument is also parsed into its own positional parameter.

getopt recognizes two hyphens (**--**) to delimit the end of the options. If absent, **getopt** places **--** at the end of the options.

The most common use of **getopt** is in the shell's **set** command (see the example below) where **getopt** converts the command line to a more easily parsed form. **getopt** writes the modified command line to the standard output.

EXTERNAL INFLUENCES

Environment Variables

LC_MESSAGES determines the language in which messages are displayed.

If **LC_MESSAGES** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable.

If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**. If any internationalization variable contains an invalid setting, **getopt** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single-byte and multi-byte character code sets are supported.

DIAGNOSTICS

getopt prints an error message on the standard error when it encounters an option letter not included in *optstring*.

EXAMPLES

The following code fragment processes the arguments for a command that can take the options **a** or **b**, and the option **o** which requires an argument:

```
set -- ‘getopt abo: $*‘

if [ $? -ne 0 ]; then
    echo $USAGE
    exit 2
fi

while [ $# -gt 0 ]; do
    case $1 in
    -a | -b)
        FLAG=$1
        shift
        ;;
    -o)
        OARG=$2
        shift 2
        ;;
    --)
        shift
        break
        ;;
    esac
```

```
        done
```

This code accepts any of the following as equivalent:

```
        cmd -aoarg file file
        cmd -a -o arg file file
        cmd -oarg -a file file
        cmd -a -oarg -- file file
```

WARNINGS

getopt option arguments must not be null strings nor contain embedded blanks.

SEE ALSO

sh(1), getopt(3C).

NAME

getopts - parse utility (command) options

SYNOPSIS

getopts *optstring name* [*arg* ...]

DESCRIPTION

getopts is used to retrieve options and option-arguments from a list of parameters.

Each time it is invoked, **getopts** places the value of the next option in the shell variable specified by the **name** operand and the index of the next argument to be processed in the shell variable **OPTIND**. Whenever the shell is invoked, **OPTIND** is initialized to 1.

When the option requires an option-argument, **getopts** places it in the shell variable **OPTARG**. If no option was found, or if the option that was found does not have an option-argument, **OPTARG** is unset.

If an option character not contained in the *optstring* operand is found where an option character is expected, the shell variable specified by *name* is set to the question-mark (**?**) character. In this case, if the first character in *optstring* is a colon (**:**), the shell variable **OPTARG** is set to the option character found, but no output is written to standard error; otherwise, the shell variable **OPTARG** is unset and a diagnostic message is written to standard error. This condition is considered to be an error detected in the way arguments were presented to the invoking application, but is not an error in **getopts** processing.

If an option-argument is missing:

- If the first character of *optstring* is a colon, the shell variable specified by *name* is set to the colon character and the shell variable **OPTARG** is set to the option character found.
- Otherwise, the shell variable specified by *name* is set to the question-mark character, the shell variable **OPTARG** is unset, and a diagnostic message is written to the standard error. This condition is considered to be an error detected in the way arguments are presented to the invoking application, but is not an error in **getopts** processing; a diagnostic message is written as stated, but the exit status is zero.

When the end of options is encountered, **getopts** exits with a return value greater than zero. The shell variable **OPTIND** is set to the index of the first nonoption-argument, where the first **--** argument is considered to be an option argument if there are no other non-option arguments appearing before it, or the value **$#** + 1 if there are no nonoption-arguments; the *name* variable is set to the question-mark character. Any of the following identifies the end of options: the special option **--**, finding an argument that does not begin with a **-**, or encountering an error.

The shell variables **OPTIND** and **OPTARG** are local to the caller of **getopts** and are not exported by default.

The shell variable specified by the *name* operand, **OPTIND**, and **OPTARG** affect the current shell execution environment.

Operands

The following operands are supported:

optstring A string containing the option characters recognized by the utility invoking **getopts**. If a character is followed by a colon (**:**), the option will be expected to have an argument, which should be supplied as a separate argument. Applications should specify an option character and its option-argument as separate arguments, but **getopts** will interpret the characters following an option character requiring arguments as an argument whether or not this is done. An explicit null option-argument need not be recognised if it is not supplied as a separate argument when **getopts** is invoked. The characters question-mark (**?**) and colon (**:**) must not be used as option characters by an application. The use of other option characters that are not alphanumeric produces unspecified results. If the option-argument is not supplied as a separate argument from the option character, the value in **OPTARG** will be stripped of the option character and the **-**. The first character in *optstring* will determine how **getopts** will behave if an option character is not known or an option-argument is missing.

name The name of a shell variable that is set by **getopts** to the option character that was found.

getopts by default parses positional parameters passed to the invoking shell procedures. If *args* are given, they are parsed instead of the positional parameters.

EXTERNAL INFLUENCES

Environment Variable

The following environment variable affects the execution of the **getopts** utility:

OPTIND	Used by **getopts** as the index of the next argument to be processed.

ERRORS

Whenever an error is detected and the first character in the *optstring* operand is not a colon (**:**), a diagnostic message will be written to standard error with the following information in an unspecified format:

- The invoking program name will be identified in the message. The invoking program name will be the value of the shell special parameter 0 at the time the **getopts** utility is invoked. A name equivalent to:

  ```
  basename "$0"
  ```

  may be used.
- If an option is found that was not specified in *optstring*, this error will be identified and the invalid option character will be identified in the message.
- If an option requiring an option-argument is found, but an option-argument is not found, this error will be identified and the invalid option character will be identified in the message.

EXAMPLES

Since **getopts** affects the current shell execution environment, it is generally provided as a shell regular built-in. If it is called in a subshell or separate utility execution environment such as one of the following:

```
(getopts abc value "$@")
nohup getopts ...
find -exec getopts ...\;
```

it does not affect the shell variables in the caller's environment.

Note that shell functions share **OPTIND** with the calling shell even though the positional parameters are changed. Functions that use **getopts** to parse their arguments should save the value of **OPTIND** on entry and restore it before returning. However, there will be cases when a function must change **OPTIND** for the calling shell.

The following example script parses and displays its arguments:

```
aflag=
bflag=

while getopts ab: name
    do
        case $name in
        a)
                aflag=1;;
        b)
                bflag=1
                bval="$OPTARG";;
        ?)
                printf "Usage: %s: [-a] [-b value] args\n" $0
                exit 2;;
        esac
    done
if [ ! -z "$aflag" ] ; then
        printf "Option -a specified\n"
fi
if [ ! -z "$bflag" ] ; then
        printf "Option -b "%s" specified\n" "$bval"
fi
shift $(($OPTIND -1))
printf "Remaining arguments are: %s\n" "$*"
```

g

SEE ALSO

getopt(1), ksh(1), sh-posix(1), sh(1), getopt(3C).

STANDARDS CONFORMANCE

`getopts`: XPG4, POSIX.2

NAME

getprivgrp - get special attributes for group

SYNOPSIS

getprivgrp [**-g**| *group_name*]

DESCRIPTION

getprivgrp lists the access privileges of privileged groups set by **setprivgrp** (see *setprivgrp*(1M)). If *group_name* is supplied, access privileges are listed for that group only. If the caller is not a member of *group_name*, no information is displayed. If **-g** is used, **getprivgrp** lists access privileges that have been granted to all groups. Otherwise, access privileges are listed for all privileged groups to which the caller belongs.

The super-user is a member of all groups. Access privileges include **RTPRIO**, **RTSCHED**, **MLOCK**, **CHOWN**, **LOCKRDONLY**, **SETRUGID**, and **SERIALIZE**.

AUTHOR

getprivgrp was developed by HP.

SEE ALSO

setprivgrp(1M), getprivgrp(2), privgrp(4).

NAME

gprof - display call graph profile data

SYNOPSIS

gprof [*options*] [*a.out* [*gmon.out* ...]]

DESCRIPTION

The **gprof** command produces an execution profile of C, Pascal, and FORTRAN programs. The effect of called routines is incorporated into the profile of each caller. Profile data is taken from the call graph profile file (**gmon.out** default) that is created by programs compiled with the **-G** option of **cc**, **pc**, and **f77** (see *cc*(1), *pc*(1), and *f77*(1)). That option also links in versions of the library routines that are compiled for profiling. The symbol table in the named object file (**a.out** default) is read and correlated with the call graph profile file. If more than one profile file is specified, **gprof** output shows the sum of the profile information in the given profile files.

First, a flat profile is given, similar to that provided by **prof** (see *prof*(1)). This listing gives the total execution times and call counts for each function in the program, sorted by decreasing time.

Next, these times are propagated along the edges of the call graph. **gprof** discovers all cycles in the call graph. All calls made into the cycle share the time of that cycle. A second listing shows the functions sorted according to the time they represent including the time of their call graph descendants. Below each function entry is shown its (direct) call graph children, and how their times are propagated to this function. A similar display above the function shows how the time of this function and the time of its descendants are propagated to its (direct) call graph parents.

Cycles are also shown, with an entry for the cycle as a whole and a listing of the members of the cycle, each with their contributions to the time and call counts of the cycle.

Options

The **gprof** command recognizes the following options:

-a Suppress printing statically declared functions. If this option is given, all relevant information about the static function (such as time samples, calls to other functions, and calls from other functions) belongs to the function loaded just before the static function in the **a.out** file.

-b Suppress printing a description of each field in the profile.

-e *name* Suppress printing the graph profile entry for routine *name* and all its descendants (unless they have other ancestors that are not suppressed). More than one **-e** option can be given. Only one *name* can be given with each **-e** option.

-E *name* Suppress printing the graph profile entry for routine *name* (and its descendants) as **-e** above, and also exclude the time spent in *name* (and its descendants) from the total and percentage time computations. (For example, **-E mcount -E mcleanup** is the default.)

-f *name* Print only the graph profile entry of the specified routine *name* and its descendants. More than one **-f** option can be given. Only one *name* can be given with each **-f** option.

-F *name* Print only the graph profile entry of the routine *name* and its descendants (as **-f** above) and also uses only the times of the printed routines in total time and percentage computations. More than one **-F** option can be given. Only one *name* can be given with each **-F** option. The **-F** option overrides the **-E** option.

-s Produce a profile file **gmon.sum** that represents the sum of the profile information in all specified profile files. This summary profile file can be given to subsequent executions of **gprof** (probably also with a **-s** option) to accumulate profile data across several runs of an **a.out** file.

-z Display routines that have zero usage (as indicated by call counts and accumulated time).

The name of the file created by a profiled program is controlled by the environment variable **GPROFDIR**. If **GPROFDIR** is not set, **gmon.out** is produced in the current directory when the program terminates. If **GPROFDIR=string**, **string/***pid.progname* is produced, where *progname* consists of *argv[0]* with any path prefix removed, and *pid* is the program's process ID. If **GPROFDIR** is set to a null string, no profiling output is produced.

WARNINGS

Beware of quantization errors. The granularity of the sampling is shown, but remains statistical at best. It is assumed that the time for each execution of a function can be expressed by the total time for the function, divided by the number of times the function is called. Thus the time propagated along the call graph arcs to parents of that function is directly proportional to the number of times that arc is traversed.

Parents that are not profiled have the time of their profiled children propagated to them, but they appear to be spontaneously invoked in the call graph listing, and do not have their time propagated further. Similarly, signal catchers, even though profiled, appear to be spontaneous (although for more obscure reasons). Any profiled children of signal catchers should have their times propagated properly unless the signal catcher was invoked during the execution of the profiling routine, in which case all is lost.

The profiled program must call **exit()** (see *exit*(2)) or return normally, for the profiling information to be saved in the **gmon.out** file.

DEPENDENCIES

gprof cannot be used with dynamically linked executables.

AUTHOR

gprof was developed by the University of California, Berkeley.

FILES

a.out*	Default object file.
gmon.out*	Default dynamic call graph and profile.
gmon.sum*	Summarized dynamic call graph and profile.
/usr/lib/gprof.callg*	Call graph description.
/usr/lib/gprof.flat*	Flat profile description.

SEE ALSO

cc(1), f77(1), pc(1), prof(1), exit(2), profil(2), crt0(3), monitor(3C).

gprof: A Call Graph Execution Profiler; Graham, S.L., Kessler, P.B., McKusick, M.K.;

Proceedings of the SIGPLAN '82 Symposium on Compiler Construction; SIGPLAN Notices; Vol. 17, No. 6, pp. 120-126, June 1982.

NAME

grep, egrep, fgrep - search a file for a pattern

SYNOPSIS

Plain call with pattern

grep [**-E**|**-F**] [**-c**|**-l**|**-q**] [**-bhinsvx**] *pattern* [*file* ...]

Call with (multiple) –e pattern

grep [**-E**|**-F**] [**-c**|**-l**|**-q**] [**-bhinsvx**] **-e** *pattern*... [**-e** *pattern*] ... [*file* ...]

Call with –f file

grep [**-E**|**-F**] [**-c**|**-l**|**-q**] [**-bhinsvx**] [**-f** *pattern_file*] [*file* ...]

Obsolescent:

egrep [**-cefilnsv**] [*expression*] [*file* ...]

fgrep [**-cefilnsvx**] [*strings*] [*file* ...]

DESCRIPTION

The **grep** command searches the input text *files* (standard input default) for lines matching a pattern. Normally, each line found is copied to the standard output. **grep** supports the Basic Regular Expression syntax (see *regexp*(5)). The **-E** option (**egrep**) supports Extended Regular Expression (ERE) syntax (see *regexp*(5)). The **-F** option (**fgrep**) searches for fixed *strings* using the fast Boyer-Moore string searching algorithm. The **-E** and **-F** options treat newlines embedded in the *pattern* as alternation characters. A null expression or string matches every line.

The forms **egrep** and **fgrep** are maintained for backward compatibility. The use of the **-E** and **-F** options is recommended for portability.

Options

-E Extended regular expressions. Each pattern specified is a sequence of one or more EREs. The EREs can be separated by newline characters or given in separate **-e** *expression* options. A pattern matches an input line if any ERE in the sequence matches the contents of the input line without its trailing newline character. The same functionality is obtained by using **egrep**.

-F Fixed strings. Each pattern specified is a sequence of one or more strings. Strings can be separated by newline characters or given in separate **-e** *expression* options. A pattern matches an input line if the line contains any of the strings in the sequence. The same functionality is obtained by using **fgrep**.

-b Each line is preceded by the block number on which it was found. This is useful in locating disk block numbers by context. Block numbers are calculated by dividing by 512 the number of bytes that have been read from the file and rounding down the result.

-c Only a count of matching lines is printed.

-e *expression* Same as a simple *expression* argument, but useful when the *expression* begins with a hyphen (**-**). Multiple **-e** options can be used to specify multiple patterns; an input line is selected if it matches any of the specified patterns.

-f *pattern_file* The regular *expression* (**grep** and **grep -E**) or *strings* list (**grep -F**) is taken from the *pattern_file*.

-h Suppress printing of filenames when searching multiple files.

-i Ignore uppercase/lowercase distinctions during comparisons.

-l Only the names of files with matching lines are listed (once), separated by newlines. If standard input is searched, a path name of **(standard input)** will be written, in the POSIX locale. In other locales, **(standard input)** may be replaced by something more appropriate in those locales.

-n Each line is preceded by its relative line number in the file starting at 1. The line number is reset for each file searched. This option is ignored if **-c**, **-b**, **-l**, or **-q** is specified.

-q	(Quiet) Do not write anything to the standard output, regardless of matching lines. Exit with zero status upon finding the first matching line. Overrides any options that would produce output.
-s	Error messages produced for nonexistent or unreadable files are suppressed.
-v	All lines but those matching are printed.
-x	(eXact) Matches are recognized only when the entire input line matches the fixed string or regular expression.

The file name is output in all the cases in which output is generated if there are more than one input file, unless the -h option is specified. Care should be taken when using the characters **$**, *****, **[**, **^**, **|**, **(**, **)**, and \ in *expression*, because they are also meaningful to the shell. It is safest to enclose the entire *expression* argument in single quotes (**'**...**'**).

EXTERNAL INFLUENCES

Environment Variables

LANG determines the locale to use for the locale categories when both **LC_ALL** and the corresponding environment variable (beginning with **LC_**) do not specify a locale. If **LANG** is not specified or is set to the empty string, a default of **C** (see *lang*(5)) is used.

LC_ALL determines the locale to use to override any values for locale categories specified by the settings of **LANG** or any environment variables beginning with **LC_**.

LC_COLLATE determines the collating sequence used in evaluating regular expressions.

LC_CTYPE determines the interpretation of text as single byte and/or multi-byte characters, the classification of characters as letters, the case information for the **-i** option, and the characters matched by character class expressions in regular expressions.

LC_MESSAGES determines the language in which messages are displayed.

If any internationalization variable contains an invalid setting, the commands behave as if all internationalization variables are set to **C**. See *environ*(5).

International Code Set Support

Single-byte and multi-byte character code sets are supported.

RETURN VALUE

Upon completion, **grep** returns one of the following values:

0	One or more matches found.
1	No match found.
2	Syntax error or inaccessible file (even if matches were found).

EXAMPLES

In the Bourne shell (*sh*(1)) the following example searches two files, finding all lines containing occurrences of any of four strings:

```
grep -F 'if
then
else
fi' file1 file2
```

Note that the single quotes are necessary to tell **grep -F** when the strings have ended and the file names have begun.

For the C shell (see *csh*(1)) the following command can be used:

```
grep -F 'if\  then\  else\ fi' file1  file2
```

To search a file named **address** containing the following entries:

```
Ken   112 Warring St.  Apt. A
Judy  387 Bowditch  Apt. 12
Ann   429 Sixth St.
```

the command:

```
grep Judy address
```

prints:

```
Judy  387 Bowditch  Apt. 12
```

To search a file for lines that contain either a **Dec** or **Nov**, use either of the following commands:

```
grep -E '[Dd]ec|[Nn]ov' file
egrep -i 'dec|nov' file
```

Search all files in the current directory for the string **xyz**:

```
grep xyz *
```

Search all files in the current directory subtree for the string **xyz**, and ensure that no error occurs due to file name expansion exceeding system argument list limits:

```
find . -type f -print |xargs grep xyz
```

The previous example does not print the name of files where string **xyz** appears. To force **grep** to print file names, add a second argument to the **grep** command portion of the command line:

```
find . -type f -print |xargs grep xyz /dev/null
```

In this form, the first file name is that produced by **find**, and the second file name is the null file.

WARNINGS

(XPG4 only.) If the **-q** option is specified, the exit status will be zero if an input line is selected, even if an error was detected. Otherwise, default actions will be performed.

SEE ALSO

sed(1), sh(1), regcomp(3C), environ(5), lang(5), regexp(5).

STANDARDS CONFORMANCE

grep: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

egrep: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

fgrep: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

NAME

groups - show group memberships

SYNOPSIS

groups [**-p**] [**-g**] [**-l**] [*user*]

DESCRIPTION

groups shows the groups to which the caller or the optionally specified *user* belong. If invoked with no arguments, **groups** prints the current access list returned by **getgroups()** (see *getgroups*(2)).

Each user belongs to a group specified in the password file **/etc/passwd** and possibly to other groups as specified in the files **/etc/group** and **/etc/logingroup**. A user is granted the permissions of those groups specified in **/etc/passwd** and **/etc/logingroup** at login time. The permissions of the groups specified in **/etc/group** are normally available only with the use of **newgrp** (see *newgrp*(1)). If a user name is specified with no options, **groups** prints the union of all these groups.

The **-p**, **-g**, and **-l** options limit the printed list to those groups specified in **/etc/passwd**, **/etc/group**, and **/etc/logingroup**, respectively. If a user name is not specified with any of these options, **cuserid()** is called to determine the default user name (see *cuserid*(3S)).

The printed list of groups is sorted in ascending collation order (see Environment Variables below).

EXTERNAL INFLUENCES

Environment Variables

LC_COLLATE determines the order in which the output is sorted.

If **LC_COLLATE** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default. If **LANG** is not specified or is set to the empty string, a default of “C” (see *lang*(5)) is used instead of **LANG**. If any internationalization variable contains an invalid setting, **groups** behaves as if all internationalization variables are set to “C” (see *environ*(5)).

EXAMPLES

Check file **/etc/logingroup** and display all groups to which user **tim** belongs:

```
groups -l tim
```

AUTHOR

groups was developed by the University of California, Berkeley.

FILES

/etc/group
/etc/logingroup
/etc/passwd

SEE ALSO

id(1), newgrp(1), getgroups(2), initgroups(3C), cuserid(3S), group(4).

NAME

head - give first few lines

SYNOPSIS

head [**-c**|**-l**] [**-n** *count*] [*file* ...]

Obsolescent:

head [**-***count*] [*file* ...]

DESCRIPTION

head prints on standard output the first *count* lines of each of the specified files, or of the standard input. If *count* is omitted it defaults to 10.

If multiple *files* are specified, **head** outputs before each file a line of this form:

==> *file* **<==**

Options

-c	The quantity of output is measured in bytes.
-*count*	The number of units of output. This option is provided for backward compatibility (see **-n** below) and is mutually exclusive of all other options.
-l	The quantity of output is measured in lines; this is the default.
-n *count*	The number of lines (default) or bytes output. *count* is an unsigned decimal integer. If **-n** (or **-***count*) is not given, the default quantity is 10. This option provides the same functionality as the **-***count* option, but in a more standard way. Use of the **-n** option is recommended where portability between systems is important.

EXTERNAL INFLUENCES

Environment Variables

LC_CTYPE determines the interpretation of text within file as single and/or multi-byte characters.

LC_MESSAGES determines the language in which messages are displayed.

If **LC_CTYPE** or **LC_MESSAGES** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**.

If any internationalization variable contains an invalid setting, **head** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported.

WARNINGS

The length of the input lines is limited to {**LINE_MAX**} bytes.

SEE ALSO

tail(1), cat(1), more(1), pg(1).

STANDARDS CONFORMANCE

head: SVID3, XPG4, POSIX.2

NAME

hostname - set or display name of current host system

SYNOPSIS

hostname [*name_of_host*]

DESCRIPTION

The **hostname** command displays the name of the current host, as given in the **gethostname()** system call (see *gethostname*(2)). Users who have appropriate privileges can set the hostname by giving the argument *name_of_host*; this is usually done in the startup script **/sbin/init.d/hostname**. The *name_of_host* argument is restricted to **MAXHOSTNAMELEN** characters as defined in **<sys/param.h>**.

The system might be known by other names if networking products are supported. See the node manager documentation supplied with your system.

WARNINGS

If the *name_of_host* argument is specified, the resulting host name change lasts only until the system is rebooted. To change the host name permanently, run the special initialization script **/sbin/set_parms** (see *Using Your HP Workstation*).

Many types of networking services are supported on HP-UX, each of which uses a separately assigned system name and naming convention. To ensure predictable system behavior, it is essential that system names (also called host names or node names) be assigned in such a manner that they do not create conflicts when the various networking facilities interact with each other.

The system does not rely on a single system name in a specific location, partly because different services use dissimilar name formats as explained below. The **hostname** and **uname** commands assign system names as follows:

Node Name	**Command**	*name* **Format**	**Used By**
Internet name	**hostname** *name*	*sys*[**.***x***.***y***.***z*...]	ARPA and NFS Services
UUCP name	**uname -S** *name*	*sys*	**uucp** and related programs

where *sys* represents the assigned system name. It is *strongly* recommended that *sys* be identical for all commands and locations and that the optional **.***x***.***y***.***z*... follow the specified notation for the particular ARPA/NFS environment.

Internet names are also frequently called host names or domain names (which are different from NFS domain names). Refer to *hostname*(5) for more information about Internet naming conventions.

Whenever the system name is changed in any file or by the use of any of the above commands, it should also be changed in all other locations as well. Other files or commands in addition to those above (such as **/etc/uucp/Permissions** if used to circumvent **uname**, for example) may contain or alter system names. To ensure correct operation, they should also use the same system name.

System names are normally assigned by the **/sbin/init.d/hostname** script at start-up, and should not be altered elsewhere.

AUTHOR

hostname was developed by the University of California, Berkeley.

SEE ALSO

uname(1), gethostname(2), sethostname(2), uname(2), hostname(5).

Using Your HP Workstation

NAME

hp - handle special functions of HP 2640 and HP 2621-series terminals

SYNOPSIS

hp [**-e**] [**-m**]

DESCRIPTION

hp supports special functions of the Hewlett-Packard HP 2640 and HP 2621 series of terminals, with the primary purpose of producing accurate representations of most **nroff** output. A typical use is:

nroff -h *files* ... **|** **hp**

Regardless of the hardware options on a given terminal, **hp** tries to do sensible things with underlining and reverse line-feeds. If the terminal has the "display enhancements" feature, subscripts and superscripts can be indicated in distinct ways. If it has the "mathematical-symbol" feature, Greek and other special characters can be displayed.

Options

hp recognizes the following options:

-e Specify that your terminal has the "display enhancements" feature, to make maximal use of the added display modes. Overstruck characters are presented in the Underline mode. Superscripts are shown in Half-bright mode, and subscripts in Half-bright, Underlined mode. If this flag is omitted, **hp** assumes that your terminal lacks the "display enhancements" feature. In this case, all overstruck characters, subscripts, and superscripts are displayed in Inverse Video mode; that is, dark-on-light, rather than light-on-dark.

-m Request minimization of output by removing new-lines. Any contiguous sequence of 3 or more new-lines is converted into a sequence of only 2 new-lines; that is, any number of successive blank lines produces only a single blank output line. This allows you to retain more actual text on the screen.

DIAGNOSTICS

line too long
The representation of a line exceeds 1,024 characters.

RETURN VALUE

hp returns zero for normal termination, and 2 for all errors.

WARNINGS

An "overstriking sequence" is defined as a printing character followed by a backspace followed by another printing character. In such sequences, if either printing character is an underscore, the other printing character is shown underlined or in Inverse Video; otherwise, only the first printing character is shown (again, underlined or in Inverse Video). Nothing special is done if a backspace is adjacent to an ASCII control character. Sequences of control characters (e.g., reverse line-feeds, backspaces) can make text "disappear"; in particular, tables generated by **tbl** that contain vertical lines will often be missing the lines of text that contain the "foot" of a vertical line, unless the input to **hp** is piped through **col** (see *col*(1)).

Although some terminals do support numerical superscript characters, no attempt is made to display them.

SEE ALSO

col(1), neqn(1), nroff(1), tbl(1).

NAME

hyphen - find hyphenated words

SYNOPSIS

hyphen [*files*]

DESCRIPTION

hyphen finds all the hyphenated words ending lines in *files* and prints them on the standard output. If no arguments are given, the standard input is used; thus, **hyphen** can be used as a filter.

EXAMPLES

Prepare an **nroff** hyphenation proofreading file for *textfile*.

```
mm textfile | hyphen
```

WARNINGS

hyphen cannot cope with hyphenated *italics* (i.e., underlined) words; it often misses them completely or mangles them.

hyphen occasionally gets confused, but with no ill effects other than spurious extra output.

SEE ALSO

mm(1), nroff(1).

NAME

iconv - code set conversion

SYNOPSIS

iconv -f *fromcode* **-t** *tocode* [*file* ...]

DESCRIPTION

iconv converts the encoding of characters in the input files from the *fromcode* code set to the *tocode* code set, and writes the results on standard output. If no input files are given, **iconv** reads from standard input. If **-** appears as an input file name, **iconv** reads standard input at that point. **--** can be used to delimit the end of options (see *getopt*(3C)).

Options

iconv recognizes the following options:

-f *fromcode* Identify the code set corresponding to option argument *fromcode* as the code set that the input will be converted "from".

-t *tocode* Identify the code set corresponding to option argument *tocode* as the code set that the input will be converted "to".

The *fromcode* and *tocode* names can be any of the base and alias names listed in the iconv configuration file, **/usr/lib/nls/iconv/config.iconv**. See *iconv*(3C) for details and the configuration file for a list of supported code set names.

EXTERNAL INFLUENCES

Environment Variables

LANG provides a default value for the internationalization variables that are unset or null. If **LANG** is unset or null, the default value of "C" (see *lang*(5)) is used. If any of the internationalization variables contains an invalid setting, **iconv** will behave as if all internationalization variables are set to "C". See *environ*(5).

LC_ALL If set to a non-empty string value, overrides the values of all the other internationalization variables.

LC_CTYPE determines the interpretation of text as single and/or multi-byte characters, the classification of characters as printable, and the characters matched by character class expressions in regular expressions. During translation of the file, this variable is superseded by the use of the *fromcode* option-argument.

LC_MESSAGES determines the locale that should be used to affect the format and contents of diagnostic messages written to standard error and informative messages written to standard output.

NLSPATH determines the location of message catalogues for the processing of **LC_MESSAGES.**

International Code Set Support

Single and multi-byte character code sets are supported.

WARNINGS

If an input character does not have a valid equivalent in the code set selected by the **-t** option (the "to" code set), it is mapped to the "galley character", if it has been defined for that conversion. (see *genxlt*(1) and *iconv*(3C)).

If an input character does not belong to the code set selected by the **-f** option (the "from" code set), the command terminates.

EXAMPLES

Convert the contents of file **foo** from code set Roman8 to ISO 8859/1 and store the results in file **bar**.

```
iconv -f roman8 -t iso8859_1 foo > bar
```

FILES

/usr/lib/nls/iconv/config.iconv iconv configuration file

AUTHOR

iconv was developed by HP.

SEE ALSO

getopt(3C),iconv(3C)

STANDARDS CONFORMANCE

iconv: XPG2, XPG3, XPG4

NAME

id - print user and group IDs and names

SYNOPSIS

id [**-u**|**-g**|**-G**] [**-nr**] [**-P**] [*user*]

DESCRIPTION

The **id** command writes a message to standard output, giving the user and group IDs and names for the process. If the effective and real IDs are different, both are printed.

If the process has supplementary group affiliations (see *groups*(1)), the supplementary group affiliations are also written.

If the *user* operand is specified, and the effective user ID of the process is superuser, the user and group IDs of the selected user are written. In this case, effective IDs are assumed to be identical to real IDs.

Options

The following options modify the behavior described above.

-g Display only the group ID. The default is the effective group ID; to modify, use the **-r** option. If the process has supplementary group affiliations that are different from the effective group ID (or the real ID if the **-r** option is used), display each such affiliation on the same line. The default is decimal format; to modify, use the **-n** option.

-G Output all different group IDs (effective, real, and supplementary) only, using the format "%u\n". If there is more than one distinct group affiliation, output each such affiliation, using the format " %u", before the <newline> is output.

-n With A **-u**, **-g**, or **-G**, display the ID name instead of the ID number.

-r With **-u**, **-g**, or **-G**, display the real ID instead of the effective ID.

-u Display only the user ID. The default is the effective user ID; to modify, use the **-r** option. The default is decimal format; to modify, use the **-n** option.

-P Display the process resource group ID. See HP Process Resource Manager in DEPENDENCIES.

EXAMPLES

To display the current user and group data:

```
id
```

produces:

```
uid=1834(allanp) gid=20(users)
```

To display the group ID number for the current process:

```
id -g
```

produces:

```
20
```

To display the group name for the current process:

```
id -gn
```

produces:

```
users
```

To display the user and group data for another user:

```
id ralford
```

produces:

```
uid=329(ralford) gid=20(users)
```

if the effective user ID of the process is superuser. Otherwise, it produces the data for the invoking process.

i

AUTHOR

id was developed by HP and AT&T.

DEPENDENCIES

HP Process Resource Manager

The **-P** option requires that the optional HP Process Resource Manager (PRM) software be installed and configured. See *prmconfig*(1) for a description of how to configure HP PRM, and *prmconf*(4) for the definition of the process resource group.

SEE ALSO

groups(1), logname(1), getuid(2).

HP Process Resource Manager: prmconfig(1), prmconf(4) in *HP Process Resource Manager User's Guide*.

STANDARDS CONFORMANCE

id: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

NAME

ident - identify files in RCS

SYNOPSIS

ident *file* ...

DESCRIPTION

ident searches the named files for all occurrences of the pattern **$***keyword***:**...**$**, where *keyword* is one of the following:

```
Author      Log
Date        Revision
Header      Source
Locker      State
```

These patterns are normally inserted automatically by the RCS **co** command, but can also be inserted manually (see *co*(1)).

ident works on text files as well as object files. For example, if the C program in file **f.c** contains:

```
char rcsid[] = "$Header:  Header information $";
```

and **f.c** is compiled into **f.o**, the command:

```
ident f.c f.o
```

prints:

```
f.c:
     $Header: Header information $
f.o:
     $Header: Header information $
```

AUTHOR

ident was developed by Walter F. Tichy.

SEE ALSO

ci(1), co(1), rcs(1), rcsdiff(1), rcsintro(5), rcsmerge(1), rlog(1), rcsfile(4).

NAME

idlookup - identify the user of a particular TCP connection

SYNOPSIS

idlookup *host-or-ip-number local-port foreign-port*

DESCRIPTION

idlookup can be used to identify the user at the remote end of a TCP connection, assuming the host at the other end is running an Identification Server.

host-or-ip-number is the name of the host at the other end of the connection, or its IP address.

local-port and *foreign-port* are the port numbers, or service names of the ports at the two ends of the connection.

WARNING

Note that the references to *local-port* and *foreign-port* follow the terminology in RFC931, and are from the point of view of the *server* rather than the user.

AUTHOR

idlookup was originally written by Peter Eriksson. The manual page was originally written by Dave Sheild.

SEE ALSO

sendmail(1M), identd(1M), *RFC931*.

NAME

ied - input editor and command history for interactive programs

SYNOPSIS

ied [**-dirt**] [**-h** *file*] [**-s** *size*] [**-p** *prompt*] [**-k** *charmap*] *utility* [*arguments* ...]

DESCRIPTION

ied is a utility command that is intended to act as an interface between the user and an interactive program such as bc, bs, or Bourne shell, providing most of the line editing and history functionality found in the Korn shell. **ied** interprets the *utility* name as the command to be executed, and passes *arguments* as the arguments to the utility. Subsequent input to *utility* then has access to editing and history functions very similar to those provided by *ksh*.

ied monitors the state of the pty it uses to run the command, and, whenever the application it is running, changes the state from the state of the tty when **ied** started, **ied** becomes "transparent". This allows programs to do shell escapes to screen-smart programs. In general, **ied** should not in any way interfere with any action taken by any program for which it provides a front end. This includes Korn shell itself: in this case **ied** would provide history for any application that was run by **ksh**, and **ksh** would provide its own independent history. In a useful extreme case, **ied** can be used as a front end to the login shell (which might be **ksh** or **csh**). In this case, all applications that use normal line editing gain line editing and history, sharing a single history. The shell would continue to have its own independent history if it provides such a mechanism.

i

When **ied** is in its transparent mode, no history is saved. In particular the **ex** mode of **vi** does not use normal line editing (rather, it simulates it) and **ied** cannot provide history in this case. The **Subject:** and address line editing of mailx also cannot be edited with **ied**.

Options

Several options and command-line arguments control **ied**'s operation:

-d Debug mode. Print information about the operation of the program. It is best used to determine if a program puts **ied** into transparent mode unexpectedly.

-h *filename* Keep the history in a file named *filename*. If a file of that name already exists and is a history file, the latter part of it (the last *size* lines as specified by the **-s** option) is used as the initial value of the history. If the **-h** option is not used, the environment variable **IEDHISTFILE** is used to supply the name. If neither are present an unnamed temporary file is used, and no initial value is provided.

-i Force interactive mode. Normally **ied** simply **exec**s the command to which it is asked to be a front end when the standard input is not a tty (this allows aliases to be used for commands used in shells without interfering with their operation). This option forces **ied** to remain as a front end, and all editing functions are in place. This permits a utility that behaves differently in interactive and batch modes to be driven from a pipe or file in interactive mode. This is particularly useful in testing commands that make this distinction.

-k *charmap* *charmap* is a file of 256 or fewer lines. The line number in the file is the ordinal of a character as seen as input by **ied**, and the character on the line is the character generated as output (and also used as editing characters). This allows remapping of (ordinary) keys such as for a Dvorak keyboard. Characters must start in column one of each line, and be represented as 1-4 characters followed by a space or the new-line character for the next line. Characters after the space are ignored as comments. Single-character entries represent themselves. Two-character entries where the first character is a circumflex (^) converts the second character to the corresponding control character. Two-character sequences where the first character is backslash (\) use the C language conventions:

\n	newline	**\s**	space
\	escape	**\0**	null
\r	return	**\f**	form feed
\t	tab	**\v**	vertical tab
\b	backspace		

Three- and four-character sequences must be **\***nn* or **\***nnn*, giving the octal value for the character. If *charmap* is less than 256 lines long, the remaining characters are mapped to themselves.

-p *prompt* Many commands do not prompt when ready for input. **ied** approximates a prompting mechanism for such commands. This is not always perfectly successful, but for many commands it helps. In the worst case, the prompt is interspersed with output in the wrong location. *prompt* is a string as used in the format argument to *printf*(3S). The only **%** conversions that can be included are up to one instance of **%d** which is converted to the sequential number of the command, and any number of occurrences of **%%** which is treated as a literal **%** character. Prompting is suppressed when **ied** is operating in transparent mode.

-r This sets "non-raw" mode. Normally **ied** uses its own editing capabilities when reading simple text. This causes **ied** to use tty line discipline most of the time. The disadvantage of the default mode is that more context switches and general processing are required. The advantage is that **ied** is more transparent. For example, to specifically send an end-of-file in the non-raw mode requires that the end-of-file character (usually Ctrl-D) be followed by a carriage return. Similarly the “literal next” function (Ctrl-V) cannot escape the line-erase and line-kill functions in non-raw mode.

-s *size* This option specifies the size of the history buffer. When **ied** is started with an existing history file, approximately the last *size* lines are available to the history mechanism (the number is not guaranteed to be exactly *size*). Other lines in the file are retained until such time as **ied** is started on that history file and it exceeds approximately 4K bytes in size, at which time **ied** discards older entries at the beginning of the file until it is near 4 Kbytes in size. Since this occurs only at startup, history files can grow to be quite large between restarts. Larger values of *size* make the process image larger.

If **-s** is not specified, the value of the environment variable **IEDHISTSIZE** is used. If neither is specified, a default is used.

-t Set transparent mode. This forces **ied** to permanently be in transparent mode (as discussed above). It is primarily useful with **-i** for some classes of automated processing. In particular, it is useful for driving a command if the command takes as input what **ied** would interpret as editing characters. Thus with the appropriate combinations of **-i** and **-t**, it is possible to drive an editor such as **vi** or a screen-smart application from a batch file.

Should something go wrong with **ied**, the **SIGQUIT** signal, repeated 3 times, usually aborts **ied**. The exception is the case of a fully transparent application, where **ied** must be killed from another window or terminal. This is really relevant only when there is no way to direct the serviced process to terminate itself.

The editing capabilities of **ied** are essentially those found in **ksh**. Only those that differ from **ksh** are described below. As in **ksh**, the style of editing is determined from the environment variable **VISUAL**, or from **EDITOR** if **VISUAL** is not specified. The value examined should end in **vi**, **emacs**, or **gmacs** to specify an editor type. If it does not, **ied** does no editing, and history is not accessible.

In vi mode:

J Join lines. Considering the most recently edited line (which is empty immediately after a line is sent to the application) to be the “last line” of the history, the current line being displayed from the history is appended to the end of the last line, and the position in the history is reset to be at the last line which is then displayed. A space is inserted between the old and new text on the last line. The cursor is left on that space. Because **ied**'s understanding of line continuation is minimal, this is useful for editing long statements.

v Not supported.

V Not supported.

Sends nothing to the application, but inserts the line in the history (useful for adding comments to history file).

*<esc>***,*,=** (Filename expansion). Not supported.

i

@	Macro expansion. Not supported. Note however that **ksh** has a rarely-used function _ that substitutes words from the previous line (this is not the macro **$**_, but rather an editor command). If a preceding *count* is given, it uses the *count*th word of the last line. This is much more useful with **ied**.

In emacs/gmacs mode:

M-*, **M-=**, **M-**<*esc*>	(filename expansion) Not supported. Note that the command **M-.** (and it's synonym **M-**_) provide the same functionality as the vi mode _ command.
Macro expansion.	Not supported.
^O	Although supported, it may not always appear correctly on the screen. The **^L** command can be used to redraw the line. See below for the discussion on prompting.

EXAMPLES

Add interactive editing to the **bc** command:

```
ied bc
```

Execute **vi** on **testfile** using comands taken from **script**:

```
cat script | ied -i -t vi testfile
```

Note that without the use of **ied**, **vi** would misbehave because its standard input would not be a terminal device. In this case the **-t** is not required because **vi** puts itself in raw mode, but for an application that does not, **-t** might be required.

The command line

```
ied -i -t grep '^x:' data_file | tee x_lines
```

searches the file **data_file** for lines beginning with **x:**, sending one copy to the terminal and a second to file **x_lines**, just like the command line

```
grep '^x:' data_file | tee x_lines
```

The difference is that in the command line without **ied**, **grep** writes directly to a pipe, and thus buffers its output. If **data_file** is very large and not many lines match the pattern, output to the terminal is delayed. By using **ied**, the output of **grep** goes to a pty instead, which causes **grep** to output each line as it is ready.

WARNINGS

Since **ied** cannot know everything about every application, it is possible that it can become confused, with either the timing or the prompt being out of phase with the application. Since the use of **ied** is never required, it is the user's choice to determine whether the application is more usable with or without **ied**. In general, however, programs that do not confuse **ied** are usually also the most likely to benefit from its use.

ied tries to intuit the currently active prompt when it is not providing one itself. However, this is not always successful. Even when it is successful, the timing of **ied** and the serviced command may occasionally confuse the output. The **^L** commands in both emacs and vi modes redraw the edit line in a consistent fashion that can be used to create the next command.

AUTHOR

ied was developed by HP.

SEE ALSO

ksh(1).

NAME

insertmsg - use findstr(1) output to insert calls to catgets(3C)

SYNOPSIS

insertmsg [**-h**] [**-n***number*] [**-i***amount*] [**-s***number*] *stringlist*

DESCRIPTION

insertmsg examines the file *stringlist*, which is assumed to be the output of **findstr** after subsequent editing to remove any strings that do not need to be localized (see *findstr*(1)). If the **-h** option is specified, **insertmsg** places the following lines at the beginning of each file named in *stringlist*:

```
#ifndef NLS
#define catgets(i,sn,mn,s) (s)
#else NLS
#define NL_SETN number
#include <nl_types.h>
#endif NLS
```

where *number* is a set number defined by the **-s** option; the default is **1**. For each string in *stringlist*, **insertmsg** surrounds the string in the corresponding file with an expression of the form:

```
(catgets(catd,NL_SETN,msg_num,"default string"))
```

The *default string* is the original string referenced by the line in *stringlist*, and *msg_num* is replaced by the message number assigned to that string. The assigned message numbers begin with the number defined by the **-n** option and are incremented by the amount defined by the **-i** option. The default is **1** for both the starting message number and the increment. If *name***.c** is the file to be modified, as specified within the *stringlist* file, *insertmsg* places the modified source in **nl_***name***.c**. The user must then manually edit the file **nl_***name***.c** to insert the following statements:

```
nl_catd catd;
catd = catopen("appropriate message catalog",0);
```

The data type **nl_catd** is defined in <**nl_types.h**> and **catd** is a parameter to the calls to *catgets*, which are inserted for each string from *stringlist*.

insertmsg also sends to the standard output a file that can be used as input to **gencat** (see *gencat*(1)).

EXTERNAL INFLUENCES

Environment Variables

LC_CTYPE determines the interpretation of text as single- and/or multi-byte characters.

LC_MESSAGES determines the language in which messages are displayed.

If **LC_CTYPE** or **LC_MESSAGES** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**. If any internationalization variable contains an invalid setting, **insertmsg** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported.

DIAGNOSTICS

If **insertmsg** does not find opening or closing double quotes where required in the strings file, it prints **insertmsg exiting : lost in strings file** and aborts. If this happens, check the strings file to ensure that the lines that have been kept there have not been altered.

WARNINGS

If the **-h** option is not used, it may be necessary to manually add the following statement to the file created by **insertmsg**:

```
#include <nl_types.h>
```

insertmsg inserts a pointer to a static area that is overwritten on each call.

The **insertmsg** command is HP proprietary, not portable to other vendors' systems, and will not be provided in future HP-UX releases.

AUTHOR

insertmsg was developed by HP.

SEE ALSO

findstr(1), gencat(1), catgets(3C), catopen(3C).

NAME

iostat - report I/O statistics

SYNOPSIS

iostat [**-t**] [*interval* [*count*]]

DESCRIPTION

iostat iteratively reports I/O statistics for each active disk on the system. Disk data is arranged in a four-column format:

Column Heading	**Interpretation**
device	Device name
bps	Kilobytes transferred per second
sps	Number of seeks per second
msps	Milliseconds per average seek

If two or more disks are present, data is presented on successive lines for each disk.

To compute this information, seeks, data transfer completions, and the number of words transferred are counted for each disk. Also, the state of each disk is examined **HZ** times per second (as defined in <**sys/param.h**>) and a tally is made if the disk is active. These numbers can be combined with the transfer rates of each device to determine average seek times for each device.

With the advent of new disk technologies, such as data striping, where a single data transfer is spread across several disks, the number of milliseconds per average seek becomes impossible to compute accurately. At best it is only an approximation, varying greatly, based on several dynamic system conditions. For this reason and to maintain backward compatibility, the milliseconds per average seek (**msps**) field is set to the value 1.0.

Options

iostat recognizes the following options and command-line arguments:

-t Report terminal statistics as well as disk statistics. Terminal statistics include:

tin	Number of characters read from terminals.
tout	Number of characters written to terminals.
us	Percentage of time system has spent in user mode.
ni	Percentage of time system has spent in user mode running low-priority (*nice*) processes.
sy	Percentage of time system has spent in system mode.
id	Percentage of time system has spent idling.

interval Display successive lines which are summaries of the last *interval* seconds. The first line reported is for the time since a reboot and each subsequent line is for the last interval only.

count Repeat the statistics *count* times.

EXAMPLES

Show current I/O statistics for all disks:

```
iostat
```

Display I/O statistics for all disks every 10 seconds until INTERRUPT or QUIT is pressed:

```
iostat 10
```

Display I/O statistics for all disks every 10 seconds and terminate after 5 successive readings:

```
iostat 10 5
```

Display I/O statistics for all disks every 10 seconds, also show terminal and processor statistics, and terminate after 5 successive readings:

```
iostat -t 10 5
```

WARNINGS

Users of **iostat** must not rely on the exact field widths and spacing of its output, as these will vary depending on the system, the release of HP-UX, and the data to be displayed.

AUTHOR

iostat was developed by the University of California, Berkeley, and HP.

FILES

/usr/include/sys/param.h

SEE ALSO

vmstat(1).

NAME

ipcrm - remove a message queue, semaphore set, or shared memory identifier

SYNOPSIS

ipcrm [*option*]...

DESCRIPTION

The **ipcrm** command removes one or more specified message queue, semaphore set, or shared memory identifiers.

Options

The identifiers are specified by the following *option*s:

-m *shmid*	Remove the shared memory identifier *shmid* from the system. The shared memory segment and data structure associated with it are destroyed after the last detach.
-q *msqid*	Remove the message queue identifier *msqid* from the system and destroy the message queue and data structure associated with it.
-s *semid*	Remove the semaphore identifier *semid* from the system and destroy the set of semaphores and data structure associated with it.
-M *shmkey*	Remove the shared memory identifier, created with key *shmkey*, from the system. The shared memory segment and data structure associated with it are destroyed after the last detach.
-Q *msgkey*	Remove the message queue identifier, created with key *msgkey*, from the system and destroy the message queue and data structure associated with it.
-S *semkey*	Remove the semaphore identifier, created with key *semkey*, from the system and destroy the set of semaphores and data structure associated with it.

The details of the removals are described in *msgctl*(2), *shmctl*(2), and *semctl*(2). The identifiers and keys can be found by using **ipcs** (see *ipcs*(1)).

SEE ALSO

ipcs(1), msgctl(2), msgget(2), msgop(2), semctl(2), semget(2), semop(2), shmctl(2), shmget(2), shmop(2).

STANDARDS CONFORMANCE

ipcrm: SVID2, SVID3

NAME

ipcs - report status of interprocess communication facilities

SYNOPSIS

ipcs [**-mqs**] [**-abcopt**] [**-C** *core*] [**-N** *namelist*]

DESCRIPTION

ipcs displays certain information about active interprocess communication facilities. With no options, ipcs displays information in short format for the message queues, shared memory segments, and semaphores that are currently active in the system.

Options

The following options restrict the display to the corresponding facilities.

(none)	This is equivalent to **-mqs**.
-m	Display information about active shared memory segments.
-q	Display information about active message queues.
-s	Display information about active semaphores.

The following options add columns of data to the display. See "Column Description" below.

(none)	Display default columns: for all facilities: **T**, **ID**, **KEY**, **MODE**, **OWNER**, **GROUP**.
-a	Display all columns, as appropriate. This is equivalent to **-bcopt**.
-b	Display largest-allowable-size information: for message queues: **QBYTES**; for shared memory segments: **SEGSZ**; for semaphores: **NSEMS**.
-c	Display creator's login name and group name: for all facilities: **CREATOR**, **CGROUP**.
-o	Display information on outstanding usage: for message queues: **CBYTES**, **QNUM**; for shared memory segments: **NATTCH**.
-p	Display process number information: for message queues: **LSPID**, **LRPID**; for shared memory segments: **CPID**, **LPID**.
-t	Display time information: for all facilities: **CTIME**; for message queues: **STIME**, **RTIME**; for shared memory segments: **ATIME**, **DTIME**; for semaphores: **OTIME**.

The following options redefine the sources of information.

-C *core*	Use *core* in place of **/dev/kmem**. *core* can be a core file or a directory created by **savecrash** or **savecore**.
-N *namelist*	Use file *namelist* or the *namelist* within *core* in place of **/stand/vmunix**

Column Descriptions

The column headings and the meaning of the columns in an **ipcs** listing are given below. The columns are printed from left to right in the order shown below.

T Facility type:

m	Shared memory segment
q	Message queue
s	Semaphore

ID The identifier for the facility entry.

KEY The key used as an argument to **msgget()**, **semget()**, or **shmget()** to create the facility entry. (Note: The key of a shared memory segment is changed to **IPC_PRIVATE** when the segment has been removed until all processes attached to the segment detach it.)

MODE The facility access modes and flags: The mode consists of 11 characters that are interpreted as follows:

The first two characters can be:

R A process is waiting on a **msgrcv()**.

S	A process is waiting on a **msgsnd()**.
D	The associated shared memory segment has been removed. It will disappear when the last process attached to the segment detaches it.
C	The associated shared memory segment is to be cleared when the first attach is executed.
-	The corresponding special flag is not set.

The next 9 characters are interpreted as three sets of three characters each. The first set refers to the owner's permissions, the next to permissions of others in the group of the facility entry, and the last to all others.

Within each set, the first character indicates permission to read, the second character indicates permission to write or alter the facility entry, and the last character is currently unused.

r	Read permission is granted.
w	Write permission is granted.
a	Alter permission is granted.
-	The indicated permission is not granted.

OWNER	The login name of the owner of the facility entry.
GROUP	The group name of the group of the owner of the facility entry.
CREATOR	The login name of the creator of the facility entry.
CGROUP	The group name of the group of the creator of the facility entry.
CBYTES	The number of bytes in messages currently outstanding on the associated message queue.
QNUM	The number of messages currently outstanding on the associated message queue.
QBYTES	The maximum number of bytes allowed in messages outstanding on the associated message queue.
LSPID	The process ID of the last process to send a message to the associated message queue.
LRPID	The process ID of the last process to receive a message from the associated message queue.
STIME	The time the last **msgsnd()** message was sent to the associated message queue.
RTIME	The time the last **msgrcv()** message was received from the associated message queue.
CTIME	The time when the associated facility entry was created or changed.
NATTCH	The number of processes attached to the associated shared memory segment.
SEGSZ	The size of the associated shared memory segment.
CPID	The process ID of the creating process of the shared memory segment.
LPID	The process ID of the last process to attach or detach the shared memory segment.
ATIME	The time the last **shmat()** attach was completed to the associated shared memory segment.
DTIME	The time the last **shmdt()** detach was completed on the associated shared memory segment.
NSEMS	The number of semaphores in the set associated with the semaphore entry.
OTIME	The time the last **semop()** semaphore operation was completed on the set associated with the semaphore entry.

WARNINGS

ipcs produces only an approximate indication of actual system status because system processes are continually changing while **ipcs** is acquiring the requested information.

Do not rely on the exact field widths and spacing of the output, as these will vary depending on the system, the release of HP-UX, and the data to be displayed.

FILES

`/dev/kmem`	Kernel virtual memory
`/etc/group`	Group names
`/etc/passwd`	User names
`/stand/vmunix`	System namelist

SEE ALSO

msgop(2), semop(2), shmop(2).

STANDARDS CONFORMANCE

`ipcs`: SVID2, SVID3

NAME

join - relational database operator

SYNOPSIS

join [*options*] *file1 file2*

DESCRIPTION

join forms, on the standard output, a join of the two relations specified by the lines of *file1* and *file2*. If *file1* or *file2* is **-**, the standard input is used.

file1 and *file2* must be sorted in increasing collating sequence (see Environment Variables below) on the fields on which they are to be joined; normally the first in each line.

The output contains one line for each pair of lines in *file1* and *file2* that have identical join fields. The output line normally consists of the common field followed by the rest of the line from *file1*, then the rest of the line from *file2*.

The default input field separators are space, tab, or new-line. In this case, multiple separators count as one field separator, and leading separators are ignored. The default output field separator is a space.

Some of the below options use the argument *n*. This argument should be a **1** or a **2** referring to either *file1* or *file2*, respectively.

Options

-a *n* In addition to the normal output, produce a line for each unpairable line in file *n*, where *n* is **1** or **2**.

-e *s* Replace empty output fields by string *s*.

-j *m* Join on field *m* of both files. The argument *m* must be delimited by space characters. This option and the following two are provided for backward compatibility. Use of the **-1** and **-2** options (see below) is recommended for portability.

-j1 *m* Join on field *m* of *file1*.

-j2 *m* Join on field *m* of *file2*.

-o *list* Each output line comprises the fields specified in *list*, each element of which has the form *n***.***m*, where *n* is a file number and *m* is a field number. The common field is not printed unless specifically requested.

-t *c* Use character *c* as a separator (tab character). Every appearance of *c* in a line is significant. The character *c* is used as the field separator for both input and output.

-v *file_number*
Instead of the default output, produce a line only for each unpairable line in *file_number*, where *file_number* is **1** or **2**.

-1 *f* Join on field *f* of file 1. Fields are numbered starting with 1.

-2 *f* Join on field *f* of file 2. Fields are numbered starting with 1.

EXTERNAL INFLUENCES

Environment Variables

LC_COLLATE determines the collating sequence **join** expects from input files.

LC_CTYPE determines the alternative blank character as an input field separator, and the interpretation of data within files as single and/or multi-byte characters. **LC_CTYPE** also determines whether the separator defined through the **-t** option is a single- or multi-byte character.

If **LC_COLLATE** or **LC_CTYPE** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**. If any internationalization variable contains an invalid setting, **join** behaves as if all internationalization variables are set to "C" (see *environ*(5)).

International Code Set Support

Single- and multi-byte character code sets are supported with the exception that multi-byte-character file names are not supported.

EXAMPLES

The following command line joins the password file and the group file, matching on the numeric group ID, and outputting the login name, the group name, and the login directory. It is assumed that the files have been sorted in the collating sequence defined by the **LC_COLLATE** or **LANG** environment variable on the group ID fields.

```
join -1 4 -2 3 -o 1.1 2.1 1.6 -t: /etc/passwd /etc/group
```

The following command produces an output consisting all possible combinations of lines that have identical first fields in the two sorted files *sf1* and *sf2*, with each line consisting of the first and third fields from **sorted_file1** and the second and fourth fields from **sorted_file2**:

```
join -j1 1 -j2 1 -o 1.1,2.2,1.3,2.4 sorted_file1 sorted_file2
```

WARNINGS

With default field separation, the collating sequence is that of **sort -b**; with **-t**, the sequence is that of a plain sort.

The conventions of **join**, **sort**, **comm**, **uniq**, and **awk** are incongruous.

Numeric filenames may cause conflict when the **-o** option is used immediately before listing filenames.

AUTHOR

join was developed by OSF and HP.

SEE ALSO

awk(1), comm(1), sort(1), uniq(1).

STANDARDS CONFORMANCE

join: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

NAME

kermit - C-Kermit 6.0.192 communications software for serial and network connections: modem dialing, file transfer and management, terminal connection, character-set translation, and script programming.

SYNOPSIS

kermit [*command-file*] [*options* ...]

DESCRIPTION

Kermit is a family of file transfer, management, and communication software programs from Columbia University available for most computers and operating systems. The version of Kermit for Hewlett-Packard HP-UX, called **C-Kermit**, supports both serial connections (direct or dialed) and TCP/IP connections. C-Kermit can be thought of as a user-friendly and powerful alternative to cu, tip, uucp, ftp, and telnet; a single package for both network and serial communications, offering automation, convenience, and language features not found in the other packages, and having a great deal in common with its cousins, C-Kermit on other UNIX platforms, Kermit 95 for Windows 95 and NT and OS/2, MS-DOS Kermit for PCs with DOS and Windows 3.x, and IBM Mainframe Kermit-370 for VM/CMS, MVS/TSO, and CICS. C-Kermit itself also runs on Digital VMS, Data General AOS/VS, Stratus VOS, OS-9, QNX, the BeBox, Plan 9, the Commodore Amiga, and elsewhere. Together, C-Kermit, Kermit 95, MS-DOS Kermit, and IBM Mainframe Kermit offer a consistent and nearly universal approach to inter-computer communications.

C-Kermit 6.0.192 is Copyright (C) 1985, 1996 by the Trustees of Columbia University in the City of New York. The C-Kermit software may not be, in whole or in part, licensed or sold for profit as a software product itself, nor may it be included in or distributed with commercial products or otherwise distributed by commercial concerns to their clients or customers without written permission of the Office of Kermit Development and Distribution, Columbia University. This copyright notice must not be removed, altered, or obscured.

j

C-Kermit 6.0.192 is included with HP-UX 10.0 and later HP-UX releases by Hewlett-Packard in partnership with the Office of Kermit Development and Distribution, Columbia University.

C-Kermit 6.0 is thoroughly documented in the book *Using C-Kermit* by Frank da Cruz and Christine M. Gianone, Digital Press, Second Edition, 1997; see REFERENCES at the end of this manual page. This manual page is not a substitute for the book. If you are a serious user of C-Kermit, particularly if plan to write C-Kermit script programs, you should purchase the manual. Book sales are the primary source of funding for the nonprofit Kermit Project.

Any new features added since the most recent edition of the book was published are documented in the online file *ckcker.upd*. Hints, tips, limitations, restrictions are listed in *ckcker.bwr* (general C-Kermit) and *ckuker.bwr* (UNIX-specific); see FILES below. Please consult all of these references before reporting problems or asking for technical support.

Kermit software is available for hundreds of different computers and operating systems from Columbia University. For best file-transfer results, please use C-Kermit in conjunction with real Columbia University Kermit software on other computers, such as Kermit 95 for Windows 95 and NT or MS-DOS Kermit for DOS 3.x or Windows. See CONTACTS below.

MODES OF OPERATION

C-Kermit can be used in two "modes": remote and local. In **remote mode**, you connect to the HP-UX system from a desktop computer and transfer files between your desktop computer and HP-UX C-Kermit. In that case, connection establishment (dialing, TELNET connection, etc.) is handled by the Kermit program on your desktop computer.

In **local mode**, C-Kermit establishes a connection to another computer by direct serial connection, by dialing a modem, or by making a network connection. When used in local mode, C-Kermit gives you a terminal connection to the remote computer, using your actual terminal, emulator, or UNIX workstation terminal window or console driver for specific terminal emulation.

C-Kermit also has two types of commands: the familiar UNIX-style command-line options, and an interactive dialog with a prompt. **Command-line options** give you access to a small but useful subset of C-Kermit's features for terminal connection and file transfer, plus the ability to pipe files into or out of Kermit for transfer.

Interactive commands give you access to dialing, script programming, character-set translation, and, in general, detailed control and display of all C-Kermit's features. Interactive commands can also be collected into command files or macros.

STARTING C-KERMIT

You can start C-Kermit by typing "/usr/bin/kermit", or just "kermit" if your PATH includes "/usr/bin", possibly followed by command-line options. If there are no "action options" on the command line (explained below), C-Kermit starts in interactive command mode; you will see a greeting message and then the "C-Kermit>" prompt. If you do include action options on the command line, C-Kermit takes the indicated actions and then exits directly back to UNIX. Either way, C-Kermit executes the commands in its initialization file, **/usr/share/lib/kermit/ckermit.ini**, before it executes any other commands, unless you have included the '**-Y**' (uppercase) command-line option, which means to skip the initialization file, or you have included the '**-y** *filename*' option to specify an alternative initialization file.

FILE TRANSFER

Here is the most common scenario for Kermit file transfer. Many other methods are possible, most of them more convenient, but this basic method should work in all cases.

- Start Kermit on your local computer and establish a connection to the remote computer. If C-Kermit is on your local computer, use the sequence SET MODEM TYPE *modem-name*, SET LINE *device-name*, SET SPEED *bits-per-second*, and DIAL *phone-number* if you are dialing; SET LINE and SPEED for direct connections; SET NETWORK *network-type* and SET HOST *host-name-or-address* for network connections.
- SET any other necessary communication parameters, such as PARITY, DUPLEX, and FLOW-CONTROL.
- Give the CONNECT command.
- Log in to the remote computer.
- Start Kermit on the remote computer, give it any desired SET commands for file-, communication-, or protocol-related parameters. If you will be transferring binary files, give the command SET FILE TYPE BINARY to the Kermit program that will be sending them.
- To **download** a file or file group, give the remote Kermit a SEND command, following by a filename or "wildcard" file specification, for example:

  ```
  send oofa.txt          #  (send one file)
  send oofa.*            #  (send a group of files)
  ```

  To **upload** a file or files, give the remote Kermit a RECEIVE command. The sending Kermit will tell the receiving Kermit the name (and other attributes) of each file.
- Escape back to the Kermit program on your local (desktop) computer. If your local computer is running C-Kermit, type Ctrl-\ c (Control-backslash followed by the letter 'c') (on NeXT workstations, use Ctrl-] c). If MS-DOS or OS/2 or Windows Kermit, use Alt-x (hold down the Alt key, press 'x'). Now you should see your local Kermit program's prompt.
- If you will be transferring binary files, give the command SET FILE TYPE BINARY to the Kermit program that is sending the files.
- If you are *downloading* files, tell the local Kermit program to RECEIVE. If you are *uploading*, give your local Kermit program a SEND command, specifying a filename or wildcard file specification. In other words, tell the *remote* Kermit program what to do first, SEND or RECEIVE, then escape back to the *local* Kermit and give it the opposite command, RECEIVE or SEND.
- When the transfer is complete, give a CONNECT command. Now you are talking to Kermit on the remote computer again. Type EXIT to get back to the command prompt on the remote computer. When you are finished using the remote computer, log out and then (if necessary) escape back to Kermit on your local computer. Then you can make another connection or EXIT from the local Kermit program.

C-Kermit's file transfer protocol defaults are deliberately conservative, resulting in file transfer that almost always works, but might be somewhat slow. To increase file transfer performance on computers and connections that permit it, use SET RECEIVE PACKET-LENGTH to increase the packet length, SET WINDOW to increase the packet window size, and use SET PREFIXING to reduce the overhead of control-character prefixing. (Hint: try the **FAST** command to enable all these performance options at once.) On serial connections, use hardware flow control (SET FLOW RTS/CTS) if available, rather than software (XON/XOFF) flow control. On TCP/IP connections, SET FLOW NONE. For details, including benchmarks, read Chapter 12 of *Using C-Kermit*.

j

OTHER FEATURES

C-Kermit includes features too numerous to be explained in a man page. For further information about connection establishment, modem dialing, networks, terminal connection, key mapping, logging, file transfer options and features, troubleshooting, client/server operation, character-set translation during terminal connection and file transfer, "raw" up- and downloading of files, macro construction, script programming, convenience features, and shortcuts, plus numerous tables, examples, and illustrations, please consult *Using C-Kermit*.

HELP

C-Kermit has extensive built-in help. You can find out what commands exist by typing ? at the C-Kermit> prompt. You can type HELP at the C-Kermit> prompt for "getting-started" message, or HELP followed by the name of a particular command for information about that command, for example:

```
help send
help set file
```

You can type ? anywhere within a command to get brief help about the current command field. You can also type the INTRO command to get a brief introduction to C-Kermit, and the NEWS command to find out what's new in your version. Finally, you can use the BUG command to learn how to report bugs.

ENTERING COMMANDS

You can use upper or lower case for interactive-mode commands, but remember that UNIX filenames are case-sensitive. You can abbreviate commands as long as the abbreviation matches only one possibility. While typing a command, you can use the following editing characters:

Delete, Backspace, or Rubout erases the rightmost character.
Ctrl-W erases the rightmost "word".
Ctrl-U erases the current command line.
Ctrl-R redisplays the current command.
Ctrl-P recalls a previous command (scrolls back in command buffer).
Ctrl-N scrolls forward in a scrolled-back command buffer.
Ctrl-C cancels the current command.
Tab, Esc, or Ctrl-I tries to complete the current keyword or filename.
? gives help about the current field.

To enter the command and make it execute, press the Return or Enter key.

BACKSLASH NOTATION

Within an interactive command, the "\" character (backslash) is a prefix used to enter special quantities, including ordinary characters that would otherwise be illegal. At the end of a line, \ or - (dash) makes the next line a continuation of the current line. Other than that, the character following the \ identifies what the special quantity is:

%	A user-defined simple (scalar) variable such as \%a or \%1
&	an array reference such as \&a[3]
$	an environment variable such as \$(TERM)
v (or V)	a built-in variable such as \v(time)
f (or F)	a function such as \Fsubstring(\%a,3,2)
d (or D)	a decimal (base 10) number (1 to 3 digits, 0..255) such as \d27
o (or O)	an octal (base 8) number (1 to 3 digits, 0..377) such as \o33
x (or X)	a hexadecimal (base 16) number (2 digits, 00..ff) like \x1b
\	the backslash character itself
b (or B)	the BREAK signal (OUTPUT command only)
l (or L)	a Long BREAK signal (OUTPUT only)
n (or N)	a NUL (0) character (OUTPUT only)
a decimal digit	a 1-, 2-, or 3-digit decimal number, such as \27
{}	used for grouping, e.g. \{27}123
anything else:	following character taken literally.

Note that numbers turn into the character with that binary code (0-255), so you can use \7 for a bell, \13 for carriage return, \10 for linefeed. For example, to have C-Kermit send a BELL to your screen, type:

```
echo \7
```

COMMAND LIST

The commands most commonly used, and important for beginners to know, are marked with "*":

Program Management

BUG	Learn how to report bugs.
CHECK	See if a particular feature is configured.
CLOSE	Close a log or other local file.
COMMENT	Introduce a full-line comment.
DATE	Display current date and time.
* EXIT	Leave the program, return to UNIX.
* HELP	Display a help message for a given command.
* INTRO	Print a brief introduction to C-Kermit.
LOG	Open a log file -- debugging, packet, session, transaction.
PUSH	Invoke local system's interactive command interpreter.
QUIT	Synonym for EXIT.
REDIRECT	Redirect standard I/O of command to communication device.
RUN	Run a program or system command.
SET COMMAND	Command-related parameters: bytesize, recall buffer size.
SET DEBUG	Log or display debugging information.
SET EXIT	Items related to C-Kermit's action upon exit or SET LINE/HOST.
SET PROMPT	The C-Kermit program's interactive command prompt.
SHOW EXIT	Display SET EXIT parameters.
SHOW FEATURES	Show features that C-Kermit was built with.
SHOW VERSIONS	Show version numbers of each source module.
SUSPEND	Suspend Kermit (use only if shell supports job control!).
* SHOW	Display values of SET parameters.
* TAKE	Execute commands from a file.
VERSION	Display the C-Kermit program version number.
Z	Synonym for SUSPEND.
Ctrl-C	Interrupt a C-Kermit command in progress.
Ctrl-Z	Synonym for SUSPEND.
; or #	Introduce a full-line or trailing comment.
! or @	Synonym for RUN.
<	Synonym for REDIRECT.

Connection Establishment and Release

* DIAL	Dial a telephone number.
* HANGUP	Hang up the phone or network connection.
PAD	Command for X.25 PAD (SunOS / Solaris / VOS only).
PING	Check status of remote TCP/IP host.
REDIAL	Dial the most recently DIALed number again.
SET CARRIER	Treatment of carrier on terminal connections.
* SET DIAL	Parameters related to modem dialing.
* SET FLOW	Communication line flow control: AUTO, RTS/CTS, XON/XOFF, etc.
* SET HOST	Specify remote network host name or address.
* SET LINE	Specify serial communication device name, like /dev/cul0p0.
* SET MODEM TYPE	Specify type of modem on SET LINE device, like HAYES.
* SET NETWORK	Network type, TCP/IP or X.25 (SunOS / Solaris / VOS only).
SET TCP	Specify TCP protocol options (advanced).
SET TELNET	Specify TELNET protocol options.
SET PAD	X.25 X.3 PAD parameters (SunOS / Solaris / VOS only).
* SET PARITY	Character parity (none, even, etc.) for communications.
* SET SPEED	Serial communication device speed, e.g. 2400, 9600, 57600.
SET X.25	Specify X.25 connection parameters (SunOS / Solaris / VOS only).
SHOW COMM	Display all communications settings.
SHOW DIAL	Display SET DIAL values.
SHOW MODEM	Display modem type, signals, etc.
SHOW NETWORK	Display network-related items.
* TELNET	= SET NETWORK TCP/IP, SET HOST ..., CONNECT.
TELOPT	Send a TELNET option negotiation (advanced).

Terminal Connection

* C	Special abbreviation for CONNECT.
* CONNECT	Establish a terminal connection to a remote computer.
SET COMMAND	Bytesize between C-Kermit and your keyboard and screen.
* SET DUPLEX	Specify which side echoes during CONNECT.
SET ESCAPE	Prefix for "escape commands" during CONNECT.
SET KEY	Key mapping and macros for use in CONNECT mode.
SET TERMINAL	Terminal connection items: bytesize, character-set, echo, etc.
SHOW ESCAPE	Display current CONNECT-mode escape character.
SHOW KEY	Display keycode and assigned value or macro.
SHOW TERMINAL	Display SET TERMINAL items.
* Ctrl-\	CONNECT-mode escape character, followed by another character: C to return to C-Kermit> prompt. B to send BREAK signal. ? to see other options.

File Transfer

ADD	Add a file specification to the SEND-LIST.
LOG SESSION	Download a file with no error checking.
MOVE	Send a file and then delete it.
MMOVE	Multiple MOVE - accepts a list of files, separated by spaces.
MSEND	Multiple SEND - accepts a list of files, separated by spaces.
* RECEIVE	Passively wait for files to arrive from other Kermit.
* R	Special abbreviation for RECEIVE.
* SEND	Send files.
* S	Special abbreviation for SEND.
REGET	Continue an incomplete download from a server.
RESEND	Continue an incomplete transmission.
PSEND	Send part of a file.
SET ATTRIB	Control transmission of file attributes.
* SET BLOCK	Choose error-checking level, 1, 2, or 3.
SET BUFFERS	Size of send and receive packet buffers.
SET PREFIX	Which control characters to "unprefix" during file transfer.
SET DELAY	How long to wait before sending first packet.
SET DESTINATION	DISK, PRINTER, or SCREEN for incoming files.
* SET FILE	Transfer mode (type), character-set, collision action, etc.
* SET RECEIVE	Parameters for inbound packets: packet-length, etc.
SET REPEAT	Repeat-count compression parameters.
SET RETRY	Packet retransmission limit.
SET SEND	Parameters for outbound packets: length, etc.
SET HANDSHAKE	Communication line half-duplex packet turnaround character.
SET LANGUAGE	Enable language-specific character-set translations.
SET SESSION-LOG	File type for session log, text or binary.
SET TRANSFER	File transfer parameters: character-set, display, etc.
SET TRANSMIT	Control aspects of TRANSMIT command execution.
SET UNKNOWN	Specify handling of unknown character sets.
* SET WINDOW	File transfer packet window size, 1-31.
SHOW ATTRIB	Display SET ATTRIBUTE values.
SHOW CONTROL	Display control-character prefixing map.
* SHOW FILE	Display file-related settings.
SHOW PROTOCOL	Display protocol-related settings.
SHOW LANGUAGE	Display language-related settings.
SHOW TRANSMIT	Display SET TRANSMIT values.
* STATISTICS	Display statistics about most recent file transfer.
TRANSMIT	Upload a file with no error checking.
XMIT	Synonym for TRANSMIT.

File Management

* CD	Change Directory.
* DELETE	Delete a file or files.
* DIRECTORY	Display a directory listing.

MAIL	Send a file to other Kermit, to be delivered as e-mail.
PRINT	Print a local file on a local printer.
* PWD	Display current working directory.
RENAME	Change the name of a local file.
SET PRINTER	Choose printer device.
SPACE	Display current disk space usage.
SHOW CHARACTER-SETS	Display character-set translation info.
TRANSLATE	Translate a local file's character set.
TYPE	Display a file on the screen.
XLATE	Synonym for TRANSLATE.

Client/Server Operation

BYE	Terminate a remote Kermit server and log out its job.
DISABLE	Disallow access to selected features during server operation.
E-PACKET	Send an Error packet.
ENABLE	Allow access to selected features during server operation.
FINISH	Instruct a remote Kermit server to exit, but not log out.
G	Special abbreviation for GET.
GET	Get files from a remote Kermit server.
RETRIEVE	Like GET but server deletes files after.
REMOTE xxx	Command for server, can be redirected with > or |.
REMOTE CD	Tell remote Kermit server to change its directory.
REMOTE ASSIGN	Assign a variable.
REMOTE DELETE	Tell server to delete a file.
REMOTE DIRECTORY	Ask server for a directory listing.
REMOTE HELP	Ask server to send a help message.
REMOTE HOST	Ask server to ask its host to execute a command.
REMOTE KERMIT	Send an interactive Kermit command to the server.
REMOTE LOGIN	Authenticate yourself to a remote Kermit server.
REMOTE LOGOUT	Log out from a Kermit server previously LOGIN'd to.
REMOTE PRINT	Print a local file on the server's printer.
REMOTE QUERY	Get value of a variable.
REMOTE SET	Send a SET command to a remote server.
REMOTE SPACE	Ask server how much disk space it has left.
REMOTE TYPE	Ask server to display a file on your screen.
REMOTE WHO	Ask server for a "who" or "finger" listing.
SERVER	Be a Kermit server.
SET SERVER	Parameters for server operation.
SHOW SERVER	Show SET SERVER, ENABLE/DISABLE items.

j

Script programming

ASK	Prompt the user, store user's reply in a variable.
ASKQ	Like ASK, but does not echo (useful for passwords).
ASSIGN	Assign an evaluated string to a variable or macro.
CLEAR	Clear communication device input buffer.
CLOSE	Close a log or other local file.
DECLARE	Declare an array.
DECREMENT	Subtract one (or other number) from a variable.
DEFINE	Define a variable or macro.
DO	Execute a macro ("DO" can be omitted).
ECHO	Display text on the screen.
ELSE	Used with IF.
END	A command file or macro.
EVALUATE	An arithmetic expression.
FOR	Execute commands repeatedly in a counted loop.
FORWARD	GOTO in the forward direction only.
GETC	Issue a prompt, get one character from keyboard.
GETOK	Ask question, get Yes or No answer, set SUCCESS or FAILURE.
GOTO	Go to a labeled command in a command file or macro.
IF	Conditionally execute the following command.
INCREMENT	Add one (or other number) to a variable.

INPUT	Match characters from another computer against a given text.
LOCAL	Declares local variables in a macro.
MINPUT	Like INPUT, but allows several match strings.
MSLEEP	Sleep for given number of milliseconds.
OPEN	Open a local file for reading or writing.
O	Special abbreviation for OUTPUT.
OUTPUT	Send text to another computer.
PAUSE	Do nothing for a given number of seconds.
READ	Read a line from a local file into a variable.
REINPUT	Reexamine text previously received from another computer.
RETURN	Return from a user-defined function.
SCRIPT	Execute a UUCP-style login script.
SET ALARM	Set a timer to be used with IF ALARM; SHOW ALARM shows it.
SET CASE	Treatment of alphabetic case in string comparisons.
SET COMMAND	QUOTING turns on/off interpretation of backslash notation.
SET COUNT	For counted loops.
SET INPUT	Control behavior of INPUT command.
SET MACRO	Control aspects of macro execution.
SET TAKE	Control aspects of TAKE file execution.
SHOW ARGUMENTS	Display arguments to current macro.
SHOW ARRAYS	Display information about active arrays.
SHOW COUNT	Display current COUNT value.
SHOW FUNCTIONS	List names of available \f() functions.
SHOW GLOBALS	List defined global variables \%a..\%z.
SHOW MACROS	List one or more macro definitions.
SHOW SCRIPTS	Show script-related settings.
SHOW VARIABLES	Display values all \v() variables.
SLEEP	Sleep for given number of seconds.
STOP	Stop executing macro or command file, return to prompt.
UNDEFINE	Undefine a variable.
WAIT	Wait for the specified modem signals.
WHILE	Execute commands repeatedly while a condition is true.
WRITE	Write material to a local file.
WRITE-LINE	Write a line (record) to a local file.
WRITELN	Synonym for WRITE-LINE.
XECHO	Like ECHO but no CRLF at end.
XIF	Extended IF command.

BUILT-IN VARIABLES

Built-in variables are referred to by \v(name), can be used in any command, usually used in script programming. They cannot be changed. Type SHOW VARIABLES for a current list.

\v(argc)	Number of arguments in current macro
\v(args)	Number of program command-line arguments
\v(charset)	Current file character-set
\v(cmdfile)	Name of current command file, if any
\v(cmdlevel)	Current command level
\v(cmdsource)	Where command are currently coming from, macro, file, etc.
\v(cols)	Number of screen columns
\v(connection)	Connection type: serial, tcp/ip, etc.
\v(count)	Current COUNT value
\v(cps)	Speed of most recent file transfer in chars per second
\v(cpu)	CPU type C-Kermit was built for
\v(crc16)	16-bit CRC of most recent file transfer
\v(d$ac)	SET DIAL AREA-CODE value
\v(d$cc)	SET DIAL COUNTRY-CODE value
\v(d$ip)	SET DIAL INTL-PREFIX value
\v(d$lc)	SET DIAL LD-PREFIX value
\v(date)	Date as 8 Feb 1993
\v(day)	Day of week
\v(directory)	Current/default directory

\v(dialstatus)	Return code from DIAL command (0 = OK, 22 = BUSY, etc.)
\v(download)	Current download directory if any
\v(errno)	Current "errno" (system error number) value
\v(errstring)	Error message string associated with errno
\v(evaluate)	Result of most recent EVALUATE command
\v(exitstatus)	Current EXIT status (0 = good, nonzero = something failed)
\v(filespec)	Filespec given in most recent SEND/RECEIVE/GET command
\v(fsize)	Size of file most recently transferred
\v(ftype)	SET FILE TYPE value (text, binary)
\v(home)	Home directory
\v(host)	Computer host name (computer where C-Kermit is running)
\v(input)	Current INPUT buffer contents
\v(inchar)	Character most recently INPUT
\v(incount)	How many characters arrived during last INPUT
\v(inidir)	Directory where initialization file was found
\v(instatus)	Status of most recent INPUT command
\v(line)	Current communications device, set by LINE or HOST
\v(local)	0 if in remote mode, 1 if in local mode
\v(macro)	Name of currently executing macro, if any
\v(minput)	Result of most recent MINPUT command
\v(modem)	Current modem type
\v(m_aa_off)	Modem command to turn autoanswer off
\v(m_aa_on)	Modem command to turn autoanswer on
\v(m_dc_off)	Modem command to turn data compression off
\v(m_dc_on)	Modem command to turn data compression on
\v(m_dial)	Telephone number most recently dialed
\v(m_ec_off)	Modem command to turn error correction off
\v(m_ec_on)	Modem command to turn error correction on
\v(m_fc_hw)	Modem command to turn hardware flow control on
\v(m_fc_no)	Modem command to turn flow control off
\v(m_fc_sw)	Modem command to turn software flow control on
\v(m_hup)	Modem command to hang up connection
\v(m_init)	Modem initialization string
\v(m_pulse)	Modem command to select pulse dialing
\v(m_tone)	Modem command to select tone dialing
\v(ndate)	Current date as 19930208 (yyyymmdd)
\v(nday)	Numeric day of week (0 = Sunday)
\v(newline)	System-independent newline character or sequence
\v(ntime)	Current local time in seconds since midnight (noon = 43200)
\v(packetlen)	Current SET RECEIVE PACKET-LENGTH value
\v(parity)	Current parity setting
\v(platform)	Specific machine and/or operating system
\v(program)	Name of this program ("C-Kermit")
\v(query)	Result of most recent REMOTE QUERY command
\v(protocol)	Currently selected file transfer protocol
\v(return)	Most recent RETURN value
\v(rows)	Number of rows on the terminal screen
\v(speed)	Current speed, if known, or "unknown"
\v(status)	0 or 1 (SUCCESS or FAILURE of previous command)
\v(sysid)	Kermit attribute code for system ID
\v(system)	UNIX
\v(terminal)	Terminal type
\v(tfsize)	Total size of file group most recently transferred
\v(time)	Time as 13:45:23 (hh:mm:ss)
\v(tmpdir)	Temporary directory
\v(ttyfd)	File descriptor of current communication device
\v(version)	Numeric version of Kermit, e.g. 501190.
\v(window)	Current window size (SET WINDOW value)

j

BUILT-IN FUNCTIONS

Builtin functions are invoked as \Fname(args), can be used in any command, and are usually used in script programs. Type SHOW FUNCTIONS for a current list.

\Fbasename(file)	basename of file
\Fbreak(s,c)	left substring of s up to first occurrence of char c
\Fcapitalize(s)	uppercase first letter of s and lowercase the rest
\Fcharacter(arg)	convert numeric arg to character
\Fchecksum(s)	32-bit arithmetic checksum of string s
\Fcode(char)	numeric code for character
\Fcontents(v)	return current definition of variable
\Fcrc16(s)	16-bit CRC of string s
\Fdate(filename)	return file's modification date/time
\Fdefinition(m)	return current definition of macro
\Feval(expr)	evaluate arithmetic expression
\Fexecute(m,a,b,..)	execute macro "m" with arguments "a", "b", etc.
\Ffiles(f)	number of files matching file spec
\Fhexify(s)	translate s into a hexadecimal string
\Findex(a1,a2,a3)	position of string a2 in a1, starting at pos a3
\Fipaddress(s)	returns first IP address from string s
\Flength(arg)	length of the string "arg"
\Fliteral(arg)	copy argument literally, no evaluation
\Flower(arg)	convert to lower case
\Flpad(text,n,c)	left pad text to length n with char c
\Fltrim(s)	Trim whitespace from left of s
\Fmax(a1,a2)	max of two numbers
\Fmin(a1,a2)	min of two numbers
\Fmodulus(n1,n2)	modulus n2 of integer n1
\Fnextfile()	next file name from list in last \Ffiles()
\Fpathname(file)	full pathname of file
\Frepeat(a1,a2)	repeat a1 a2 times
\Freplace(a1,a2,a3)	replace a2 by a3 in a1
\Freverse(arg)	reverse characters in arg
\Fright(a1,a2)	rightmost a2 characters of string a1
\Frindex(a1,a2,a3)	like \Findex, but searching from right
\Frpad(text,n,c)	right pad text to length n with char c
\Fsize(filename)	return file's length in bytes
\Fspan(a1,a2)	left substring of a1 containing only chars from a2
\Fsubstr(a1,a2,a3)	substring of a1, starts at a2, length a3
\Ftod2secs(s)	converts hh:mm:ss to seconds since midnight
\Ftrim(s)	removes trailing whitespace from s
\Funhexify(s)	converts a hexadecimal string s back to original
\Fupper(s)	converts s to upper case
\Fverify(a1,a2,n)	returns index of first char in a2 that is not in a1 starting at position n of a2.

\Feval() allows the following operators in the expression. The expression can contain variables. Only integer arithmetic is supported. Precedences are shown as numbers, 1 is highest precedence, 6 is lowest.

( )	1	parentheses
n !	2	factorial
~n	3	logical NOT
- n	4	negative
n ^ n	4	power
n * n	5	multiplication
n / n	5	division
n % n	5	modulus
n & n	5	logical AND
n + n	6	plus
n - n	6	minus
n | n	6	logical OR
n # n	6	exclusive OR
n @ n	6	greatest common divisor

COMMAND LINE OPTIONS

C-Kermit accepts commands (or "options") on the command line, in the time-honored UNIX style. Alphabetic case is significant. All options are optional. If one or more action options are included, Kermit exits immediately after executing the command-line options, otherwise it enters interactive command mode.

kermit [*filename*] [**-***x arg* [**-***x arg*]...[**-***yyy*]...]]
where:

filename is the name of a command file to execute,

-*x* is an option requiring an argument,

-*y* an option with no argument.

Actions

-s files	send files
-s -	send files from stdin
-r	receive files
-k	receive files to stdout
-x	enter server mode
-f	finish remote server
-g files	get remote files from server (quote wildcards)
-a name	alternate file name, used with -s, -r, -g
-c	connect (before file transfer), used with -l or -j
-n	connect (after file transfer), used with -l or -j

Settings

-l line	communication line device (to make a serial connection)
-l n	open file descriptor of communication device
-j host	TCP/IP network host name (to make a network connection)
-J host	connect like TELNET, exit when connection closes
-l n	open file descriptor of TCP/IP connection (n = number)
-X	X.25 network address
-Z	open file descriptor of X.25 connection
-o n	X.25 closed user group call info
-u	X.25 reverse-charge call
-q	quiet during file transfer
-8	8-bit clean
-i	transfer files in binary mode
-T	transfer files in text mode
-b bps	serial line speed, e.g. 1200
-m name	modem type, e.g. hayes
-p x	parity, x = e,o,m,s, or n
-t	half duplex, xon handshake
-e n	receive packet-length
-v n	window size
-Q	Quick file-transfer settings
-w	write over files of same name, do not backup old file
-D n	delay n seconds before sending a file

j

Other

-y name	alternate init file name
-Y	Skip init file
-R	Advise C-Kermit it will be used only in remote mode
-d	log debug info to file debug.log
-S	Stay, do not exit, after action command
-C "cmds"	Interactive-mode commands, comma-separated
-z	Force foreground operation
-B	Force background (batch) operation
-h	print command-line option help screen
=	Ignore all text that follows

Examples

Remote-mode example (C-Kermit is on the far end): send file **oofa.bin** in binary mode (**-i**) using a window size of 4 (**-v 4**):

```
kermit -v 4 -i -s oofa.bin
```

Local-mode example (C-Kermit makes the connection): make a 19200-bps direct connection out through **/dev/tty0p0**, CONNECT (**-c**) so you can log in and, presumably start a remote Kermit program and tell it to send a file, RECEIVE the file (**-r**), then CONNECT back (**-n**) so you can finish up and log out:

```
kermit -l /dev/tty0p0 -b 19200 -c -r -n
```

For dialing out, you must specify a modem type, and you might have to use a different device name:

```
kermit -m hayes -l /dev/cul0p0 -b 2400 -c -r -n
```

FILES

$HOME/.mykermrc	Your personal C-Kermit customization file.
$HOME/.kdd	Your personal dialing directory.
$HOME/.ksd	Your personal services directory.
/usr/share/lib/kermit/READ.ME	Overview of HP-UX C-Kermit, please read
/usr/share/lib/kermit/ckermit.ini	System-wide initialization file
/usr/share/lib/kermit/ckermod.ini	Sample customization file
/usr/share/lib/kermit/ckermit.kdd	Sample dialing directory
/usr/share/lib/kermit/ckermit.ksd	Sample services directory
/usr/share/lib/kermit/ckcker.upd	Supplement to "Using C-Kermit"
/usr/share/lib/kermit/ckcker.bwr	C-Kermit "beware" file - hints & tips
/usr/share/lib/kermit/ckuker.bwr	UNIX-specific beware file
/usr/share/lib/kermit/ckedemo.ini	Macros from "Using C-Kermit"
/usr/share/lib/kermit/ckevt.ini	Macros from "Using C-Kermit"
/usr/share/lib/kermit/ckepager.ksc	Alpha pager script
/var/spool/locks/LCK..*	UUCP lockfiles

To make **personalized customizations**, copy the file /usr/share/lib/kermit/ckermod.ini file to your home directory, make any desired changes, and rename it to **.mykermrc**.

You may also create a personalized **dialing directory** like the sample one in /usr/share/lib/kermit/ckermit.kdd. Your personalized dialing directory should be stored in your home directory as **.kdd** and your personal network directory as **.knd**. See Chapters 5 and 6 of *Using C-Kermit* for details.

And you may also create a personalized **services directory** like the sample one in /usr/share/lib/kermit/ckermit.ksd. Your personalized services directory should be stored in your home directory as **.ksd**. See Chapter 7 of *Using C-Kermit* for instructions.

The demonstration files illustrate C-Kermit's script programming constructs; they are discussed in chapters 17-19 of the book. You can run them by typing the appropriate TAKE command at the C-Kermit> prompt, for example: "take /usr/share/lib/kermit/ckedemo.ini".

AUTHORS

Frank da Cruz, Columbia University, with contributions from hundreds of other volunteer programmers all over the world. See Acknowledgements in *Using C-Kermit*.

REFERENCES

Frank da Cruz and Christine M. Gianone,
Using C-Kermit, Second Edition, 1997, 622 pages, Digital Press / Butterworth-Heinemann, 225 Wildwood Street, Woburn, MA 01801, USA. ISBN 1-55558-164-1. (In the USA, call +1 800 366-2665 to order Digital Press books.) Also available in a German edition from Verlag Heinze Heise, Hannover.

Frank da Cruz,
Kermit, A File Transfer Protocol, Digital Press / Butterworth-Heinemann, Woburn, MA, USA (1987). ISBN 0-932376-88-6. The Kermit file transfer protocol specification.

Christine M. Gianone,
Using MS-DOS Kermit, Digital Press / Butterworth-Heinemann, Woburn, MA, USA (1992). ISBN 1-5558-082-3. Also available in a German edition from Heise, and a French edition from Heinz Schiefer & Cie, Versailles.

Kermit News,
Issues 4 (1990) and 5 (1993), Columbia University, for detailed discussions of Kermit file transfer performance.

DIAGNOSTICS

The diagnostics produced by *C-Kermit* itself are intended to be self-explanatory. In addition, every command returns a SUCCESS or FAILURE status that can be tested by IF FAILURE or IF SUCCESS. In addition, the program itself returns an exit status code of 0 upon successful operation or nonzero if any of various operations failed.

BUGS

See the comp.protocols.kermit.* newsgroups on Usenet for discussion, or the files, ckcker.bwr and ckuker.bwr, for a list of bugs, hints, tips. etc. Report bugs via e-mail to **`kermit-support@columbia.edu`**.

CONTACTS

For more information about Kermit software and documentation, visit the Kermit Web site:

`http://www.columbia.edu/kermit/`

Or write to:

The Kermit Project
Columbia University
612 West 115th Street
New York, NY 10025-7221
USA

Or send e-mail to **`kermit@columbia.edu`**. Or call +1 212 854-3703. Or fax +1 212 663-8202.

NAME

keylogin - decrypt and store secret key with keyserv

SYNOPSIS

/usr/bin/keylogin [**-r**]

DESCRIPTION

The **keylogin** command prompts for a password, and uses it to decrypt the user's secret key. The key may be found in the **/etc/publickey** file (see *publickey*(4)) or the NIS map "publickey.byname" or the NIS+ table "cred.org_dir" in the user's home domain. The sources and their lookup order are specified in the **/etc/nsswitch.conf** file (see *nsswitch.conf*(4)). Once decrypted, the user's secret key is stored by the local key server process, *keyserv*(1M). This stored key is used when issuing requests to any secure RPC services, such as NIS+. The program *keylogout*(1) can be used to delete the key stored by **keyserv**.

keylogin will fail if it cannot get the caller's key, or the password given is incorrect. For a new user or host, a new key can be added using *newkey*(1M), *nisaddcred*(1M), or *nisclient*(1M).

Options

-r Update the **/etc/.rootkey** file. This file holds the unencrypted secret key of the super-user. Only the super-user may use this option. It is used so that processes running as super-user can issue authenticated requests without requiring that the administrator explicitly run **keylogin** as super-user at system startup time (see *keyserv*(1M)). The **-r** option should be used by the administrator when the host's entry in the publickey database has changed, and the **/etc/.rootkey** file has become out-of-date with respect to the actual key pair stored in the publickey database. The permissions on the **/etc/.rootkey** file are such that it may be read and written by the super-user but by no other user on the system.

AUTHOR

keylogin was developed by Sun Microsystems, Inc.

FILES

/etc/.rootkey Super-user's secret key

SEE ALSO

chkey(1), keylogout(1), login(1), keyserv(1M), newkey(1M), nisaddcred(1M), nisclient(1M), publickey(4), nsswitch.conf(4).

NAME

keylogout - delete stored secret key with keyserv

SYNOPSIS

/usr/bin/keylogout [**-f**]

DESCRIPTION

keylogout deletes the key stored by the key server process *keyserv*(1M). Further access to the key is revoked; however, current session keys may remain valid until they expire or are refreshed.

Deleting the keys stored by **keyserv** will cause any background jobs or scheduled *at*(1) jobs that need secure RPC services to fail. Since only one copy of the key is kept on a machine, it is a bad idea to place a call to this command in your **.logout** file since it will affect other sessions on the same machine.

Options

-f Force **keylogout** to delete the secret key for the super-user. By default, **keylogout** by the super-user is disallowed because it would break all RPC services that are started by the super-user.

AUTHOR

keylogout was developed by Sun Microsystems, Inc.

SEE ALSO

at(1), chkey(1), login(1), keylogin(1), keyserv(1M), newkey(1M), publickey(4).

NAME

keysh - context-sensitive softkey shell

SYNOPSIS

keysh

DESCRIPTION

keysh is an extension of the standard Korn-shell (for a description of the basic Korn-shell functionality, see *ksh*(1)).

keysh uses hierarchical softkey menus and context-sensitive help to aid users in building command-lines, combining the power of the Korn-shell with the ease-of-use of a menu system.

And **keysh** is entirely data-driven, allowing its menus and help to be easily extended as needed.

Note that during **keysh** invocation, the environment variable **$TERM** must specify the terminal type, as defined in the *terminfo*(4) database (see ENVIRONMENT VARIABLES below).

COMMAND ENTRY

keysh continually parses the command-line and always presents the user with an appropriate set of *current choices* on the softkey labels.

The user can select these softkeys to create readable *softkey commands* on the command-line. **keysh** automatically translates these softkey commands into equivalent *HP-UX commands* prior to executing them.

Alternatively, the user can ignore the softkeys altogether in favor of entering the traditional HP-UX commands directly, as when using the Korn-shell.

During command entry, **keysh** ordinarily displays a *status-line* near the bottom of the screen. This status-line contains information such as the host name, current directory, and time and date.

Whenever the user *must* perform an action to complete the current softkey command, **keysh** temporarily displays a *prompt message* in place of the status-line. This message briefly describes the required action.

Softkey Types

keysh presents four basic softkey types:

--Help-- Selecting the **--Help--** softkey causes **keysh** to display help information associated with the next selected softkey, rather than actually performing its action.

--More-- If there are more current choices than there are softkeys, **keysh** breaks the choices into banks and displays a special **--More--** softkey along with the first bank. Selecting the **--More--** softkey causes **keysh** to display the next bank of softkeys in sequence, eventually cycling back to the first.

<*param*> *parameter* softkeys are displayed as a name enclosed between a pair of less-than and greater-than symbols. They indicate that the user-supplied text (such as a file name) should be entered into the command-line at that point, rather than actually selecting the softkey. (Actually selecting the softkey only causes **keysh** to display a hint message on the status line; the command-line remains unchanged.)

option All other softkeys are *option* softkeys that can be used to insert the corresponding command or option name into the command-line.

Softkeys can be selected from left to right.

Editing The Command-Line

keysh supports the normal Korn-shell command-line editing modes. In addition, **keysh** also recognizes the cursor movement and editing keys found on most terminals, as defined in the *terminfo*(4) database. These include:

<Clear display>	Clear the screen and command-line. If the screen is scrolled, clear only from the cursor position to the end of scrolling memory.
<Clear line>	Clear from the cursor position to the end of the command-line.
<Delete line>	Clear the entire command-line.

<Insert line>	Translate any softkey commands in the current command-line and then edit the result.
<Delete char>	Delete the character under the cursor.
<Insert char>	Toggle between insert and overwrite modes.
<Up/Down arrow>	Recall the previous/next command from the history buffer.
<Left/Right arrow>	Move the cursor left/right.
<Home up/down>	Move the cursor to the beginning/end of the command-line.
<Tab>	If no <Insert line> key is present, perform the <Insert line> function (see above). Otherwise, if no **--Help--** softkey is present, perform the **--Help--** function (also see above). Otherwise, perform the normal tab function.
<Backtab>	Move the cursor to the beginning of the previous word.
<Ctrl-L>	Redraw the lower lines of the screen and restore any necessary terminal modes.

Visible Softkey Commands

If the **visibles** configuration option is enabled (see CONFIGURATION below), **keysh** displays a list of configured softkey commands on the softkey labels whenever it is expecting a new command. This is the the top-level softkey menu.

If the user selects one of these softkey commands, **keysh** inserts its command name into the command-line then displays a sub-menu listing the command's major parameters and/or options.

The user can then (from left to right) select option softkeys and/or enter text in place of parameter softkeys. **keysh** automatically navigates the hierarchical softkey menu, always presenting the user with an appropriate set of current choices on the softkey labels.

Note that **keysh** automatically redisplays the top-level softkey menu when it detects that a command separator (such as a pipe or semi-colon) has been entered, thus allowing the user to use softkeys for subsequent commands on the command-line as well as the first.

k

Invisible Softkey Commands

If the **invisibles** configuration option is enabled (see CONFIGURATION below) and **keysh** recognizes a traditional HP-UX command being entered, it gives the user one last chance to use the softkeys by again presenting an appropriate set of current choices on the softkey labels. As with the top-level softkey menu options, the user can choose to ignore the softkeys in favor of entering the traditional HP-UX options directly.

Backup Softkeys

If the **backups** configuration option is enabled (see CONFIGURATION below), **keysh** displays the *backup softkeys* and programs the terminal function keys appropriately whenever it has no other softkeys to display (such as when a command is running). These provide the traditional static softkey control which many users may be used to.

Traditional HP-UX Commands

If the user enters a traditional HP-UX command when **keysh** is displaying its top-level softkey menu, **keysh** simply displays the backup softkeys and allows the user to proceed.

If **keysh** subsequently detects a command separator, it again redisplays the top-level softkey menu.

Softkey Command Syntax Errors

Many softkey commands present the user with a set of softkey options from which exactly one (or at least one) *must* be selected. If the user fails to do this, **keysh** treats it as a syntax error, displaying an error message and not accepting the command until the error has been corrected.

Similarly, many softkey commands require that the user enter one or more softkey parameters before the command is semantically complete. If the user fails to do this, **keysh** again treats it as a syntax error.

Softkey Command Redirections

The user can append redirection symbols (such as a less-than or greater-than symbol followed by a file name) following a softkey command. These are appended *verbatim* to the translated HP-UX command.

USING KEYSH WITH TERMINAL SESSION MANAGER

When operating under the Terminal Session Manager (see *tsm*(1)), **keysh** displays the **tsm** softkeys instead of the backup softkeys. If desired, this interaction can be overridden by setting the **$KEYTSM** environment variable (see ENVIRONMENT VARIABLES below).

When operating under **tsm**, **keysh** also automatically displays the **tsm** window number in the status-line.

CONFIGURATION

All **keysh** configuration functions are accessed through the top-level **Keysh_config** softkey command or **kc** built-in command. These functions include:

- adding, placing, and deleting softkeys,
- specifying backup softkeys,
- selecting global options,
- selecting status-line items,
- restarting keysh,
- writing configuration changes, and
- undoing other configuration changes.

Each time the user changes **keysh**'s configuration, **keysh** automatically updates the user's **$HOME/.keyshrc** file. Upon subsequent invocations, **keysh** automatically reconfigures itself as configured previously.

Adding, Placing, And Deleting Softkeys

Any of the standard softkeys (see STANDARD SOFTKEY DEFINITIONS below) can be added to the top-level softkey menu using the **kc softkey add** command. If desired, an alternate softkey label may be specified (usually in place of a cryptic HP-UX command name) using the **with_label** option.

By default, added softkeys are placed at the end of the last **--More--** bank of the top-level softkey menu. This placement can be overridden using the **and_place** option of the **kc softkey add** command or using the **kc softkey move** command.

In addition to the standard softkeys, custom softkeys can also be added from custom softkey files using the **from_user** or **from_file** options. For a description of the softkey file format, see *softkeys*(4).

Note that any time a softkey is added from a particular softkey file, all of the remaining softkeys from that file are automatically loaded for use as invisible softkey commands. All softkeys from a file can also be loaded for use as invisible softkey commands using the **kc softkey add invisibles** command.

Any of the softkeys in the top-level softkey menu can be deleted using the **kc softkey delete** command.

Specifying Backup Softkeys

Backup softkeys are typically specified in the user's **$HOME/.softkeys** file. The basic backup softkey definition line resembles:

backup softkey "<*softkey*>**" literal "**<*string*>**";**

Where <*softkey*> is the softkey label to display and <*string*> is the text string to program the terminal function key with. A maximum of eight backup softkeys can be specified.

Note that backup softkeys must be explicitly added using the **kc softkey add backups** command before **keysh** can program them.

Selecting Global Options

Various global options can be configured using the **kc option** command, including:

backups	Enable or disable the programming of the backup softkeys.
help	Enable or disable the **--Help--** softkey.
invisibles	Enable or disable the recognition of invisible softkey commands.
prompts	Enable or disable the automatic generation of prompt messages. When enabled, **keysh** displays a prompt message whenever the user *must* perform an action to complete the current softkey command. This message briefly describes the required action.

k

selectors Enable or disable the use of keyboard selectors. When enabled, **keysh** displays an upper-case selector character in each softkey label. Typing the unquoted (upper-case) character selects the softkey just as if its corresponding function key had been pressed. Quoting the selector character in any way restores its traditional meaning. Selector keys are intended to be used on terminals that do not support a sufficient number of softkeys.

translations
Enable or disable the display of HP-UX command translations.

visibles Enable or disable the presentation and recognition of visible softkey commands.

Selecting Status-Line Items

Various information items can be configured into the status-line displayed at the bottom of the screen using the **kc status_line** command, including:

host_name	The host name.
user_name	The user name.
current_dir	The current directory.
mail_status	The mail status based on the **$MAIL** environment variable (i.e., **No mail**, **You have mail**, or **You have new mail**).
date	The date.
time	The time of day.

In addition, the **$KEYSH** environment variable, if set, is always displayed first in the status-line.

Restarting Keysh

keysh can be forced to reread the **$HOME/.keyshrc** file with the **kc restart** command. This command is typically used to update a **keysh** to a new configuration specified in another window.

keysh can also be forced to remove the **$HOME/.keyshrc** file and restart from the default user configuration with the **kc restart default** command.

Writing Configuration Changes

keysh can be forced to rewrite the **$HOME/.keyshrc** file with the **kc write** command.

Undoing Other Configuration Changes

keysh can also be forced to rewrite the **$HOME/.keyshrc** file with its original contents, thus undoing all configuration changes made since **keysh** was invoked, using the **kc undo** command.

Scaling Keysh Functionalities

keysh provides a scalable set of functionalities which can be tailored to suit personal preferences.

For users who are familiar with the HP-UX command names (though not necessarily with the command options) or for users who prefer to usually have the **tsm** softkeys visible, the command **kc options visibles off** prevents **keysh** from displaying its top-level softkey menu while waiting for a command; instead, it displays the backup softkeys or **tsm** softkeys, as appropriate. (**keysh** start-up time can then be decreased significantly by editing the **$HOME/.keyshrc** file and removing the lines which add visible softkeys.)

For users who are also familiar with the HP-UX command options, the command **kc options invisibles off** prevents **keysh** from displaying softkey menus for invisible softkey commands, also.

And for users who have no need for the backup softkeys, the command **kc options backups off** prevents **keysh** from ever programming the backup softkeys.

Note that if **visibles**, **invisibles**, and **backups** are all turned off, **keysh** performs *no* softkey processing at all. **keysh** effectively transforms into a Korn-shell which displays a status-line and recognizes the cursor movement and editing keys.

EXAMPLES

To add the **od** (see *od*(1)) softkey to the end of the top-level softkey menu and label it **Octal_dump**,

```
kc softkey add od with_label Octal_dump
```

To add the *paste*(1) softkey to the beginning of the top-level softkey menu and label it **Paste**,

```
kc softkey add paste and_place as_first_softkey
```

To add the custom emacs softkey from the file **~rpt/.softkeys** to the top-level softkey menu immediately before the **ls** (see *ls*(1)) softkey,

```
kc softkey add emacs from_user rpt and_place before_softkey ls
```

To add all invisible softkeys from the file **~rpt/.softkeys**,

```
kc softkey add invisibles from_user rpt
```

To add the backup softkeys from the file **$HOME/.softkeys**,

```
kc softkey add backups
```

To delete the **Edit_file** softkey from the top-level softkey menu,

```
kc softkey delete Edit_file
```

To disable the **--Help--** softkey,

```
kc options help off
```

To configure the user name into the status-line,

```
kc status_line user_name on
```

To configure the exit-value of the last command executed into the status-line,

```
KEYSH="\${?#0}"
```

To list the ten largest files in the current directory,

```
ls long_format | Sort_lines numerically reverse_order \
    starting_at_field 5 | head
```

STANDARD SOFTKEY DEFINITIONS

Copy_files, **Move_files**, **Print_files**, **Set_file_attribs**, **Switch**.

adjust, **ar**, **bdf**, **cal**, **cancel**, **cat**, **cd**, **cdb**, **chatr**, **chgrp**, **chmod**, **chown**, **cmp**, **col**, **comm**, **cpio**, **cut**, **dd**, **df**, **diff**, **dircmp**, **disable**, **du**, **elm**, **enable**, **exit**, **find**, **fold**, **grep**, **head**, **jobs**, **kill**, **lp**, **lpstat**, **ls**, **mailx**, **make**, **man**, **mkdir**, **more**, **nm**, **nroff**, **od**, **paste**, **pg**, **pr**, **ps**, **remsh**, **rlogin**, **rm**, **rmdir**, **sdiff**, **set**, **shar**, **sort**, **tail**, **tar**, **tcio**, **tee**, **touch**, **tr**, **umask**, **uname**, **vi**, **wc**, **who**, **write**, **xd**, **xdb**.

ENVIRONMENT VARIABLES

TERM	Specifies the terminal type, as defined in the *terminfo*(4) database. This variable must be either part of **keysh**'s invocation environment or it must be set within one of the standard Korn-shell start-up files.
COLUMNS	Specifies the number of columns in the terminal screen if different than the *terminfo*(4) default.
LINES	Specifies the number of lines in the terminal screen if not the same as the *terminfo*(4) default.
PAGER	Specifies the preferred pager to be used to display help. The default is **more** (see *more*(1)).
TZ	Specifies the time-zone to be used for time and date representations on the status-line. The default is **en_US.roman8**.
KEYBEL	Specifies the character sequence sent to the terminal by **keysh** to ring the bell. The default is **^G**.
KEYENV	Specifies an alternate **keysh** configuration file. The default is **$HOME/.keyshrc**.
KEYESC	Specifies the maximum allowable delay between characters (in milliseconds) if they are to be treated as part of a terminal escape sequence. The default is 350 ms.

k

KEYKSH	If set, specifies that **keysh** should mimic the behavior of the Korn-shell as closely as possible. No softkeys or status-line are displayed. This mode is particularly useful over slow modem lines.
KEYLOC	If set, specifies that **keysh** should leave the terminal keypad in local mode while commands are being entered. This mimics the behavior of the Korn-shell.
KEYPS1	If set, specifies that **keysh** should not reset the initial values of **$PS1**, **$PS2**, and **$PS3**. Note that **$PS1** must be a constant character string in order for **keysh** to recognize it and provide subsequent softkey assistance.
KEYSH	Specifies arbitrary text to be included in the **keysh** status-line.
KEYSIM	If set, specifies that **keysh** should always simulate softkey labels and not use the built-in labels on HP terminals.
KEYTSM	If set, specifies that **keysh** should *not* use the **tsm** softkeys when **tsm** is running. In this case, the user can either use the **tsm hotkey**, the backup softkeys, or the **Switch** softkey command (see STANDARD SOFTKEY DEFINITIONS above) to switch **tsm** windows.

KSH DIFFERENCES

keysh is an extension of *ksh*(1) with the following exceptions:

Screen Updates

keysh optimizes its display output to take advantage of available terminal capabilities. Unlike the Korn-shell which often has to redraw large portions of the command-line, **keysh** can simply insert or delete characters at the appropriate screen position.

This makes **keysh** significantly faster over slow modem lines, especially if the **$KEYKSH** environment variable is set (see ENVIRONMENT VARIABLES above).

k

Emacs-Mode Editing

The new **<ESC>v** command performs the function of the vi-mode **v** command.

An initial **^N** command recalls the history line *following* the history line executed as the previous command. This provides an easy mechanism to repeat a sequence of history commands.

gmacs editing mode is not supported; **emacs** editing mode follows the GNU emacs (18.54) definition of **^T**.

The **^@** and <ESC>**n** **^K** commands are not supported.

The **M-**<*letter*> and **M-]**<*letter*> alias functions are not supported (in lieu of true softkey support).

Vi-Mode Editing

The new **o** command performs the function of the emacs-mode **^O** command.

An initial **j** command recalls the history line *following* the history line executed as the previous command. This provides an easy mechanism to repeat a sequence of history commands.

The **|** command is not supported.

The **@**<*letter*> alias function is not supported (in lieu of true softkey support).

The **u** command performs an emacs-style nested undo; **u**<space> performs a traditional vi-style undo.

WARNINGS

keysh requires that the **$TERM** environment variable be set appropriately in your **$HOME/.profile** file. It also requires that **$LINES** and **$COLUMNS** be set appropriately if running on a non-standard size terminal. Otherwise, an error message or a garbled screen display results.

keysh requires that option softkeys be selected from left to right. When editing a command-line, it is possible to back up and insert a softkey out-of-order -- resulting in a command error.

keysh initializes **$PS1**, **$PS2**, and **$PS3** and types them *read-only* — do not change them. Instead, use **$KEYSH** to display additional status information.

keysh normally maintains the **$HOME/.keyshrc** file without user intervention; however, start-up errors may occasionally occur and persist. In this case, either execute the command **kc restart default** (to remove the file and revert to the default user configuration) or execute the command **kc write** (to rewrite the file with the current configuration).

keysh assumes that HP-UX commands are not heavily aliased; otherwise unexpected command translations may occur.

keysh neglects the effects of the Korn-shell expansion mechanisms when counting command-line parameters, causing it to occasionally underestimate the true number of parameters specified. The <ESC>**`*`** emacs-mode or vi-mode editing command can often be used to pre-expand these parameters.

The <ESC>**v** emacs-mode editing command and **v** vi-mode editing command cannot be used to edit (pre-translated) softkey commands, since no subsequent command translation can occur.

Adding a large number of softkeys can cause **keysh** to overflow a 1-Mbyte Korn-shell data size limitation, causing disconcerting behavior.

keysh can only program the function keys on terminals whose *terminfo*(4) entry defines the **pfkey** capability; similarly, it can only use hardware softkey labels on terminals whose *terminfo*(4) entry defines the **pln** capability (along with specifying **lh** equal to 2).

The default value for **$KEYESC** was chosen to provide reasonable response in both local and networked environments. If keysh misinterprets quickly typed emacs-mode or vi-mode editing commands as terminal escape sequences, it may be necessary to decrease this value.

Specifying a **\n** (new-line) in the literal key sequence for a backup softkey causes undesired results on HP terminals; use a **\r** (carriage-return) instead.

keysh does not display **tsm** softkeys when simulating softkey labels.

A limited number of environment variables and arguments are exported to the pager when displaying help.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which softkeys and messages are displayed.

LC_TIME determines the format and contents of date and time strings in the status-line.

International Code Set Support

Single-byte character code sets are supported.

AUTHOR

keysh was developed by HP and AT&T.

FILES

/usr/bin/keysh	main executable
/usr/lib/keysh/builtins	**Keysh_config** softkey definition file
/usr/lib/keysh/$LANG/softkeys	standard softkey definitions file
/usr/lib/keysh/$LANG/keyshrc	default user configuration file
/usr/lib/nls/$LANG/keysh.cat	message catalog
$HOME/.keyshrc	user configuration file
$HOME/.softkeys	user softkey definitions file

SEE ALSO

ksh(1), tsm(1), softkeys(4), terminfo(4).

NAME

kill - send a signal to a process; terminate a process

SYNOPSIS

kill [**-s** *signame*] *pid* ...

kill [**-s** *signum*] *pid* ...

kill -l

Obsolescent Versions:

kill -*signame pid* ...

kill -*signum pid* ...

DESCRIPTION

The **kill** command sends a signal to each process specified by a *pid* process identifier. The default signal is **SIGTERM**, which normally terminates processes that do not trap or ignore the signal.

pid is a process identifier, an unsigned or negative integer that can be one of the following:

> **0** The number of a process.

= **0** All processes, except special system processes, whose process group ID is equal to the process group ID of the sender.

=**-1** All processes, except special system processes, if the user has appropriate privileges. Otherwise, all processes, except special system processes, whose real or effective user ID is the same as the user ID of the sending process.

<**-1** All processes, except special system processes, whose process group ID is equal to the absolute value of *pid* and whose real or effective user ID is the same as the user of the sending process.

Process numbers can be found with the **ps** command (see *ps*(1)) and with the built-in **jobs** command available in some shells.

Options

kill recognizes the following options:

-l (ell)	List all values of *signame* supported by the implementation. No signals are sent with this option. The symbolic names of the signals (without the **SIG** prefix) are written to standard output, separated by spaces and newlines.
-s *signame*	Send the specified signal name. The default is **SIGTERM**, number **15**. *signame* can be specified in upper- and/or lowercase, with or without the **SIG** prefix. These values can be obtained by using the **-l** option. The symbolic name **SIGNULL** represents signal value zero. See "Signal Names and Numbers" below.
-s *signum*	Send the specified decimal signal number. The default is **15**, **SIGTERM**. See "Signal Names and Numbers" below.
-*signame*	(Obsolescent.) Equivalent to **-s** *signame*.
-*signum*	(Obsolescent.) Equivalent to **-s** *signum*.

Signal Names and Numbers

The following table describes a few of the more common signals that can be useful from a terminal. For a complete list and a full description, see the header file **<signal.h>** and the manual entry *signal*(5).

signum	*signame*	**Name**	**Description**
0	**SIGNULL**	Null	Check access to *pid*
1	**SIGHUP**	Hangup	Terminate; can be trapped
2	**SIGINT**	Interrupt	Terminate; can be trapped
3	**SIGQUIT**	Quit	Terminate with core dump; can be trapped
9	**SIGKILL**	Kill	Forced termination; cannot be trapped
15	**SIGTERM**	Terminate	Terminate; can be trapped
24	**SIGSTOP**	Stop	Pause the process; cannot be trapped
25	**SIGTSTP**	Terminal stop	Pause the process; can be trapped
26	**SIGCONT**	Continue	Run a stopped process

SIGNULL (**0**), the null signal, invokes error checking but no signal is actually sent. This can be used to test the validity or existence of *pid*.

SIGTERM (**15**), the (default) terminate signal, can be trapped by the receiving process, allowing the receiver to execute an orderly shutdown or to ignore the signal entirely. For orderly operations, this is the perferred choice.

SIGKILL (**9**), the kill signal, forces a process to terminate immediately. Since **SIGKILL** cannot be trapped or ignored, it is useful for terminating a process that does not respond to **SIGTERM**.

The receiving process must belong to the user of the sending process, unless the user has appropriate privileges.

As a single special case, the continue signal SIGCONT can be sent to any process that is a member of the same session as the sending process.

RETURN VALUE

Upon completion, **kill** returns with one of the following values:

0 At least one matching process was found for each *pid* operand, and the specified signal was successfully processed for at least one matching process.

>0 An error occurred.

EXAMPLES

The command:

```
kill 6135
```

signals process number 6135 to terminate. This gives the process an opportunity to exit gracefully (removing temporary files, etc.).

The following equivalent commands:

```
kill -s SIGKILL 6135
kill -s KILL 6135
kill -s 9 6135
kill -SIGKILL 6135
kill -KILL 6135
kill -9 6135
```

terminate process number 6135 abruptly by sending a **SIGKILL** signal to the process. This tells the kernel to remove the process immediately.

WARNINGS

If a process hangs during some operation (such as I/O) so that it is never scheduled, it cannot die until it is allowed to run. Thus, such a process may never go away after the kill. Similarly, defunct processes (see *ps*(1)) may have already finished executing, but remain on the system until their parent reaps them (see *wait*(2)). Using **kill** to send signals to them has no effect.

Some non-HP-UX implementations provide **kill** only as a shell built-in command.

DEPENDENCIES

This manual entry describes the external command **/usr/bin/kill** and the built-in **kill** command of the POSIX shell (see *sh-posix*(1)). Other shells, such as C and Korn (see *csh*(1) and *ksh*(1) respectively), also provide **kill** as a built-in command. The syntax for and output from these built-ins may be different.

SEE ALSO

csh(1), ksh(1), ps(1), sh(1), sh-bourne(1), sh-posix(1), kill(2), wait(2), signal(5).

STANDARDS CONFORMANCE

kill: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

NAME

ksh, rksh - shell, the standard/restricted command programming language

SYNOPSIS

ksh [**-aefhikmnoprstuvx**] [**+aefhikmnoprstuvx**] [**-o** *option*] ... [**+o** *option*] ... [**-c** *string*] [*arg* ...]

rksh [**-aefhikmnoprstuvx**] [**+aefhikmnoprstuvx**] [**-o** *option*] ... [**+o** *option*] ... [**-c** *string*] [*arg* ...]

DESCRIPTION

ksh is a command programming language that executes commands read from a terminal or a file. **rksh** is a restricted version of the command interpreter **ksh**, used to set up login names and execution environments whose capabilities are more controlled than those of the standard shell. See *Invoking ksh* and *Special Commands* sections later in this entry for details about command line options and arguments, particularly the **set** command.

Definitions

metacharacter
One of the following characters:

; **&** **(** **)** **|** **<** **>** *new-line space tab*

blank	A tab or space character.
identifier	A sequence of letters, digits, or underscores starting with a letter or underscore. Identifiers are used as names for **functions** and **named parameters**.
word	A sequence of characters separated by one or more non-quoted metacharacters .
command	A sequence of characters in the syntax of the shell language. The shell reads each command and carries out the desired action, either directly or by invoking separate utilities.
special command	A command that is carried out by the shell without creating a separate process. Often called “built-in commands”. Except for documented side effects, most special commands can be implemented as separate utilities.
#	The **#** character is interpreted as the beginning of a comment. See *Quoting* below.

k

Commands

A **simple-command** is a sequence of blank-separated words that can be preceded by a parameter assignment list. (See *Environment* below). The first word specifies the name of the command to be executed. Except as specified below, the remaining words are passed as arguments to the invoked command. The command name is passed as argument 0 (see *exec*(2)). The **value** of a simple-command is its exit status if it terminates normally, or (octal) 200+**status** if it terminates abnormally (see *signal*(5) for a list of status values).

A **pipeline** is a sequence of one or more **commands** separated by **|**. The standard output of each command except the last is connected by a pipe (see *pipe*(2)) to the standard input of the next command. Each command is run as a separate process; the shell waits for the last command to terminate. The exit status of a pipeline is the exit status of the last command in the pipeline.

A **list** is a sequence of one or more pipelines separated by **;**, **&**, **&&**, or **||**, and optionally terminated by **;**, **&**, or **|&**. Of these five symbols, **;**, **&**, and **|&** have equal precedence. **&&** and **||** have a higher but also equal precedence. A semicolon (**;**) causes sequential execution of the preceding pipeline; an ampersand (**&**) causes asynchronous execution of the preceding pipeline (that is, the shell does not wait for that pipeline to finish). The symbol **|&** causes asynchronous execution of the preceding command or pipeline with a two-way pipe established to the parent shell (known as a **co-process**). The standard input and output of the spawned command can be written to and read from by the parent shell using the **-p** option of the special commands **read** and **print** described later. The symbol **&&** (**||**) causes the *list* following it to be executed only if the preceding pipeline returns a zero (non-zero) value. An arbitrary number of new-lines can appear in a *list*, instead of semicolons, to delimit commands.

A **command** is either a simple-command or one of the following. Unless otherwise stated, the value returned by a command is that of the last simple-command executed in the command.

for *identifier* [**in** *word* ...] **do** *list* **done**
: Each time **for** is executed, *identifier* is set to the next *word* taken from the **in** *word* list. If **in** *word* ... is omitted, **for** executes the **do** *list* once for each positional parameter set (see *Parameter Substitution* below). Execution ends when there are no more words in the list.

select *identifier* [**in** *word*...] **do** *list* **done**
: A **select** command prints on standard error (file descriptor 2), the set of *word*s, each preceded by a number. If **in** *word* ... is omitted, the positional parameters are used instead (see *Parameter Substitution* below). The **PS3** prompt is printed and a line is read from the standard input. If this line consists of the number of one of the listed *word*s, the value of the parameter *identifier* is set to the *word* corresponding to this number. If this line is empty, the selection list is printed again. Otherwise the value of the parameter *identifier* is set to null. The contents of the line read from standard input is saved in the parameter **REPLY**. The *list* is executed for each selection until a **break** or end-of-file (*eof*) is encountered.

case *word* **in** [[**(**] *pattern* [**|** *pattern*] **...** **)** *list* **;;**] **...** **esac**
: A **case** command executes the *list* associated with the first *pattern* that matches *word*. The form of the patterns is identical to that used for file name generation (see *File Name Generation* below).

if *list* **then** *list* [**elif** *list* **then** *list*] **...** [**else** *list*] **fi**
: The *list* following **if** is executed and, if it returns a zero exit status, the *list* following the first **then** is executed. Otherwise, the *list* following **elif** is executed and, if its value is zero, the *list* following the next **then** is executed. Failing that, the **else** *list* is executed. If no **else** *list* or **then** *list* is executed, **if** returns a zero exit status.

while *list* **do** *list* **done**
until *list* **do** *list* **done**
: A **while** command repeatedly executes the **while** *list*, and if the exit status of the last command in the list is zero, executes the **do** *list*; otherwise the loop terminates. If no commands in the **do** *list* are executed, **while** returns a zero exit status; **until** can be used in place of **while** to negate the loop termination test.

(*list* **)**
: Execute *list* in a separate environment. If two adjacent open parentheses are needed for nesting, a space must be inserted to avoid arithmetic evaluation as described below.

{ *list* **;}**
: Execute *list*, but not in a separate environment. Note that **{** is a keyword and requires a trailing blank to be recognized.

[[*expression* **]]**
: Evaluates *expression* and returns a zero exit status when *expression* is true. See *Conditional Expressions* below, for a description of *expression*. Note that **[[** and **]]** are keywords and require blanks between them and *expression*.

function *identifier* **{** *list* **;}**
identifier **()** **{** *list* **;}**
: Define a function referred to by *identifier*. The body of the function is the *list* of commands between **{** and **}** (see *Functions* below).

time *pipeline*
: *pipeline* is executed and the elapsed time, user time, and system time are printed on standard error.

The following keywords are recognized only as the first word of a command and when not quoted:

```
if then else elif fi case esac for while
until do done { } function select time [[ ]]
```

Comments

A word beginning with **#** causes that word and all subsequent characters up to a new-line to be ignored.

Aliasing

The first word of each command is replaced by the text of an **alias**, if an alias for this word has been defined. An **alias name** consists of any number of characters excluding metacharacters, quoting characters, file expansion characters, parameter and command substitution characters, and **=**. The replacement string can contain any valid shell script, including the metacharacters listed above. The first word of each command in the replaced text, other than any that are in the process of being replaced, is tested for

k

additional aliases. If the last character of the alias value is a **blank**, the word following the alias is also checked for alias substitution. Aliases can be used to redefine special built-in commands, but cannot be used to redefine the keywords listed above. Aliases can be created, listed, and exported with the **alias** command and can be removed with the **unalias** command. Exported aliases remain in effect for subshells but must be reinitialized for separate invocations of the shell (see *Invoking ksh* below).

Aliasing is performed when scripts are read, not while they are executed. Therefore, for it to take effect, **alias** must be executed before the command referring to the alias is read.

Aliases are frequently used as a shorthand for full path names. An option to the aliasing facility allows the value of the alias to be automatically set to the full path name of the corresponding command. These aliases are called **tracked aliases**. The value of a tracked alias is defined the first time the identifier is read and becomes undefined each time the **PATH** variable is reset. These aliases remain tracked so that the next reference redefines the value. Several tracked aliases are compiled into the shell. The **-h** option of the **set** command converts each command name that is an *identifier* into a tracked alias.

The following **exported aliases** are compiled into the shell but can be unset or redefined:

```
autoload='typeset -fu'
false='let 0'
functions='typeset -f'
hash='alias -t -'
history='fc -l'
integer='typeset -i'
nohup='nohup '
r='fc -e -'
stop='kill -STOP'
suspend='kill -STOP $$'
true=':'
type='whence -v'
```

Tilde Substitution

After alias substitution is performed, each word is checked to see if it begins with an unquoted **~**. If it does, the word up to a **/** is checked to see if it matches a user name in the **/etc/passwd** file. If a match is found, the **~** and the matched login name are replaced by the login directory of the matched user. This is called a tilde substitution. If no match is found, the original text is left unchanged. A **~**, alone or before a **/**, is replaced by the value of the **HOME** parameter. A **~** followed by a **+** or **-** is replaced by the value of the parameter **PWD** and **OLDPWD**, respectively. In addition, tilde substitution is attempted when the value of a parameter assignment begins with a **~**.

Command Substitution

The standard output from a command enclosed in parenthesis preceded by a dollar sign (**$(***command***)**) or a pair of back single quotes (accent grave) (**`***command***`**) can be used as part or all of a word; trailing new-lines are removed. In the second (archaic) form, the string between the quotes is processed for special quoting characters before the command is executed (see *Quoting* below). The command substitution **$(cat file)** can be replaced by the equivalent but faster **$(<file)**. Command substitution of most special commands (built-ins) that do not perform I/O redirection are carried out without creating a separate process. However, command substitution of a function creates a separate process to execute the function and all commands (built-in or otherwise) in that function.

An arithmetic expression enclosed in double parenthesis preceded by a dollar sign (**$((***expression***))**) is replaced by the value of the arithmetic expression within the double parenthesis (see *Arithmetic Evaluation* below for a description of arithmetic expressions).

Parameter Substitution

A **parameter** is an **identifier**, one or more digits, or any of the characters *****, **@**, **#**, **?**, **-**, **$**, and **!**. A **named parameter** (a parameter denoted by an identifier) has a value and zero or more attributes. Named parameters can be assigned values and attributes by using the **typeset** special command. Attributes supported by **ksh** are described later with the **typeset** special command. Exported parameters pass values and attributes to the environment.

The shell supports a limited one-dimensional array facility. An element of an array parameter is referenced by a subscript. A subscript is denoted by a **[** followed by an arithmetic expression (see *Arithmetic Evaluation* below) followed by a **]**. To assign values to an array, use **set -A** *name value ...*. The value of all subscripts must be in the range of **0** through **1023**. Arrays need not be declared. Any reference to a named parameter with a valid subscript is legal and an array is created if necessary. Referencing an array without a subscript is equivalent to referencing the first element.

The value of a named parameter can also be assigned by writing:

name=*value* [*name*=*value*] ...

If the **-i** integer attribute is set for *name*, the *value* is subject to arithmetic evaluation as described below.

Positional parameters, parameters denoted by a number, can be assigned values with the **set** special command. Parameter **$0** is set from argument zero when the shell is invoked.

The character **$** is used to introduce substitutable *parameters*.

${***parameter***}** Substitute the value of the parameter, if any. Braces are required when *parameter* is followed by a letter, digit, or underscore that should not be interpreted as part of its name or when a named parameter is subscripted. If *parameter* is one or more digits, it is a positional parameter. A positional parameter of more than one digit must be enclosed in braces. If *parameter* is ***** or **@** all the positional parameters, starting with **$1, are substituted (separated by a field separator character). If an array *identifier* with subscript ***** or **@** is used, the value for each element is substituted (separated by a field separator character). The shell reads all the characters from **${** to the matching **}** as part of the same word even if it contains braces or metacharacters.

${#*parameter***}** If *parameter* is ***** or **@**, the number of positional parameters is substituted. Otherwise, the length of the value of the *parameter* is substituted.

${#*identifier***[*]}** Substitute the number of elements in the array *identifier*.

${*parameter***:-***word***}**
If *parameter* is set and is non-null, substitute its value; otherwise substitute *word*.

${*parameter***:=***word***}**
If *parameter* is not set or is null, set it to *word*; then substitute the value of the parameter. Positional parameters cannot be assigned in this way.

${*parameter***:?***word***}**
If *parameter* is set and is non-null, substitute its value; otherwise, print *word* and exit from the shell. If *word* is omitted, a standard message is printed.

${*parameter***:+***word***}**
If *parameter* is set and is non-null, substitute *word*; otherwise substitute nothing.

${*parameter***#***pattern***}**
${*parameter***##***pattern***}**
If the shell *pattern* matches the beginning of the value of *parameter*, the value of this substitution is the value of the *parameter* with the matched portion deleted; otherwise the value of this *parameter substituted.* In the former case, the smallest matching pattern is deleted; in the latter case, the largest matching pattern is deleted.

${*parameter***%***pattern***}**
${*parameter***%%***pattern***}**
If the shell *pattern* matches the end of the value of *parameter*, the value of *parameter* with the matched part is deleted; otherwise substitute the value of *parameter*. In the former, the smallest matching pattern is deleted; in the latter, the largest matching pattern is deleted.

In the above, *word* is not evaluated unless it is used as the substituted string. Thus, in the following example, **pwd** is executed only if **d** is not set or is null:

```
echo ${d:-$(pwd)}
```

If the colon (**:**) is omitted from the above expressions, the shell only checks to determine whether or not *parameter* is set.

The following parameters are set automatically by the shell:

k

#	The number of positional parameters in decimal.
-	Options supplied to the shell on invocation or by the **set** command.
?	The decimal value returned by the last executed command.
$	The process number of this shell.
_	Initially, the value of _ is an absolute pathname of the shell or script being executed as passed in the *environment*. Subsequently it is assigned the last argument of the previous command. This parameter is not set for commands which are asynchronous. This parameter is also used to hold the name of the matching **MAIL** file when checking for mail.
!	The process number of the last background command invoked.
COLUMNS	If this variable is set, its value is used to define the width of the edit window for the shell edit modes and for printing **select** lists. In a windowed environment, if the shell detects that the window size has changed, the shell updates the value of **COLUMNS**.
ERRNO	The value of **errno** as set by the most recently failed system call. This value is system dependent and is intended for debugging purposes.
LINENO	The line number of the current line within the script or function being executed.
LINES	If this variable is set, the value is used to determine the column length for printing **select** lists. **select** lists print vertically until about two-thirds of **LINES** lines are filled. In a windowed environment, if the shell detects that the window size has changed, the shell updates the value of **LINES**.
OLDPWD	The previous working directory set by the **cd** command.
OPTARG	The value of the last option argument processed by the **getopts** special command.
OPTIND	The index of the last option argument processed by the **getopts** special command.
PPID	The process number of the parent of the shell.
PWD	The present working directory set by the **cd** command.
RANDOM	Each time this parameter is evaluated, a random integer, uniformly distributed between 0 and 32767, is generated. The sequence of random numbers can be initialized by assigning a numeric value to **RANDOM**.
REPLY	This parameter is set by the **select** statement and by the **read** special command when no arguments are supplied.
SECONDS	Each time this parameter is referenced, the number of seconds since shell invocation is returned. If this parameter is assigned a value, the value returned upon reference is the value that was assigned plus the number of seconds since the assignment.

The following parameters are used by the shell:

CDPATH	The search path for the **cd** command.
EDITOR	If the value of this variable ends in **emacs**, **gmacs**, or **vi** and the **VISUAL** variable is not set, the corresponding option is turned on (see **set** in *Special Commands* below).
ENV	If this parameter is set, parameter substitution is performed on the value to generate the path name of the script to be executed when the shell is invoked (see *Invoking ksh* below). This file is typically used for *alias* and *function* definitions.
FCEDIT	The default editor name for the **fc** command.
FPATH	The search path for function definitions. This path is searched when a function with the **-u** attribute is referenced and when a command is not found. If an executable file is found, then it is read and executed in the current environment.
IFS	Internal field separators, normally *space*, *tab*, and *new-line* that are used to separate command words resulting from command or parameter substitution, and for separating words with the special command **read**. The first character of the **IFS** parameter is used to separate arguments for the **"$*"** substitution (see *Quoting* below).
HISTFILE	If this parameter is set when the shell is invoked, its value is the path name of the file that is used to store the command history. The default value is **$HOME/.sh_history**. If the user has appropriate privileges and no **HISTFILE** is given, then no history file is used (see *Command Re-entry* below).
HISTSIZE	If this parameter is set when the shell is invoked, the number of previously entered commands accessible to this shell will be greater than or equal to this number. The default is **128**.
HOME	The default argument (home directory) for the **cd** command.
MAIL	If this parameter is set to the name of a mail file and the **MAILPATH** parameter is not set, the shell informs the user of arrival of mail in the specified file.

MAILCHECK	This variable specifies how often (in seconds) the shell checks for changes in the modification time of any of the files specified by the **MAILPATH** or **MAIL** parameters. The default value is **600** seconds. When the time has elapsed the shell checks before issuing the next prompt.
MAILPATH	A list of file names separated by colons (:). If this parameter is set, the shell informs the user of any modifications to the specified files that have occurred within the last **MAILCHECK** seconds. Each file name can be followed by a **?** and a message to be printed, in which case the message undergoes parameter and command substitution with the parameter **$_** defined as the name of the changed file. The default message is **you have mail in $_**.
PATH	The search path for commands (see *Execution* below). The user cannot change **PATH** if executing **rksh** (except in the **.profile** file).
PS1	The value of this parameter is expanded for parameter substitution, to define the primary prompt string which, by default, is **$** followed by a space character. The character **!** in the primary prompt string is replaced by the command number (see *Command Re-entry* below). To include a **!** in the prompt, use **!!**.
PS2	Secondary prompt string, by default **>** followed by a space character.
PS3	Selection prompt string used within a **select** loop, by default **#?** followed by a space character.
PS4	The value of this variable is expanded for parameter substitution and precedes each line of an execution trace. If **PS4** is unset, the execution trace prompt is **+** followed by a space character.
SHELL	The path name of the shell is kept in the environment. When invoked, the shell is restricted if the value of this variable contains an **r** in the basename.
TMOUT	If set to a value greater than zero, the shell terminates if a command is not entered within the prescribed number of seconds after issuing the **PS1** prompt.
VISUAL	Invokes the corresponding option when the value of this variable ends in *emacs*, *gmacs*, or *vi* (see **set** in *Special Commands* below).

The shell gives default values to **PATH**, **PS1**, **PS2**, **MAILCHECK**, **TMOUT**, and **IFS**. **HOME**, **SHELL**, **ENV**, and **MAIL** are never set automatically by the shell (although **HOME**, **SHELL**, and **MAIL** are set by *login*(1)).

Blank Interpretation

After parameter and command substitution, the results of substitution are scanned for field separator characters (found in **IFS**), and split into distinct arguments where such characters are found. **ksh** retains explicit null arguments (or **' '**) but removes implicit null arguments (those resulting from *parameters* that have no values).

File Name Generation

Following substitution, each command *word* is processed as a pattern for file name expansion unless the **-f** option has been **set**. The form of the patterns is the Pattern Matching Notation defined by *regexp*(5). The word is replaced with sorted file names matching the pattern. If no file name is found that matches the pattern, the word is left unchanged.

In addition to the notation described in *regexp*(5), **ksh** recognizes composite patterns made up of one or more pattern lists separated from each other with a **|**. Composite patterns can be formed with one or more of the following:

?(*pattern-list***)**	Optionally matches any one of the given patterns.
(pattern-list***)**	Matches zero or more occurrences of the given patterns.
+(*pattern-list***)**	Matches one or more occurrences of the given patterns.
@(*pattern-list***)**	Matches exactly one of the given patterns.
!(*pattern-list***)**	Matches anything, except one of the given patterns.

Quoting

Each of the *metacharacters* listed above (See *Definitions* above) has a special meaning to the shell and causes termination of a word unless quoted. A character can be **quoted** (i.e., made to stand for itself) by preceding it with a \. The pair **\***new-line* is ignored. All characters enclosed between a pair of single quote marks (**' '**), are quoted. A single quote cannot appear within single quotes. Inside double quote marks (**""**), parameter and command substitution occurs and \ quotes the characters \, **`**, **"**, and **$**. **$*** and **$@** have identical meanings when not quoted or when used as a parameter assignment value or as a file name. However, when used as a command argument, **"$*"** is equivalent to **"$1***d***$2***d*...**"**, where *d* is the first character of the **IFS** parameter, whereas **"$@"** is equivalent to **"$1" "$2"** .... Inside back single quote (accent grave) marks (**` `**) **** quotes the characters ****, **`**, and **$**. If the back single quotes occur within double quotes, **** also quotes the character **"**.

The special meaning of keywords or aliases can be removed by quoting any character of the keyword. The recognition of function names or special command names listed below cannot be altered by quoting them.

Arithmetic Evaluation

The ability to perform integer arithmetic is provided with the special command **let**. Evaluations are performed using long arithmetic. Constants take the form [*base***#**]*n*, where *base* is a decimal number between two and thirty-six representing the arithmetic base and *n* is a number in that base. If *base* is omitted, base 10 is used.

An arithmetic expression uses the same syntax, precedence, and associativity of expression of the C language. All the integral operators, other than **++**, **--**, **?:**, and **,** are supported. Variables can be referenced by name within an arithmetic expression without using the parameter substitution syntax. When a variable is referenced, its value is evaluated as an arithmetic expression.

An internal integer representation of a *variable* can be specified with the **-i** option of the **typeset** special command. Arithmetic evaluation is performed on the value of each assignment to a variable with the **-i** attribute. If you do not specify an arithmetic base, the first assignment to the variable determines the arithmetic base. This base is used when parameter substitution occurs.

Since many of the arithmetic operators require quoting, an alternative form of the **let** command is provided. For any command beginning with **((**, all characters until the matching **))** are treated as a quoted expression. More precisely, **((**...**))** is equivalent to **let "**...**"**.

k

Prompting

When used interactively, the shell prompts with the value of **PS1** before reading a command. If at any time a new-line is typed and further input is needed to complete a command, the secondary prompt (the value of **PS2**) is issued.

Conditional Expressions.

A **conditional expression** is used with the **[[** compound command to test attributes of files and to compare strings. Word splitting and file name generation are not performed on the words between **[[** and **]]**. Each expression can be constructed from one or more of the following unary or binary expressions:

-a *file*	True if *file* exists.
-b *file*	True if *file* exists and is a block special file.
-c *file*	True if *file* exists and is a character special file.
-d *file*	True if *file* exists and is a directory.
-f *file*	True if *file* exists and is an ordinary file.
-g *file*	True if *file* exists and is has its setgid bit set.
-h *file*	True if *file* exists and is a a symbolic link.
-k *file*	True if *file* exists and is has its sticky bit set.
-n *string*	True if length of *string* is non-zero.
-o *option*	True if option named *option* is on.
-p *file*	True if *file* exists and is a fifo special file or a pipe.
-r *file*	True if *file* exists and is readable by current process.
-s *file*	True if *file* exists and has size greater than zero.
-t *fildes*	True if file descriptor number *fildes* is open and associated with a terminal device.
-u *file*	True if *file* exists and is has its setuid bit set.
-w *file*	True if *file* exists and is writable by current process.
-x *file*	True if *file* exists and is executable by current process. If *file* exists and is a directory, the current process has permission to search in the directory.
-z *string*	True if length of *string* is zero.
-H *file*	True if *file* exists and is a hidden directory (see *cdf*(4)).
-L *file*	True if *file* exists and is a symbolic link.
-O *file*	True if *file* exists and is owned by the effective user ID of this process.
-G *file*	True if *file* exists and its group matches the effective group ID of this process.
-S *file*	True if *file* exists and is a socket.
file1 **-nt** *file2*	True if *file1* exists and is newer than *file2*.

file1 **-ot** *file2*	True if *file1* exists and is older than *file2*.
file1 **-ef** *file2*	True if *file1* and *file2* exist and refer to the same file.
string **=** *pattern*	True if *string* matches *pattern*.
string **!=** *pattern*	True if *string* does not match *pattern*.
string1 **<** *string2*	True if *string1* comes before *string2* based on ASCII value of their characters.
string1 **>** *string2*	True if *string1* comes after *string2* based on ASCII value of their characters.
exp1 **-eq** *exp2*	True if *exp1* is equal to *exp2*.
exp1 **-ne** *exp2*	True if *exp1* is not equal to *exp2*.
exp1 **-lt** *exp2*	True if *exp1* is less than *exp2*.
exp1 **-gt** *exp2*	True if *exp1* is greater than *exp2*.
exp1 **-le** *exp2*	True if *exp1* is less than or equal to *exp2*.
exp1 **-ge** *exp2*	True if *exp1* is greater than or equal to *exp2*.

A compound expression can be constructed from these primitives by using any of the following, listed in decreasing order of precedence.

(*expression***)**	True, if *expression* is true. Used to group expressions.
! *expression*	True if *expression* is false.
expression1 **&&** *expression2*	True, if *expression1* and *expression2* are both true.
expression1 **||** *expression2*	True, if either *expression1* or *expression2* is true.

Input/Output

Before a command is executed, its input and output can be redirected using a special notation interpreted by the shell. The following can appear anywhere in a simple-command or can precede or follow a command and are not passed on to the invoked command. Command and parameter substitution occurs before *word* or *digit* is used, except as noted below. File name generation occurs only if the pattern matches a single file and blank interpretation is not performed.

<*word*	Use file *word* as standard input (file descriptor **0**).
>*word*	Use file *word* as standard output (file descriptor **1**). If the file does not exist, it is created. If the file exists, and the **noclobber** option is on, an error occurs; otherwise, the file is truncated to zero length.
>| *word*	Sames as **>**, except that it overrides the **noclobber** option.
>>*word*	Use file *word* as standard output. If the file exists, output is appended to it (by first searching for the end-of-file); otherwise, the file is created.
<>*word*	Open file *word* for reading and writing as standard input. If the file does not exist it is created.
<<[**-**]*word*	The shell input is read up to a line that matches *word*, or to an end-of-file. No parameter substitution, command substitution, or file name generation is performed on *word*. The resulting document, called a **here-document**, becomes the standard input. If any character of *word* is quoted, no interpretation is placed upon the characters of the document. Otherwise, parameter and command substitution occurs, **\***new-line* is ignored, and \ must be used to quote the characters \, **$**, **`**, and the first character of *word*. If **-** is appended to **<<**, all leading tabs are stripped from *word* and from the document.
<&*digit*	The standard input is duplicated from file descriptor *digit* (see *dup*(2)).
>&*digit*	The standard output is duplicated to file descriptor *digit* (see *dup*(2)).
<&-	The standard input is closed.
>&-	The standard output is closed.
<&p	The input from the co-process is moved to standard input.
>&p	The output to the co-process is moved to standard output.

If one of the above is preceded by a digit, the file descriptor number cited is that specified by the digit (instead of the default **0** or **1**). For example:

```
... 2>&1
```

means file descriptor 2 is to be opened for writing as a duplicate of file descriptor 1.

Redirection order is significant because the shell evaluates redirections referencing file descriptors in terms of the currently open file associated with the specified file descriptor at the time of evaluation. For example:

... **1>***fname* **2>&1**

first assigns file descriptor 1 (standard output) to file *fname* , then assigns file descriptor 2 (standard error) to the file assigned to file descriptor 1; i.e., *fname*. On the other hand, if the order of redirection is reversed as follows:

... **2>&1 1>***fname*

file descriptor 2 is assigned to the current standard output (user terminal unless a different assignment is inherited). File descriptor 1 is then reassigned to file *fname* without changing the assignment of file descriptor 2.

The input and output of a *co-process* can be moved to a numbered file descriptor allowing other commands to write to them and read from them using the above redirection operators. If the input of the current *co-process* is moved to a numbered file descriptor, another *co-process* can be started.

If a command is followed by **&** and job control is inactive, the default standard input for the command is the empty file **/dev/null**. Otherwise, the environment for the execution of a command contains the file descriptors of the invoking shell as modified by input/output specifications.

Environment

The **environment** (see *environ*(5)) is a list of name-value pairs passed to an executed program much like a normal argument list. The names must be **identifiers** and the values are character strings. The shell interacts with the environment in several ways. When invoked, the shell scans the environment and creates a parameter for each name found, gives it the corresponding value, and marks it *export*. Executed commands inherit the environment. If the user modifies the values of these parameters or creates new ones by using the **export** or **typeset -x** commands, the values become part of the environment. The environment seen by any executed command is thus composed of any name-value pairs originally inherited by the shell whose values can be modified by the current shell, plus any additions which must be noted in **export** or **typeset -x** commands.

The environment for any *simple-command* or function can be augmented by prefixing it with one or more parameter assignments. A parameter assignment argument takes the form *identifier***=***value*. For example,

TERM=450 *cmd args*

and

(export TERM; TERM=450; *cmd args***)**

are equivalent (as far as the above execution of *cmd* is concerned except for special commands listed below that are preceded by a percent sign).

If the **-k** option is set, all parameter assignment arguments are placed in the environment, even if they occur after the command name. The following echo statement prints **a=b c**. After the **-k** option is set, the second echo statement prints only **c**:

```
echo a=b c
set -k
echo a=b c
```

This feature is intended for use with scripts written for early versions of the shell, and its use in new scripts is strongly discouraged. It is likely to disappear someday.

Functions

The **function** keyword (described in the *Commands* section above) is used to define shell functions. Shell functions are read and stored internally. Alias names are resolved when the function is read. Functions are executed like commands, with the arguments passed as positional parameters (see *Execution* below).

Functions execute in the same process as the caller except that command substitution of a function creates a new process. Functions share all files and present working directory with the caller. Traps caught by the caller are reset to their default action inside the function. If a function does not catch or specifically ignore a trap condition, the function terminates and the condition is passed on to the caller. A trap on **EXIT** set inside a function is executed after the function completes in the environment of the caller. Ordinarily, variables are shared between the calling program and the function. However, the **typeset** special command used within a function defines local variables whose scope includes the current function and all functions it calls.

The special command **return** is used to return from function calls. Errors within functions return control to the caller.

Function identifiers can be listed with the **+f** option of the **typeset** special command. Function identifiers and the associated text of the functions can be listed with the **-f** option. Functions can be undefined with the **-f** option of the **unset** special command.

Ordinarily, functions are unset when the shell executes a shell script. The **-xf** option of the **typeset** command allows a function to be exported to scripts that are executed without reinvoking the shell. Functions that must be defined across separate invocations of the shell should be placed in the **ENV** file.

Jobs

If the **monitor** option of the **set** command is turned on, an interactive shell associates a **job** with each pipeline. It keeps a table of current jobs, printed by the **jobs** command, and assigns them small integer numbers. When a job is started asynchronously with **&**, the shell prints a line resembling:

```
[1] 1234
```

indicating that job number 1 was started asynchronously and had one (top-level) process whose process ID was 1234.

If you are running a job and want to do something else, type the suspend character (usually **^Z** (Ctrl-Z)) to send a STOP signal to the current job. The shell then indicates that the job has been 'Stopped', and prints another prompt. The state of this job can be manipulated by using the **bg** command to put it in the background, running other commands (while it is stopped or running in the background), and eventually restarting or returning the job to the foreground by using the **fg** command. A **^Z** takes effect immediately and resembles an interrupt, since pending output and unread input are discarded when **^Z** is typed.

A job run in the background stops if it tries to read from the terminal. Background jobs normally are allowed to produce output, but can be disabled by giving the **stty tostop** command. If the user sets this tty option, background jobs stop when trying to produce output.

There are several ways to refer to jobs in the shell. A job can be referred to by the process ID of any process in the job or by one of the following:

%*number*	The job with the given number.
%*string*	Any job whose command line begins with *string*.
%?*string*	Any job whose command line contains *string*.
%%	Current job.
%+	Equivalent to **%%**.
%-	Previous job.

The shell learns immediately when a process changes state. It informs the user when a job is blocked and prevented from further progress, but only just before it prints a prompt.

When the monitor mode is on, each background job that completes triggers any trap set for **CHLD**.

Attempting to leave the shell while jobs are running or stopped produces the warning, **You have stopped (running) jobs**. Use the **jobs** command to identify them. An immediate attempt to exit again terminates the stopped jobs; the shell does not produce a warning the second time.

Signals

The INT and QUIT signals for an invoked command are ignored if the command is followed by **&** and the **monitor** option is off. Otherwise, signals have the values inherited by the shell from its parent, with the exception of signal 11 (but see also the **trap** command below).

Execution

Substitutions are made each time a command is executed. If the command name matches one of the *Special Commands* listed below, it is executed within the current shell process. Next, **ksh** checks the command name to determine whether it matches one of the user-defined functions. If it does, **ksh** saves the positional parameters and then sets them to the arguments of the *function* call. The positional parameter **0** is set to the function name. When the *function* completes or issues a **return**, **ksh** restores the positional parameter list and executes any trap set on **EXIT** within the function. The value of a *function* is the value of the last command executed. A function is executed in the current shell process. If a command name is not a **special command** or a user-defined **function**, **ksh** creates a process and attempts to

k

execute the command using **exec** (see *exec*(2)).

The shell parameter **PATH** defines the search path for the directory containing the command. Alternative directory names are separated by a colon (**:**). The default path is **/usr/bin:** (specifying **/usr/bin** and the current directory in that order). Note that the current directory is specified by a null path name which can appear immediately after the equals sign, between colon delimiters, or at the end of the path list. The search path is not used if the command name contains a **/**. Otherwise, each directory in the path is searched for an executable file. If the file has execute permissions but is not a directory or an executable object code file, it is assumed to be a script file, which is a file of data for an interpreter. If the first two characters of the script file are **#!**, **exec** (see *exec*(2)) expects an interpreter path name to follow. **exec** then attempts to execute the specified interpreter as a separate process to read the entire script file. If a call to **exec** fails, **/usr/bin/ksh** is spawned to interpret the script file. All non-exported aliases, functions, and named parameters are removed in this case. If the shell command file does not have read permission, or if the *setuid* and/or *setgid* bits are set on the file, the shell executes an agent to set up the permissions and execute the shell with the shell command file passed down as an open file. A parenthesized command is also executed in a sub-shell without removing non-exported quantities.

Command Re-entry

The text of the last **HISTSIZE** (default 128) commands entered from a terminal device is saved in a **history** file. The file **$HOME/.sh_history** is used if the **HISTFILE** variable is not set or writable. A shell can access the commands of all **interactive** shells that use the same named **HISTFILE**. The special command **fc** is used to list or edit a portion of this file. The portion of the file to be edited or listed can be selected by number or by giving the first character or characters of the command. A single command or range of commands can be specified. If no editor program is specified as an argument to **fc**, the value of the **FCEDIT** parameter is used. If **FCEDIT** is not defined, **/usr/bin/ed** is used. The edited command is printed and re-executed upon leaving the editor. The editor name **-** is used to skip the editing phase and to re-execute the command. In this case a substitution parameter of the form *old***=***new* can be used to modify the command before execution. For example, if **r** is aliased to **fc -e -**, typing **r bad=good c** re-executes the most recent command that starts with the letter **c** and replaces the first occurrence of the string **bad** with the string **good**.

Special Commands

The following simple-commands are executed in the shell process. They permit input/output redirection. Unless otherwise indicated, file descriptor 1 is the default output location and the exit status, when there are no syntax errors, is zero. Commands that are preceded by **%** or **%%** are treated specially in the following ways:

1. Variable assignment lists preceding the command remain in effect when the command completes.
2. I/O redirections are processed after variable assignments.
3. Errors cause a script that contains them to abort.
4. Words following a command preceded by %% that are in the format of a variable assignment are expanded with the same rules as a variable assignment. This means that tilde substitution is performed after the **=** sign and word splitting and file name generation are not performed.

% : [*arg* ...] The command only expands parameters. A zero exit code is returned.

% . *file* [*arg* ...]
Read and execute commands from *file* and return. The commands are executed in the current shell environment. The search path specified by **PATH** is used to find the directory containing *file*. If any arguments *arg* are given, they become the positional parameters. Otherwise the positional parameters are unchanged. The exit status is the exit status of the last command executed. It is not necessary that the execute permission bit be set for *file*.

%% alias [**-tx**] [*name*[**=***value*] ...]
alias with no arguments prints the list of aliases in the form *name***=***value* on standard output. An *alias* is defined for each name whose *value* is given. A trailing space in *value* causes the next word to be checked for alias substitution. The **-t** option is used to set and list tracked aliases. The value of a tracked alias is the full path name corresponding to the given *name*. The value of a tracked alias becomes undefined when the value of **PATH** is reset, but the alias remains tracked. Without the **-t** option, for each *name* in the argument list for which no *value* is given, the name and value of the alias is printed. The **-x** option is used to set or print exported aliases. An exported alias is defined across sub-shell environments. Alias returns true unless a *name* is given for which no alias has been defined.

bg [*job* ...] Puts the specified *jobs* into the background. The current job is put in the background if *job* is unspecified. See *Jobs* for a description of the format of *job*.

% break [*n*] Exit from the enclosing **for**, **while**, **until**, or **select** loop, if any. If *n* is specified, break *n* levels.

% continue [*n*]
Resume the next iteration of the enclosing **for**, **while**, **until**, or **select** loop. If *n* is specified, resume at the *n*-th enclosing loop.

cd [**-L**|**-P**][*arg*]
cd *old new* This command can take either of two forms. In the first form it changes the current directory to *arg*. If *arg* is **-** the directory is changed to the previous directory. The **-L** option (default) preserves logical naming when treating symbolic links. **cd -L ..** moves the current directory one path component closer to the root directory. The **-P** option preserves the physical path when treating symbolic links. **cd -P ..** changes the working directory to the parent directory of the current directory. The shell parameter **HOME** is the default *arg*. The parameter **PWD** is set to the current directory. The shell parameter **CDPATH** defines the search path for the directory containing *arg*. Alternative directory names are separated by a colon (**:**). If **CDPATH** is null or undefined, the default value is the current directory. Note that the current directory is specified by a null path name which can appear immediately after the equal sign or between the colon delimiters anywhere else in the path list. If *arg* begins with a **/**, the search path is not used. Otherwise, each directory in the path is searched for *arg*.

The second form of **cd** substitutes the string *new* for the string *old* in the current directory name, **PWD** and tries to change to this new directory.

The **cd** command cannot be executed by **rksh**.

echo [*arg* ...]
See *echo*(1) for usage and description.

% eval [*arg* ...]
Reads the arguments as input to the shell and executes the resulting command(s).

% exec [*arg* ...]
Parameter assignments remain in effect after the command completes. If *arg* is given, the command specified by the arguments is executed in place of this shell without creating a new process. Input/output arguments can appear and affect the current process. If no arguments are given, the effect of this command is to modify file descriptors as prescribed by the input/output redirection list. In this case, any file descriptor numbers greater than 2 opened with this mechanism are closed when invoking another program.

% exit [*n*] Causes the shell to exit with the exit status specified by *n*. If *n* is omitted, the exit status is that of the last command executed. An end-of-file also causes the shell to exit, except when a shell has the *ignoreeof* option set (see *set* below).

%% export [*name* [**=***value*] ...]
The given *names* are marked for automatic export to the *environment* of subsequently executed commands.

fc [**-e***ename*][**-nlr**] [*first* [*last*]]
fc -e - [*old***=***new*][*command*]
In the first form, a range of commands from *first* to *last* is selected from the last **HISTSIZE** commands typed at the terminal. The arguments *first* and *last* can be specified as a number or string. A given string is used to locate the most recent command. A negative number is used to offset the current command number. The **-l** option causes the commands to be listed on standard output. Otherwise, the editor program *ename* is invoked on a file containing these keyboard commands. If *ename* is not supplied, the value of the parameter **FCEDIT** (default **/usr/bin/ed**) is used as the editor. Once editing has ended, the commands (if any) are executed. If *last* is omitted, only the command specified by *first* is used. If *first* is not specified, the default is the previous command for editing and –16 for listing. The **-r** option reverses the order of the commands and the **-n** option suppresses command numbers when listing. In the latter, the *command* is re-executed after the substitution *old***=***new* is performed.

k

fg [*job* ...] Brings each *job* into the foreground in the order specified. If no *job* is specified, the current job is brought into the foreground. See *Jobs* for a description of the format of *job*.

getopts *optstring name* [*arg* ...]
Checks *arg* for legal options. If *arg* is omitted, the positional parameters are used. An option argument begins with a **+** or a **-**. An option not beginning with **+** or **-**, or the argument **--** ends the options. *optstring* contains the letters that *getopts* recognizes. If a letter is followed by a **:**, that option is expected to have an argument. The options can be separated from the argument by blanks.

getopts places the next option letter it finds inside variable *name* each time it is invoked with a **+** preceding it when *arg* begins with a **+**. The index of the next *arg* is stored in **OPTIND**. The option argument, if any, gets stored in **OPTARG**.

A leading **:** in *optstring* causes **getopts** to store the letter of an invalid option in **OPTARG**, and to set *name* to **?** for an unknown option and to **:** when a required option is missing. Otherwise, **getopts** prints an error message. The exit status is non-zero when there are no more options.

jobs [**-lnp**][*job* ...]
Lists information about each given job; or all active jobs if *job* is omitted. The **-l** option lists process ids in addition to the normal information. The **-n** option only displays jobs that have stopped or exited since last notified. The **-p** option causes only the process group to be listed. See *Jobs* for a description of the format of *job*.

kill [**-***sig*] *process* ...
Sends either the TERM (terminate) signal or the specified signal to the specified jobs or processes. Signals are given either by number or name (as given in *signal*(5), stripped of the prefix **SIG**). The signal names are listed by **kill -l**. No default exists; merely typing **kill** does not affect the current job. If the signal being sent is **TERM** (terminate) or **HUP** (hangup), the job or process is sent a **CONT** (continue) signal when stopped. The *process* argument can be either a process ID or job. If the first argument to **kill** is a negative integer, it is interpreted as a *sig* argument and not as a process group.

let *arg* ... Each *arg* is a separate **arithmetic expression** to be evaluated. See *Arithmetic Evaluation* above, for a description of arithmetic expression evaluation. The exit status is 0 if the value of the last expression is non-zero, and 1 otherwise.

% newgrp [*arg* ...]
Equivalent to **exec newgrp** *arg* ....

print[**-Rnprsu**[*n*]] [*arg* ...]
The shell output mechanism. With no options or with option **-** or **--** the arguments are printed on standard output as described by *echo*(1). Raw mode, **-R** or **-r**, ignores the escape conventions of *echo*. The **-R** option prints all subsequent arguments and options other than **-n**. The **-p** option causes the arguments to be written onto the pipe of the process spawned with **|&** instead of standard output. The **-s** option causes the arguments to be written onto the history file instead of standard output. The **-u** option can be used to specify a one-digit file descriptor unit number *n* on which the output is to be placed. The default is 1. If the option **-n** is used, no new-line character is added to the output.

pwd [**-L**|**-P**]
With no arguments prints the current working directory (equivalent to **print -r - $PWD**). The **-L** option (default) preserves the logical meaning of the current directory and **-P** preserves the physical meaning of the current directory if it is a symbolic link (see **cd** and *ln*(1)).

read [**-prsu**[*n*]] [*name*] [**?***prompt*] [*name* ...]
The shell input mechanism. One line is read and broken up into words using the characters in **IFS** as separators. In **-r** raw mode, **** at the end of a line does not signify line continuation. The first word is assigned to the first *name*, the second word to the second *name*, etc., with remaining words assigned to the last *name*. The **-p** option causes the input line to be taken from the input pipe of a process spawned by the shell using **|&**. If the **-s** option is present, the input is saved as a command in the history file. The option **-u** can be used to specify a one-digit file descriptor unit to read from. The file descriptor can be opened with the **exec** special command. The default value of *n* is **0**. If *name* is omitted, **REPLY** is used as the default *name*. The return code is **0**, unless an end-of-file is

k

encountered. An end-of-file with the **-p** option causes cleanup for this process so that another process can be spawned. If the first argument contains a **?**, the remainder of this word is used as a **prompt** when the shell is interactive. If the given file descriptor is open for writing and is a terminal device, the prompt is placed on this unit. Otherwise the prompt is issued on file descriptor 2. The return code is **0**, unless an end-of-file is encountered.

%% readonly [*name*[**=***value*] ...]
The given *names* are marked read-only and these names cannot be changed by subsequent assignment.

% return [*n*]
Causes a shell **function** to return to the invoking script with the return status specified by *n*. If *n* is omitted, the return status is that of the last command executed. Only the low 8 bits of *n* are passed back to the caller. If **return** is invoked while not in a **function** or executing a script by the **.** (dot) built-in command, it has the same effect as an **exit** command.

set [±**aefhkmnopstuvx** **|** ±**o** *option*] **...** [±**A** *name*][*arg* **...**]
The following options are used for this command:

-A	Array assignment. Unset the variable *name* and assign values sequentially from the list *arg*. If **+A** is used, the variable *name* is not unset first.
-a	All subsequent defined parameters are automatically exported.
-e	If the shell is non-interactive and if a command fails, execute the **ERR** trap, if set, and exit immediately. This mode is disabled while reading profiles.
-f	Disables file name generation.
-h	Each command whose name is an **identifier** becomes a tracked alias when first encountered.
-k	All parameter assignment arguments (not just those that precede the command name) are placed in the environment for a command.
-m	Background jobs are run in a separate process group and a line is printed upon completion. The exit status of background jobs is reported in a completion message. This option is turned on automatically for interactive shells.
-n	Read commands and check them for syntax errors, but do not execute them. The **-n** option is ignored for interactive shells.
-o	The **-o** argument takes any of several *option* names, but only one *option* can be specified with each **-o** option. If none is supplied, the current option settings are printed. The **-o** argument *option* names follow:

allexport	Same as **-a**.
bgnice	All background jobs are run at a lower priority.
errexit	Same as **-e**.
emacs	Activates an **emacs-**style in-line editor for command entry.
gmacs	Activates a **gmacs-**style in-line editor for command entry.
ignoreeof	The shell does not exit on end-of-file. The command **exit** must be used.
keyword	Same as **-k**.
markdirs	All directory names resulting from file name generation have a trailing **/** appended.
monitor	Same as **-m**.
noclobber	Prevents redirection **>** from truncating existing files. Requires **>|** to truncate a file when turned on.
noexec	Same as **-n**.
noglob	Same as **-f**.
nolog	Do not save function definitions in history file.
nounset	Same as **-u**.
privileged	Same as **-p**.
verbose	Same as **-v**.
trackall	Same as **-h**.
vi	Activates the insert mode of a **vi-**style in-line editor until you press the ESC key which puts you in move mode. A return sends the line.

viraw	Each character is processed as it is typed in **vi** mode.
xtrace	Same as **-x**.

-p Disables processing of the **$HOME/.profile** file and uses the file **/etc/suid_profile** instead of the **ENV** file. This mode is on whenever the effective uid (gid) is not equal to the real uid (gid). Turning this off causes the effective uid and gid to be set to the real uid and gid.

-s Sort the positional parameters.

-t Exit after reading and executing one command.

-u Treat unset parameters as an error when substituting.

-v Print shell input lines as they are read.

-x Print commands and their arguments as they are executed.

- Turns off **-x** and **-v** options and stops examining arguments for options.

- - Do not change any of the options; useful in setting **$1** to a value beginning with **-**. If no arguments follow this option, the positional parameters are unset.

Using **+** instead of **-** before a option causes the option to be turned off. These options can also be used when invoking the shell. The current set of options can be examined by using **$-**.

Unless **-A** is specified, the remaining *arg* arguments are positional parameters and are assigned consecutively to **$1**, **$2**, ... If neither arguments nor options are given, the values of all names are printed on the standard output.

% shift [*n*] The positional parameters from **$***n***+1** ... are renamed **$1** ... ; default *n* is 1. The parameter *n* can be any arithmetic expression that evaluates to a non-negative number less than or equal to **$#**.

k

test [*expr*] Evaluate conditional expression *expr*. See *test*(1) for usage and description. The arithmetic comparison operators are not restricted to integers. They allow any arithmetic expression. Four additional primitive expressions are allowed:

-L *file*	True if *file* is a symbolic link.
file1 **-nt** *file2*	True if *file1* is newer than *file2*.
file1 **-ot** *file2*	True if *file1* is older than *file2*.
file1 **-ef** *file2*	True if *file1* has the same device and i-node number as *file2*.

% times Print the accumulated user and system times for the shell and for processes run from the shell.

% trap [*arg*] [*sig* ...]
arg is a command read and executed when the shell receives signal(s) *sig*. (Note that *arg* is scanned once when the trap is set and once when the trap is taken.) Each *sig* can be given as a number or name of the signal. Trap commands are executed in signal number order. Any attempt to set a trap on a signal that was ignored upon entering the current shell has no effect. If *arg* is omitted or is **-**, all traps for *sig* are reset to their original values. If *arg* is the null string, this signal is ignored by the shell and by the commands it invokes. If *sig* is **DEBUG**, *arg* is executed after each command. If *sig* is **ERR**, *arg* is executed whenever a command has a non-zero exit code. If *sig* is **0** or **EXIT** and the **trap** statement is executed inside the body of a function, the command *arg* is executed after the function completes. If *sig* is **0** or **EXIT** for a **trap** set outside any function, the command *arg* is executed on exit from the shell. The **trap** command with no arguments prints a list of commands associated with each signal number.

%% typeset [±**LRZfilrtux**[*n*]] [*name*[= *value*]] ...
Parameter assignments remain in effect after the command completes. When invoked inside a function, a new instance of the parameter *name* is created. The parameter value and type are restored when the function completes. The following list of attributes can be specified:

-L Left justify and remove leading blanks from *value*. If *n* is non-zero, it defines the width of the field. Otherwise, it is determined by the width of the value of first assignment. When the *name* is assigned, the value is filled on the right with blanks or truncated, if necessary, to fit into the field. Leading zeros are removed if the **-Z** option is also set. The **-R** option is turned off.

-R Right justify and fill with leading blanks. If *n* is non-zero, it defines the width of the field. Otherwise, it is determined by the width of the value of first assignment. The field is left-filled with blanks or truncated from the end if the parameter is reassigned. The **-L** option is turned off.

-Z Right justify and fill with leading zeros if the first non-blank character is a digit and the **-L** option has not been set. If *n* is non-zero, it defines the width of the field. Otherwise, it is determined by the width of the value of first assignment.

-f Cause *name* to refer to function names rather than parameter names. No assignments can be made to the *name* declared with the **typeset** statement. The only other valid options are **-t** (which turns on execution tracing for this function) and **-x** (which allows the function to remain in effect across shell procedures executed in the same process environment).

-i Parameter is an integer. This makes arithmetic faster. If *n* is non-zero, it defines the output arithmetic base; otherwise the first assignment determines the output base.

-l Convert all uppercase characters to lowercase. The uppercase **-u** option is turned off.

-r Any given *name* is marked "read only" and cannot be changed by subsequent assignment.

-t Tag the named parameters. Tags are user definable and have no special meaning to the shell.

-u Convert all lowercase characters to uppercase characters. The lowercase **-l** option is turned off.

-x Mark any given *name* for automatic export to the environment of subsequently executed commands.

Using **+** instead of **-** causes these options to be turned off. If no *name* arguments are given but options are specified, a list of names (and optionally the values) of the parameters that have these options set is printed. Using **+** instead of **-** retains the values to be printed. If neither names nor options are given, the names and attributes of all parameters are printed.

ulimit [*n*] If *n* is given, impose a size limit of *n* 512 byte blocks on files written by child processes (files of any size can be read). If *n* is not given, the current limit is printed.

umask [*mask*]
The user file-creation mask is set to *mask* (see *umask*(2)). *mask* can either be an octal number or a symbolic value as described in *chmod*(1). If a symbolic value is given, the new umask value is the complement of the result of applying *mask* to the complement of the previous umask value. If *mask* is omitted, the current value of the mask is printed.

unalias *name* ...
The parameters given by the list of *name*s are removed from the *alias* list.

unset [**-f**] *name* ...
The parameters given by the list of *name*s are unassigned; that is, their values and attributes are erased. Read-only variables cannot be unset. If the **-f** option is set, *name*s refer to function names. Unsetting **ERRNO**, **LINENO**, **MAILCHECK**, **OPTARG**, **OPTIND**, **RANDOM**, **SECONDS**, **TMOUT**, and _ removes their special meaning even if they are subsequently assigned to.

% wait [*job*] Wait for the specified *job* to terminate or stop, and report its status. This status becomes the return code for the **wait** command. If *job* is not given, **wait** waits for all currently active child processes to terminate or stop. The termination status returned is that of the last process. See *Jobs* for a description of the format of a *job*.

whence [**-pv**] *name* ...
For each *name*, indicate how it would be interpreted if used as a command name. The **-v** option produces a more verbose report. The **-p** option does a path search for *name* even if *name* is an alias, a function, or a reserved word.

Invoking ksh

If the shell is invoked by **exec** (see *exec*(2)), and the first character of argument zero (**$0**) is **-**, the shell is assumed to be a login shell and commands are read first from **/etc/profile**. The expression **${HOME:-.}/.profile** is then evaluated and an attempt to open the resulting filename is made. If the file is opened successfully, the file is read. Next, commands are read from the file named by performing parameter substitution on the value of the environment parameter **ENV**, if the file exists. If the **-s** option is not present and *arg* is, a path search is performed on the first *arg* to determine the name of the script to execute. When running **ksh** with *arg*, the script *arg* must have read permission and any *setuid* and *getgid* settings are ignored. Commands are then read as described below. The following options are interpreted by the shell when it is invoked:

-c *string*	If the **-c** option is present, commands are read from *string*.
-s	If the **-s** option is present or if no arguments remain, commands are read from the standard input. Shell output, except for the output of some of the *Special Commands* listed above, is written to file descriptor 2.
-i	If the **-i** option is present or if the shell input and output are attached to a terminal (as reported by *tty*(3C)), the shell is interactive. In this case SIGTERM is ignored (so that **kill 0** does not kill an interactive shell) and SIGINT +1 is caught and ignored (so that **wait** is interruptible). In all cases, SIGQUIT is ignored by the shell. (See *signal*(5).)
-r	If the **-r** option is present, the shell is a restricted shell.

The remaining options and arguments are described under the **set** command above.

rksh Only

rksh is used to set up login names and execution environments where capabilities are more controlled than those of the standard shell. The actions of **rksh** are identical to those of **ksh**, except that the following are forbidden:

- Changing directory (see *cd*(1))
- Setting the value of **SHELL**, **ENV**, or **PATH**
- Specifying path or command names containing **/**
- Redirecting output (**>**, **>|**, **<>**, and **>>**)

k

The restrictions above are enforced after the **.profile** and **ENV** files are interpreted.

When a command to be executed is found to be a shell procedure, **rksh** invokes **ksh** to execute it. Thus, the end-user is provided with shell procedures accessible to the full power of the standard shell, while being restricted to a limited menu of commands. This scheme assumes that the end-user does not have write and execute permissions in the same directory.

When a shell procedure is invoked from **rksh**, the shell interpreter specified with the **#!** magic inherits all the restricted features of **rksh**. So, the shell procedures written for execution under **rksh** with the intent of utilizing the full power of the standard shell should not specify an interpreter with **#!**.

These rules effectively give the writer of the **.profile** file complete control over user actions, by performing guaranteed set-up actions and leaving the user in an appropriate directory (probably not the login directory).

The system administrator often sets up a directory of commands (usually **/usr/rbin**) that can be safely invoked by **rksh**. HP-UX systems provide a restricted editor **red** (see *ed*(1)), suitable for restricted users.

COMMAND-LINE EDITING

In-line Editing Options

Normally, each command line typed at a terminal device is followed by a new-line (carriage-return or line-feed). If either the **emacs**, **gmacs**, or **vi** option is set, the user can edit the command line. An editing option is automatically selected each time the **VISUAL** or **EDITOR** variable is assigned a value ending in either of these option names.

The editing features require that the user's terminal accept Return as carriage return without line feed and that a space character must overwrite the current character on the screen. ADM terminal users should set the "space/advance" switch to "space". Hewlett-Packard terminal users should set the straps to "bcGHxZ etX".

The editing modes enable the user to look through a window at the current line. The default window width is 80, unless the value of **COLUMNS** is defined. If the line is longer than the window width minus two, a mark displayed at the end of the window notifies the user. The mark is a **>**, **<**, or ***** if the line extends respectively on the right, left, or both side(s) of the window. As the cursor moves and reaches the window boundaries, the window is centered about the cursor.

The search commands in each edit mode provide access to the history file. Only strings are matched, not patterns, although a leading **^** in the string restricts the match to begin at the first character in the line.

Emacs Editing Mode

This mode is invoked by either the **emacs** or **gmacs** option. Their sole difference is their handling of **^T**. To edit, the user moves the cursor to the point needing correction and inserts or deletes characters or words. All editing commands are control characters or escape sequences. The notation for control characters is circumflex (**^**) followed by the character. For example, **^F** is the notation for Ctrl-**F**. This is entered by pressing the **f** key while holding down the Ctrl (control) key. The Shift key is *not* pressed. (The notation **^?** indicates the DEL (delete) key.)

The notation for escape sequences is *M*- followed by a character. For example, *M*-**f** (pronounced Meta f) is entered by depressing ESC (ASCII 033) followed by **f**. *M*-**F** would be the notation for ESC followed by Shift (capital) **F**.

All edit commands operate from any place on the line (not only at the beginning). Neither the Return nor the Line Feed key is entered after edit commands, except when noted.

^F	Move cursor forward (right) one character.
M-**f**	Move cursor forward one word. (The editor's idea of a word is a string of characters consisting of only letters, digits and underscores.)
^B	Move cursor backward (left) one character.
M-**b**	Move cursor backward one word.
^A	Move cursor to start of line.
^E	Move cursor to end of line.
^]*char*	Move cursor forward to character *char* on current line.
M-**^]***char*	Move cursor backward to character *char* on current line.
^X^X	Interchange the cursor and mark.
erase	(User defined erase character as defined by the *stty*(1) command, usually **^H** or **#**.) Delete previous character.
^D	Delete current character.
eof	End-of-file character, normally **^D**, terminates the shell if the current line is null.
M-**d**	Delete current word.
M-**^H**	(Meta-backspace) Delete previous word.
M-**h**	Delete previous word.
M-**^?**	(Meta-DEL) Delete previous word (if interrupt character is **^?** (DEL, the default) this command does not work).
^T	Transpose current character with next character in **emacs** mode. Transpose two previous characters in **gmacs** mode.
^C	Capitalize current character.
M-**c**	Capitalize current word.
M-**l**	Change the current word to lowercase.
^K	Delete from the cursor to the end of the line. If preceded by a numerical parameter whose value is less that the current cursor position, delete from the given position up to the cursor. If preceded by a numerical parameter whose value is greater than the current cursor position, from the cursor up to the given position.
^W	Kill from the cursor to the mark.
M-**p**	Push the region from the cursor to the mark on the stack.
kill	(User-defined kill character, as defined by the *stty*(1) command, usually **^G** or **@**.) Kill the entire current line. If two *kill* characters are entered in succession, all subsequent consecutive kill characters cause a line feed (useful when using paper terminals).
^Y	Restore last item removed from line (yank item back to the line).
^L	Line feed and print current line.
^@	(Null character) Set mark.
M-space	(Meta space) Set mark.
^J	(New line) Execute the current line.
^M	(Return) Execute the current line.
^P	Fetch previous command. Each time **^P** is entered, the next previous command in the history list is accessed.
^N	Fetch next command. Each time **^N** is entered the next command in the history list is accessed.
M-**<**	Fetch the least recent (oldest) history line.
M-**>**	Fetch the most recent (youngest) history line.
^R*string*	Reverse search history for a previous command line containing *string*. If a parameter of zero is given, the search is forward. *string* is terminated by a Return or New-Line. If *string* is preceded by a **^**, the matched line must begin with *string*. If *string* is omitted, the next command line containing the most recent *string* is accessed. In this case a parameter

	of zero reverses the direction of the search.
^O	Operate - Execute the current line and fetch from the history file the next line relative to current line.
M-digits	(Escape) Define numeric parameter, the digits are taken as a parameter to the next command. The commands that accept a parameter are **^F**, **^B**, *erase*, **^C**, **^D**, **^K**, **^R**, **^P**, **^N**, **^]**, *M-***.**, *M-*_, *M-***b**, *M-***c**, *M-***d**, *M-***f**, *M-***h**, *M-***l** and *M-***^H**.
M-letter	Softkey. User's alias list is searched for an alias by the name *_letter* and if an alias of this name is defined, its value is inserted on the input queue. This *letter* must not be one of the above meta-functions.
*M-***.**	The last word of the previous command is inserted on the line. If preceded by a numeric parameter, the value of this parameter determines which word to insert rather than the last word.
*M-*_	Same as *M-***..**
*M-******	Attempt file-name generation on the current word.
*M-*ESC	File-name completion. Replaces the current word with the longest common prefix of all filenames matching the current word with an asterisk appended. If the match is unique, a **/** is appended if the file is a directory and a space is appended if the file is not a directory.
*M-***=**	List files matching current word pattern as if an asterisk were appended.
^U	Multiply parameter of next command by 4.
\	Escape next character. Editing characters, the user's erase, kill and interrupt (normally **^?**) characters can be entered in a command line or in a search string if preceded by a \. The \ removes the next character's editing features (if any).
^V	Display version of the shell.
M-#	Insert a **#** at the beginning of the line and execute it. This causes a comment to be inserted in the history file.

k

Vi Editing Mode

There are two typing modes. Entering a command puts you into **input** mode. To edit, the user enters **control** mode by pressing ESC and moves the cursor to the point needing correction, then inserts or deletes characters or words. Most control commands accept an optional repeat *count* prior to the command.

In vi mode on most systems, canonical processing is initially enabled and the command is echoed again if the speed is 1200 baud or greater and contains any control characters, or if less than one second has elapsed since the prompt was printed. The ESC character terminates canonical processing for the remainder of the command and the user can then modify the command line. This scheme has the advantages of canonical processing with the type-ahead echoing of raw mode.

Setting the **viraw** option always disables canonical processing on the terminal. This mode is implicit for systems that do not support two alternate end-of-line delimiters, and can be helpful for certain terminals.

Input Edit Commands

By default the editor is in input mode.

erase	Delete previous character. (*erase* is a user-defined erase character, as defined by the *stty*(1) command, usually **^H** or **#**.)
^W	Delete the previous blank separated word.
^D	Terminate the shell.
^V	Escape next character. Editing characters, erase or kill characters can be entered in a command line or in a search string if preceded by a **^V**. **^V** removes the next character's editing features (if any).
\	Escape the next *erase* or *kill* character.

Motion Edit Commands

These commands move the cursor. The designation [*count*] causes a repetition of the command the cited number of times.

[*count*]**l**	Cursor forward (right) one character.
[*count*]**w**	Cursor forward one alphanumeric word.
[*count*]**W**	Cursor to the beginning of the next word that follows a blank.
[*count*]**e**	Cursor to end of word.
[*count*]**E**	Cursor to end of the current blank-delimited word.
[*count*]**h**	Cursor backward (left) one character.
[*count*]**b**	Cursor backward one word.

[*count*]**B**	Cursor to preceding blank separated word.
[*count*]**|**	Cursor to column *count*. Default is 1.
[*count*]**f***c*	Find the next character *c* in the current line.
[*count*]**F***c*	Find the previous character *c* in the current line.
[*count*]**t***c*	Equivalent to **f** followed by **h**.
[*count*]**T***c*	Equivalent to **F** followed by **l**.
[*count*]**;**	Repeats the last single character find command, **f**, **F**, **t**, or **T**.
[*count*]**,**	Reverses the last single character find command.
0	Cursor to start of line.
^	Cursor to first nonblank character in line.
$	Cursor to end of line.

Search Edit Commands

These commands access your command history.

[*count*]**k**	Fetch previous command. Each time **k** is pressed, the next earlier command in the history list is accessed.
[*count*]**-**	Equivalent to **k**.
[*count*]**j**	Fetch next command. Each time **j** is entered, the next later command in the history list is accessed.
[*count*]**+**	Equivalent to **j**.
[*count*]**G**	The command number *count* is fetched. The default is the first command in the history list.
/*string*	Search backward through history for a previous command containing *string*. *string* is terminated by a "Return" or "New-line". If *string* is preceded by a **^**, the matched line must begin with *string*. If *string* is null, the previous string is used.
?*string*	Same as **/** but search in the forward direction.
n	Search for next match of the last pattern to **/** or **?** commands.
N	Search for next match of the last pattern to **/** or **?**, but in reverse direction. Search history for the *string* entered by the previous **/** command.

Text Modification Edit Commands

These commands modify the line.

a	Enter input mode and enter text after the current character.
A	Append text to the end of the line. Equivalent to **$a**.
[*count*]**c***motion*	
c[*count*]*motion*	Move cursor to the character position specified by *motion*, deleting all characters between the original cursor position and new position, and enter input mode. If *motion* is **c**, the entire line is deleted and input mode entered.
C	Delete the current character through the end of line and enter input mode. Equivalent to **c$**.
S	Equivalent to **cc**.
D	Delete the current character through end of line. Equivalent to **d$**.
[*count*]**d***motion*	
d[*count*]*motion*	Move cursor to the character position specified by *motion*, deleting all characters between the original cursor position and new position. If *motion* is **d**, the entire line is deleted.
i	Enter input mode and insert text before the current character.
I	Insert text before the beginning of the line. Equivalent to the two-character sequence **0i**.
[*count*]**P**	Place the previous text modification before the cursor.
[*count*]**p**	Place the previous text modification after the cursor.
R	Enter input mode and replace characters on the screen with characters you type in overlay fashion.
[*count*]**r***c*	Replace the current character with *c*.
[*count*]**x**	Delete current character.
[*count*]**X**	Delete preceding character.
[*count*]**.**	Repeat the previous text modification command.
[*count*]**~**	Invert the case of the current character and advance the cursor.
[*count*]**_**	Causes the *count* word of the previous command to be appended at the current cursor location and places the editor in input mode at the end of the appended text. The last word is used if *count* is omitted.
*****	Appends an ***** to the current word and attempts file name generation. If no match is found, the bell rings. If a match is found, the word is replaced by the matching string and the command places the editor in input mode.

ESC	
\	Attempt file name completion on the current word. Replaces the current word with the longest common prefix of all filenames matching the current word with an asterisk appended. If the match is unique, a **/** is appended if the file is a directory and a space is appended if the file is not a directory.

Other Edit Commands

[*count*]**y***motion*	
y[*count*]*motion*	Yank current character through character that *motion* would move the cursor to and puts them into the delete buffer. The text and cursor are unchanged.
Y	Yanks from current position to end of line. Equivalent to **y$**.
u	Undo the last text modifying command.
U	Undo all the text modifying commands performed on the line.
[*count*]**v**	Returns the command **fc -e ${VISUAL:-${EDITOR:-vi}}** *count* in the input buffer. If *count* is omitted, the current line is used.
^L	Line feed and print current line. Has effect only in control mode.
^J	(New line) Execute the current line, regardless of mode.
^M	(Return) Execute the current line, regardless of mode.
#	Equivalent to **I#** followed by **Return**. Sends the line after inserting a **#** in front of the line and after each new-line. Useful for inserting the current command line in the history list without executing it.
=	List the filenames that match the current word if an asterisk were appended to it.
@*letter*	The user's alias list is searched for an alias by the name *_letter* and if an alias of this name is defined, its value is inserted on the input queue for processing.

EXTERNAL INFLUENCES

Environment Variables

LC_COLLATE determines the collating sequence used in evaluating pattern matching notation for file name generation.

LC_CTYPE determines the classification of characters as letters, and the characters matched by character class expressions in pattern matching notation.

If LC_COLLATE or LC_CTYPE is not specified in the environment or is set to the empty string, the value of LANG is used as a default for each unspecified or empty variable. If LANG is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of LANG. If any internationalization variable contains an invalid setting, **ksh** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single-byte character code sets are supported.

RETURN VALUE

Errors detected by the shell, such as syntax errors, cause the shell to return a non-zero exit status. Otherwise, the shell returns the exit status of the last command executed (also see the **exit** command above). If the shell is being used non-interactively, execution of the shell file is abandoned. Runtime errors detected by the shell are reported by printing the command or function name and the error condition. If the line number on which the error occurred is greater than one, the line number is also printed in brackets (**[ ]**) after the command or function name.

WARNINGS

File descriptors 10 and 54 through 60 are used internally by the Korn Shell. Applications using these and forking a subshell should not depend upon them surviving in the subshell or its descendants.

If a command which is a **tracked alias** is executed, and a command with the same name is installed in a directory in the search path before the directory where the original command was found, the shell continues to load and execute the original command. Use the **-t** option of the **alias** command to correct this situation.

If you move the current directory or one above it, **pwd** may not give the correct response. Use the **cd** command with a full path name to correct this situation.

Some very old shell scripts contain a caret (**^**) as a synonym for the pipe character (**|**). Note however, **ksh** does not recognize the caret as a pipe character.

If a command is piped into a shell command, all variables set in the shell command are lost when the command completes.

Using the **fc** built-in command within a compound command causes the entire command to disappear from the history file.

The built-in command **.** *file* reads the entire file before any commands are executed. Therefore, **alias** and **unalias** commands in the file do not apply to any functions defined in the file.

Traps are not processed while the shell is waiting for a foreground job. Thus, a trap on **CHLD** is not executed until the foreground job terminates.

The **export** built-in command does not handle arrays properly. Only the first element of an array is exported to the *environment*.

Background processes started from a non-interactive shell cannot be accessed by using job control commands.

In an international environment, character ordering is determined by the setting of LC_COLLATE, rather than by the binary ordering of character values in the machine collating sequence. This brings with it certain attendant dangers, particularly when using range expressions in file name generation patterns. For example, the command,

```
rm [a-z]*
```

might be expected to match all file names beginning with a lowercase alphabetic character. However, if dictionary ordering is specified by LC_COLLATE, it would also match file names beginning with an uppercase character (as well as those beginning with accented letters). Conversely, it would fail to match letters collated after **z** in languages such as Danish or Norwegian.

The correct (and safe) way to match specific character classes in an international environment is to use a pattern of the form:

```
rm [[:lower:]]*
```

This uses LC_CTYPE to determine character classes and works predictably for all supported languages and codesets. For shell scripts produced on non-internationalized systems (or without consideration for the above dangers), it is recommended that they be executed in a non-NLS environment. This requires that LANG, LC_COLLATE, etc., be set to "C" or not set at all.

Be aware that the value of the **IFS** variable in the user's environment affects the behavior of scripts.

ksh implements command substitution by creating a pipe between itself and the command. If the root file system is full, the substituted command cannot write to the pipe. As a result, the shell receives no input from the command, and the result of the substitution is null. In particular, using command substitution for variable assignment under such circumstances results in the variable being silently assigned a NULL value.

AUTHOR

ksh was developed by AT&T.

FILES

/etc/passwd	to find home directories
/etc/profile	read to set up system environment
/etc/suid_profile	security profile
$HOME/.profile	read to set up user's custom environment
/tmp/sh*	for here-documents

SEE ALSO

cat(1), cd(1), echo(1), env(1), test(1), umask(1), vi(1), dup(2), exec(2), fork(2), gtty(2), pipe(2), signal(5), umask(2), ulimit(2), wait(2), rand(3C), a.out(4), profile(4), environ(5), lang(5), regexp(5).

NAME

last, lastb - indicate last logins of users and ttys

SYNOPSIS

/usr/bin/last [**-R**] [**-***number*] [**-f** *file*] [*name* ...] [*tty* ...]

/usr/bin/lastb [**-R**] [**-***number*] [**-f** *file*] [*name* ...] [*tty* ...]

DESCRIPTION

The **last** command searches backwards through the file **/var/adm/wtmp** (which contains a record of all logins and logouts) for information about a user, a tty, or any group of users and ttys. Arguments specify names of users or ttys of interest. The names of ttys can be given fully or abbreviated. For example, **last 0** is the same as **last tty0**. If multiple arguments are given, the information that applies to any of the arguments is printed. For example, **last root console** lists all of **root**'s sessions as well as all sessions on the console terminal. The **last** command prints the sessions of the specified users and ttys, most recent first, indicating when the session began, the duration of the session, and the tty on which the session took place. **last** indicates if the session is still in progress or if it was cut short by a reboot.

The pseudo-user **reboot** logs each time the system reboots. Thus, **last reboot** is a useful command for evaluating the relative time between system reboots.

If **last** is interrupted, it indicates how far the search has progressed in **wtmp**. If interrupted by a quit signal (generated by a Ctrl-\), **last** indicates how far the search has progressed, then continues the search.

The **lastb** command searches backwards through the database file **/var/adm/btmp** to display bad login information. Access to **/var/adm/btmp** should be restricted to users with appropriate privileges (owned by and readable only by **root**) because it may contain password information.

1

Options

The **last** and **lastb** commands recognize the following options and arguments:

(none)	If no arguments are specified, **last** prints a record of all logins and logouts in reverse order, most recent first.
-R	When used with **last** and **lastb**, **-R** displays the user's host name as it is stored in the files **/var/adm/wtmp** and **/var/adm/btmp**, respectively. The host name is displayed between the tty name and the user's login time.
-*number*	Limits the report to *number* of lines.
-f *file*	Use *file* as the name of the accounting file instead of **/var/adm/wtmp** or **/var/adm/btmp**.

AUTHOR

last was developed by the University of California, Berkeley and HP.

FILES

/var/adm/btmp	Bad login database
/var/adm/wtmp	Login database

SEE ALSO

login(1), utmp(4).

NAME

lastcomm - show last commands executed in reverse order

SYNOPSIS

lastcomm [*commandname*] ... [*username*] ... [*terminalname*] ...

DESCRIPTION

lastcomm gives information on previously executed commands. If no arguments are specified, **lastcomm** prints information about all the commands recorded in the accounting file, **/var/adm/pacct** during the current accounting file's lifetime. If called with arguments, only accounting entries with a matching command name, user name, or terminal name are printed. For example, to produce a listing of all executions of commands named **a.out** by user **root** on terminal **ttyd0** use:

```
lastcomm a.out root ttyd0
```

For each process entry, the following are printed.

- Name of the user who ran the process.
- Flags, as accumulated by the accounting facilities in the system.
- Command name under which the process was called.
- Amount of cpu time used by the process (in seconds).
- What time the process started.

Flags are encoded as follows:

S	Command was executed by a user who has appropriate privileges.
F	Command ran after a fork, but without a following *exec*.
D	Command terminated with the generation of a **core** file.
X	Command was terminated with the signal SIGTERM.

1

FILES

/var/adm/pacct	current file for per-process accounting

AUTHOR

lastcomm was developed by the University of California, Berkeley.

SEE ALSO

last(1), acct(4), acctsh(1M), core(4).

NAME

ld - link editor

SYNOPSIS

The link editor. Common options:

ld [**-bdmnqrstvxzEGIOPQVZ**] [**-a** *search*] [**-c** *filename*] [**-dynamic**]
[**-e** *epsym*] [**-h** *symbol*] ... [**-l***x* | *file*] ... [**-l:** *library*]
[**-m**] [**-noshared**] [**-o** *outfile*] [**-u** *symbol*] ... [**-y** *symbol*] ...
[**-B** *bind*] [**-D** *offset*] [**-L** *dir*] ... [**-R** *offset*] [**+b** *path_list*]
[**+compat**] [**+df** *file*] [**+e** *symbol*] [**+ee** *symbol*] [**+fb**] [**+fbu**]
[**+h** *internal_name*] [**+help**] [**+k**] [**+n**] [**+O[no]fastaccess**]
[**+O[no]procelim**] [**+Oselectivepercent** *n*] [**+Oselectivesize** *size*]
[**+OselectiveO3**] [**+Ostaticprediction**] [**+pd** *size*] [**+pi** *size*]
[**+pgm** *name*] [**+s**] [**+std**] [**+v[no]shlibunsats**] [**+vallcompatwarnings**]
[**+v[no]compatwarnings**] [**+FP** *flag*] [**+I** *symbol*]

32 Bit (SOM) Link Editor options:

ld [**-NST**] [**-A** *name*] [**-C** *n*] [**+cg** *pathname*] [**+dpv**]

64 bit (ELF) Link Editor options:

ld [**-k** *filename*] [**+[no]allowunsats**] [**+[no]forceload**] [**+hideallsymbols**]
[**+nodefaultmap**] [**+noenvvar**] [**+stripunwind**] [**+vtype** *type*]

DESCRIPTION

ld takes one or more object files or libraries as input and combines them to produce a single (usually executable) file. In doing so it resolves references to external symbols, assigns final addresses to procedures and variables, revises code and data to reflect new addresses (a process called "relocation"), and updates symbolic debug information when present in the file. By default, **ld** produces an executable file that can be run by the HP-UX loader **exec()** (see *exec*(2)). Alternatively, the linker can generate a relocatable file that is suitable for further processing by **ld** (see **-r** below). It can also generate a shared library (see **-b** below). The linker marks the output file non-executable if there are any duplicate symbols or any unresolved external references remain. **ld** may or may not generate an output file (see **+k** option) if any other errors occur during its operation. **ld** recognizes three kinds of input files: object files created by the compilers, assembler, or linker (also known as **.o** files), shared libraries created by the linker, and archives of object files (called archive libraries). An archive library contains a table of all the externally-visible symbols from its component object files. (The archiver command *ar*(1) creates and maintains this index.) **ld** uses this table to resolve references to external symbols.

ld processes files in the same order as they appear on the command line. It includes code and data from an archive library element if and only if that object module provides a definition for a currently unresolved reference within the user's program. It is common practice to list libraries following the names of all simple object files on the command line.

Code and data from shared libraries is never copied into an executable program. The dynamic loader **/usr/lib/dld.sl** on 32 bit links is invoked at startup time by the startup file **crt0.o** if a program uses shared libraries. Identical copies of **crt0.o** can be found in either **/usr/ccs/lib/crt0.o** or **/opt/langtools/lib/crt0.o**. For 64 bit, the dynamic loader is **/usr/lib/pa20_64/dld.sl** and is invoked by exec for those programs that use shared libraries. **crt0.o** is not required in shared bound links. The dynamic loader attaches each required library to the process and resolves all symbolic references between the program and its libraries. The text segment of a shared library is shared among all processes that use the library; each process using the library receives its own copy of the data segment.

ld recursively examines the dependencies of shared libraries used by a program that was created by **ld**. If **ld** does not find a supporting shared library at the path recorded in the dependency list of a shared library, and if the dependency is the result of an **-l** argument used when the shared library was created, **ld** will search all the directories that it would search for a library that was specified with **-l** (see **-L** and **LPATH**).

Environment Variables

Arguments can be passed to the linker through the **LDOPTS** environment variable as well as on the command line. The linker gets the value of **LDOPTS** and places its contents before any arguments on the command line.

The **LD_PXDB** environment variable defines the full execution path for the debug preprocessor **pxdb**. The default value is **/opt/langtools/bin/pxdb**. **ld** invokes **pxdb** on its output file if that file is executable and contains debug information. To defer invocation of **pxdb** until the first debug session, set **LD_PXDB** to **/dev/null**.

The **LPATH** environment variable can be used to specify default directories to search for library files. See the **-l** option.

Options

The common **ld** options are listed first, followed by the options supported only in a 32 bit linker, and then the options only supported in a 64 bit linker.

-a *search* Specify whether shared or archive libraries are searched with the **-l** option. The value of *search* should be one of **archive**, **shared**, **archive_shared**, **shared_archive**, or **default**. This option can appear more than once, interspersed among **-l** options, to control the searching for each library. The default is to use the shared version of a library if one is available, or the archive version if not.

If either **archive** or **shared** is active, only the specified library type is accepted.

If **archive_shared** is active, the archive form is preferred, but the shared form is allowed.

If **shared_archive** is active, the shared form is preferred but the archive form is allowed.

-b Create a shared library rather than a normal executable file. Object files processed with this option must contain **position-independent code** (PIC). See the discussion of PIC in *cc*(1), *CC*(1) (part of the optional C++ compiler documentation), *f77*(1), *pc*(1), *as*(1), and *Linker and Libraries Online User Guide*.

-c *filename* Read ld options from a file. Each line contains zero or more arguments separated by white space. Each line in the file, including the last line, must end with a newline character. A **#** character implies that the rest of the line is a comment. To escape a **#** character, use the sequence **##**.

-d Force definition of "common" storage; that is, assign addresses and sizes, for **-r** output.

-e *epsym* Set the default entry point address for the output file to be that of the symbol *epsym*. (This option only applies to executable files.)

-h *symbol* Prior to writing the symbol table to the output file, mark this name as "local" so that it is no longer externally visible. This ensures that this particular entry will not clash with a definition in another file during future processing by **ld**.

More than one *symbol* can be specified, but **-h** must precede each one. If used when building a shared library or program, this option prevents the named symbol from being visible to the dynamic loader.

-l*x* Search a library **lib***x***.a** or **lib***x***.sl**, where *x* is one or more characters. The current state of the **-a** option determines whether the archive (**.a**) or shared (**.sl**) version of a library is searched. Because a library is searched when its name is encountered, the placement of a **-l** is significant. By default, 32 bit libraries are located in **/usr/lib** and **/usr/ccs/lib**. 64 bit libraries are located in **/usr/lib/pa20_64**. If the environment variable **LPATH** is present in the user's environment, it should contain a colon-separated list of directories to search. These directories are searched instead of the default directories, but **-L** options can still be used. If a program uses shared libraries, the dynamic loader **/usr/lib/dld.sl** for 32 bit or **/usr/lib/pa20_64/dld.sl** for 64 bit will attempt to load each library from the same directory in which it was found at link time (see the **+s** and **+b** options).

-l: *library* Search the library specified. Similar to the **-l** option except the current state of the **-a** option is not important. The library name can be any valid filename. (Note that previous releases required that the library name contain the prefix **lib** and end with a suffix of **.a** or **.sl**.)

-m This option produces a load map on the standard output.

-n This option is accepted but ignored by the 64 bit **ld**. Generate an executable output file with file type **SHARE_MAGIC**. This is the default. This option is incompatible with **-N** and **-q**.

-o *outfile* Produce an output object file named *outfile* (**a.out** if **-o** *outfile* is not specified).

-q This option is ignored for 64 bit links. Generate an executable output file with file type **DEMAND_MAGIC**. This option is incompatible with **-n**, **-N**, and **-Q**.

-r Retain relocation information in the output file for subsequent re-linking. The **ld** command does not report undefined symbols. This option cannot be used when building a shared library (**-b**) or in conjunction with **-A**.

-s Strip the output file of all symbol table, relocation, and debug support information. This might impair or prevent the use of a symbolic debugger on the resulting program. This option is incompatible with **-r**. (The *strip*(1) command also removes this information.)

-t Print a trace (to standard output) of each input file as **ld** processes it.

-u *symbol* Enter *symbol* as an undefined symbol in the symbol table. The resulting unresolved reference is useful for linking a program solely from object files in a library. More than one *symbol* can be specified, but each must be preceded by **-u**.

-v Display verbose messages during linking. For each library module that is loaded, the linker indicates which symbol caused that module to be loaded.

-x Strip local symbols from the output file. This reduces the size of the output file without impairing the effectiveness of object file utilities. This option is incompatible with the **-r** option . Note: use of **-x** might affect the use of a debugger.

l

-y *symbol* Indicate each file in which *symbol* appears. More than one *symbol* can be specified, but each must be preceded by **-y**.

-z Arrange for run-time dereferencing of null pointers to produce a **SIGSEGV** signal. (This is the complement of the **-Z** option.)

-B *bind* Select run-time binding behavior of a program using shared libraries or the binding preference in building a shared library. The most common values for *bind* are:

deferred
Bind addresses on first reference rather than at program start-up time. This is the default.

immediate
Bind addresses of all symbols immediately upon loading the library. Commonly followed by **-B nonfatal** to allow procedure calls that cannot be resolved at program start-up to be resolved on first reference.

Since **-B nonfatal** suppresses messages about unresolved symbols, also specify **-B verbose** to display those messages.

See the example below.

nonfatal
If also using **-B immediate**, for code symbols that could not be bound at program startup, defer binding them until they are referenced. See description of **-B immediate** above.

Since **-B nonfatal** suppresses messages about unresolved symbols, also specify **-B verbose** to display those messages.

restricted
Causes the search for a symbol definition to be restricted to those symbols that were visible when the library was loaded.

symbolic
Used only when building a shared library (with the **-b** option), this option causes all unresolved symbols inside a library to be resolved internally if possible. By default, unresolved symbols are resolved to the most visible definition in

the library or outside of the library.

verbose
Display verbose messages when binding symbols. This is the default except when **-B nonfatal** is specified. In that case, **-B verbose** must be explicitly specified to get verbose messages.

See the **+help** option or the *HP-UX Linker and Libraries User's Guide* manual for more information on the uses of binding modes.

-D *offset* Set the origin (in hexadecimal) for the data segment.

-E This option is accepted but ignored by the 64 bit **ld**. Mark all symbols defined by a program for export to shared libraries. By default, **ld** marks only those symbols that are actually referenced by a shared library seen at link time.

-G Strip all unloadable data from the output file. This option is typically used to strip debug information.

-I Instrument the code to collect profile information upon execution. The profile data gathered during program execution can be used in conjunction with the **-P** option. 32 bit programs linked with this option should use the startup file **/opt/langtools/lib/icrt0.o**. This option should not be used with the **-P**, **-A**, **-O**, or **+O** options.

-L *dir* Search for **lib***x***.a** or **lib***x***.sl** in *dir* before looking in default locations. More than one directory can be specified, but each must be preceded by **-L**. The **-L** option is effective only if it precedes the **-l** option on the command line.

-O Turn on linker optimizations. Currently the optimizations include the elimination of unnecessary **ADDIL** instructions from the code in the executable file (32 bit only), and the removal of dead procedures.

-O is passed to the linker by the compilers when the **+O4** compiler option is selected.

For more details on linker optimizations refer to the **+help** option or the *HP-UX Linker and Libraries User's Guide* manual.

-P Examine the data file produced by an instrumented program (see the **-I** option) to perform profile based optimizations on the code. This option should not be used with the **-A** option.

-Q Ignored for 64 bit links. Generate an executable output file with file type **EXEC_MAGIC** or **SHARE_MAGIC**, depending on whether **-N** or **-n** is specified. This is the default. This option is incompatible with **-q**.

-R *offset* Set the origin (in hexadecimal) for the text (i.e., code) segment.

-V Output a message giving information about the version of **ld** being used.

-Z Allow run-time dereferencing of null pointers. See the discussions of **-Z** and *pointers* in *cc*(1). (This is the complement of the **-z** option.)

-dynamic This allows the linker to create a program which can use shared libraries. This is the default for 64 bit links unless **+compat** is used.

-noshared This option forces the linker to create a fully archive bound program.

+b *path_list* Specify a colon-separated list of directories to be searched at program run-time to locate shared libraries needed by the executable output file that were specified with either the **-l** or **-l:** option. An argument consisting of a single colon (**:**) indicates that **ld** should build the list using all the directories specified by the **-L** option and the **LPATH** environment variable (see the **+s** option).

+compat This option turns on compatibility mode in the linker — 64 bit links mimic behavior of 32 bit links.

+df *file* Used together with the **-P** option, this option specifies that *file* should be used as the profile database file. The default value is **flow.data**. See the discussion of the **FLOW_DATA** environment variable for more information.

+e *symbol* When building a shared library or program, mark the symbol for export to the dynamic loader. Only symbols explicitly marked are exported. When building a

1

shared library, calls to symbols that are not exported are resolved internally.

+ee *symbol* This option is similar to the **+e** option in that it exports a symbol. However, unlike the **+e** option, **+ee** does not alter the visibility of any other symbol in the file. In a 32 bit link or **+compat** mode 64 bit link, it has the effect of exporting the specified symbol without hiding any of the symbols exported by default. In a 64 **+std** link, all symbols are exported by default, so **+ee** is not necessary to make a symbol visible. However, it has the additional side effect of identifying the symbol as necessary, so that it will not be removed when using dead code elimination (**+Oprocelim**). Of course, **+ee** still retains its export behavior if an option such as **+hideallsymbols** is also given.

+fb Instructs the linker to run the fastbind tool on the executable it has produced. The executable should be linked with shared libraries. For more details refer to *fastbind*(1), the **+help** option, or the *HP-UX Linker and Libraries User's Guide* manual.

+fbu Pass the **-u** option to the fastbind tool. For more details refer to *fastbind*(1), the **+help** option, or the *HP-UX Linker and Libraries User's Guide* manual.

+h *internal_name*
When building a shared library, record *internal_name* as the name of the library. When the library is used to link another executable file (program or shared library), this *internal_name* is recorded in the library list of the resulting output file instead of the file system pathname of the input shared library.

If *internal_name* is a fully-qualified pathname, it is recorded **as is** in the library list of any executable file it is subsequently linked against. If *internal_name* is a relative pathname or no directory component was specified, *internal_name* is **appended** to the file system directory component of the input shared library in the library list of any executable file it is subsequently linked against.

If more than one +h option is seen on the link line, the first one is used and a warning message is emitted.

+help Starts the help window utility *HP-UX Linker and Libraries Online User Guide* which comes with some HP compilers. (You must be running the X window system and your **$DISPLAY** environment variable must be set to the name of your workstation or X terminal.) For more information, refer to the *HP-UX Linker and Libraries User's Guide* manual. See *manuals*(5) for ordering information.

+k Direct the linker to only create an executable if there were no errors encountered during the link. If there were errors found (system errors or unresolved references), the output file will be removed.

+n Causes the linker to load all object modules before searching any archive or shared libraries. Then it searches the archive and shared libraries specified on the command line in left to right order. Repeats the left to right search of the libraries on the command line until there are no more unsatisfied symbols, or the last search added no new definitions. This option is useful if two libraries are specified that have symbol dependencies on each other.

+pgm *name* Used together with the **-P** option, this option specifies that *name* should be used as the look-up name in the profile database file. The default is the basename of the output file (specified by the **-o** option.)

+s Indicates that at run-time, the shared library loader can use the environment variable **SHLIB_PATH** and **LD_LIBRARY_PATH** (64 bit only) to locate shared libraries needed by the executable output file that were specified with either the **-l** or **-l:** option. The environment variables should be set to a colon-separated list of directories. If both **+s** and **+b** are used, their relative order on the command line indicates which path list will be searched first (see the **+b** option).

+vallcompatwarnings
This option is accepted but ignored by the 64 bit **ld**. Show more detail for any warnings about compatibility issues. By default, only a terse message is printed. See the **WARNINGS** section below for further details.

+v[no]compatwarnings
This option is accepted but ignored by the 64 bit **ld**. Enable [disable] printing

warnings about compatibility issues between systems. This includes any functionality which may change in future releases. The default is **+vcompatwarnings**. See the **WARNINGS** section below for further details.

+std This option is ignored for 32 bit links. Turns on standard mode, which is the default. Options turned on with this option are: **-dynamic.** Options turned off or ignored when this option is specified are: **+compat**,**+noenvvar**,**-noshared**.

+v[no]shlibunsats
Enable [disable] printing a list of unsatisfied symbols used by shared libraries. The default is **+vnoshlibunsats**. Some unsatisfied symbols reported by the linker are not required at run time because the modules which reference the symbols are not used.

+FP *flag* Specify how the environment for floating-point operations should be initialized at program start-up. By default, all behaviors are disabled. The following flags are supported (upper case flag enables; lower case flag disables):

V (**v**)	Trap on invalid floating-point operations
Z (**z**)	Trap on divide by zero
O (**o**)	Trap on floating-point overflow
U (**u**)	Trap on floating-point underflow
I (**i**)	Trap on floating-point operations that produce inexact results.
D (**d**)	Enable sudden underflow (flush to zero) of denormalized values.

Note: Enabling sudden underflow is an undefined operation on PA-RISC 1.0-based systems, but it is defined on all subsequent versions of PA-RISC. Selecting this flag enables sudden underflow only if it is available on the processor being used at run-time.

To dynamically change these settings at run-time, see *fpgetround*(3M).

+I *symbol* Specify the name of the initializer function when building a shared library. A shared library may have multiple initializers specified. Initializers are executed in the order that they are specified. For more details on the initializer function, refer to the **+help** option or the *HP-UX Linker and Libraries User's Guide* manual.

+O[no]fastaccess
Enable [disable] fast access to global data. The linker rearranges the data to further reduce the number of **ADDIL** instructions in the executable.

This optimization may reveal some subtle programming errors related to assumptions about data layout. This optimization can occur at optimization levels 2, 3 and 4. The default is **+Onofastaccess** at optimization levels 2 and 3, and **+Ofastaccess** at optimization level 4.

This option is accepted but ignored by the 64 bit **ld**.

+O[no]procelim
Enable [disable] the elimination of procedures that are not referenced by the application. The default is **+Onoprocelim**. Procedure elimination can occur at any optimization level, including level 0. For more details refer to the **+help** option or the *HP-UX Linker and Libraries User's Guide* manual.

+Oselectivepercent *n*
Instructs the interprocedural optimizer driver to pass the first *n* percent of the object files to the high level optimizer for interprocedural optimizations such as inlining.

+Oselectivesize *size*
Instructs the interprocedural optimizer driver to pass the first k routines to the high level optimizer for interprocedural optimization where the size of k routines are approaching but less than *size.*

+OselectiveO3
Instructs the interprocedural optimizer driver to compile the routines not included in the +O4 list to be compiled at +O3.

1

+Ostaticprediction
Meaningful only on PA 2.0 architecture, this option sets the branch prediction bit in the output executable file's auxiliary header.

+pd *size* Request a particular virtual memory page size that should be used for data. Sizes of **4K**, **16K**, **64K**, **256K**, **1M**, **4M**, **16M**, **64M**, **256M**, **D**, and **L** are supported. A size of D allows the kernel to choose what page size should be used. A size of **L** will result in using the largest page size available. The actual page size may vary if the requested size cannot be fulfilled.

+pi *size* Request a particular virtual memory page size that should be used for instructions. See the **+pd** option for additional information.

32 Bit Link Editor Options

-A *name* This option specifies incremental loading. Linking is arranged so that the resulting object can be read into an already executing program. The argument *name* specifies a file whose symbol table provides the basis for defining additional symbols. Only newly linked material is entered into the text and data portions of **a.out**, but the new symbol table reflects all symbols defined before and after the incremental load. Also, the **-R** option can be used in conjunction with **-A**, which allows the newly linked segment to commence at the corresponding address. The default starting address is the old value of **_end**. The **-A** option is incompatible with **-r** and **-b**. Also note that a program that dynamically loads code with **ld -A** cannot use shared libraries. See the **+help** option or the *HP-UX Linker and Libraries User's Guide* manual for a description of this option.

-C*n* Set the maximum parameter-checking level to *n*. The default maximum is 3. See the language manuals for the meanings of the parameter-checking level.

1

-S Generate an Initial Program Loader (IPL) auxiliary header for the output file, instead of the default HP-UX auxiliary header.

-N Generate an executable output file with file type **EXEC_MAGIC**. This option is incompatible with **-n** and **-q**. This option causes the data to be placed immediately following the text, and makes the text writable. Files of this type cannot be shared.

-T Save the load data and relocation information in temporary files instead of in memory during linking. This option reduces the virtual memory requirements of the linker. If the **TMPDIR** environment variable is set, the temporary files are created in the specified directory, rather than in **/var/tmp**.

+cg *pathname* Specify the use of *pathname* as the code generator for compiling ISOMs to SOMs. See the discussion of profile based optimization for more information.

+dpv Display verbose messages regarding procedures which have been removed due to dead procedure elimination. The symbol name, input object file, and the size (in bytes) of the deleted procedure are displayed. The total size (in bytes) of the deleted procedures is also displayed.

64 bit Link Editor Options

-k *filename* **filename** specifies a mapfile that describes the output file memory map. Please refer to *HP-UX Linker and Libraries User's Guide* guide for more information. Also see **+nodefaultmap**.

+[no]allowunsats
+allowunsats Does not flag errors if the resulting output file has unsatisfied symbols. This is the default for relocatable links and **dll** builds. **+noallowunsats** Flags an error if the resulting output file has unsatisfied symbols. This is the default for program files.

+[no]forceload
+forceload loads all object files from archive libraries. **+noforceload** is the default — loads only the required object files from archive libraries. The mode that is selected, either explicitly or by default, remains on until it is changed.

+hideallsymbols
This option is used to prevent all the symbols from being exported unless explicitly exported with the **+e**.

+nodefaultmap
This option tells the linker not to use the default memory map. The user needs to supply a mapfile through the **-k** linker option.

+noenvvar Instructs the dynamic loader to not look at the LD_LIBRARY_PATH and SHLIB_PATH environment variables at runtime. This is turned on if **ld +compat** is specified. This is turned off by default or if **ld +std** is specified. See **+compat** or **+std**. Generally, this option is used for secure programs (e.g. setuid).

+stripunwind
Do not output the unwind table.

+vtype *type* Produces verbose output about the link operation. *type* can have the following values:

files
Dump info about each object file loaded.

libraries
Dump info about libraries searched.

procelim
Dump info about sections that have been rejected by the **+Oprocelim** option

sections
Dump info about each input section added to the output file.

symbols
Dump info about global symbols referenced/defined from/in the input files.

all Dumps all of the above info. Same as **-v**.

Defaults

Unless otherwise directed, **ld** names its output **a.out**. The **-o** option overrides this. Executable output files are of type **SHARE_MAGIC**. The default state of **-a** is to search shared libraries if available, archive libraries otherwise. The default bind behavior is **deferred**.

The default value of the **-Z**/**-z** option is **-Z**.

For 64 bit, **+std** is on by default.

EXTERNAL INFLUENCES

Environment Variables

The following internationalization variables affect the execution of **ld**:

FLOW_DATA
An instrumented executable (see the **-I** option) writes out the profile data to a database file named flow.data in the current directory. The name and location of this file can be specified by setting FLOW_DATA to the desired path name. The profile data is stored in the database file under a look-up name that is the same as the basename of the executable file specified at run-time. A single flow.data file can hold profile data for multiple program files.

LANG
Determines the locale category for native language, local customs and coded character set in the absence of **LC_ALL** and other **LC_*** environment variables. If **LANG** is not specified or is set to the empty string, a default of **C** (see *lang*(5)) is used instead of **LANG**.

LC_ALL
Determines the values for all locale categories and has precedence over **LANG** and other **LC_*** environment variables.

LC_MESSAGES
Determines the locale that should be used to affect the format and contents of diagnostic messages written to standard error.

LC_NUMERIC
Determines the locale category for numeric formatting.

LC_CTYPE
Determines the locale category for character handling functions.

NLSPATH
Determines the location of message catalogs for the processing of **LC_MESSAGES**.

If any internationalization variable contains an invalid setting, **ld** behaves as if all internationalization variables are set to **C**. See *environ*(5).

In addition, the following environment variable affects **ld**:

TMPDIR
Specifies a directory for temporary files (see *tmpnam*(3S)).

DIAGNOSTICS

ld returns zero when the link is successful. A non-zero return code indicates that an error occurred.

EXAMPLES

Link part of a C program for later processing by **ld**. (Note the **.o** suffix for the output object file; this is an HP-UX convention for indicating a linkable object file):

```
ld -r file1.o file2.o -o prog.o
```

On 32 bit, link a simple FORTRAN program for use with the **dde** symbolic debugger (see *dde*(1)). Output file name is **a.out** since there is no **-o** option in the command line.

```
ld /usr/ccs/lib/crt0.o ftn.o -lcl -lisamstub \
     -lc /opt/langtools/lib/end.o
```

To do this on 64 bit:

```
ld ftn.o -lcl -lisamstub \
     -lc /opt/langtools/lib/pa20_64/end.o
```

On 64 bit, link a shared bound program in standard mode. Note that **crt0.o** is not specified because for shared links, it is no longer necessary.

```
ld himom.o +std -lc
```

On 64 bit, link a compatibility mode program. **crt0.o** is included because this is an archive link.

```
ld /opt/langtools/lib/pa20_64/crt0.o himom.o +compat -a archive -lc
```

Create a shared library:

```
ld -b -o libfunc.sl func1.o func2.o func3.o
```

Create a shared library with an internal name:

```
ld -b -o libfoo1.1 foo1.o foo2.o +h libfoo1.1
```

On 32 bit, link a program with **libfunc.sl** but use the archive version of the C library. Specify the immediate binding mode together with the nonfatal modifier and allow verbose diagnostics to be displayed:

```
ld /usr/ccs/lib/crt0.o -B immediate -B nonfatal -B verbose \
     program.o -L . -lfunc -a archive -lc
```

To do this on 64 bit:

```
ld -B immediate -B nonfatal -B verbose \
     program.o -L . -lfunc -a archive -lc
```

On 32 bit, link a Pascal program:

```
ld /usr/ccs/lib/crt0.o main.o -lcl -lm -lc
```

Note that in the above examples, **/usr/ccs/lib/crt0.o** can be replaced by **/opt/langtools/lib/crt0.o**.

WARNINGS

ld recognizes several names as having special meanings. The symbol **_end** is reserved by the linker to refer to the first address beyond the end of the program's address space. Similarly, the symbol **_edata** refers to the first address beyond the initialized data, and the symbol **_etext** refers to the first address beyond the program text. The symbols **end**, **edata**, and **etext** are also defined by the linker, but only if the program contains a reference to these symbols and does not define them (see *end*(3C) for details). On 32 bit, the symbol **__tdsize** is the total thread local storage size required by the program or shared library.

On 64 bit, the linker defines a few more symbols. The symbol **__TLS_SIZE** is the total thread local storage size. The symbol **_FPU_STATUS** is the initial status of the FPU status register. The symbol **__SYSTEM_ID** is the largest architecture revision level used by any compilation unit.

The linker treats a user definition of any of the symbols listed here as an error.

Through its options, the link editor gives users great flexibility. However, those who invoke the linker directly must assume some added responsibilities. Input options should ensure the following properties for programs:

- When the link editor is called through *cc*(1), a start-up routine is linked with the user's program. This routine calls *exit*(2) after execution of the main program. If users call **ld** directly, they must ensure that the program always calls **exit()** rather than falling through the end of the entry routine.
- When linking for use with the symbolic debugger *dde*, the user must ensure that the program contains a routine called **main**. Also, the user must link in the file **/opt/langtools/lib/end.o** on 32 bit and **/opt/langtools/lib/pa20_64/end.o** on 64 bit as the last file named on the command line.

There is no guarantee that the linker will pick up files from archive libraries and include them in the final program in the same relative order that they occur within the library.

The linker emits warnings where ever it detects any compatibility issues. Among other things, these issues include architectural ones, as well as functionality that may change over time. Some of these include:

- Linking a PA 2.0 object file, which will not run on a PA 1.x system.
- Incremental loading with the **-A** option.
- Procedure call parameter and return type checking, including the **-C** option.
- Symbols with the same name but different types, such as CODE and DATA.
- Checking of unsatisfied symbols by the linker, which sometimes skips certain object files from an archived library. This warning is only given if the **-v** option is also provided.
- Versioning of objects within a shared library.

These messages can be turned off with the **+vnocompatwarnings** option.

As noted in the *Options* section, certain options no longer exist in a 64 bit linker. They are:

- **-q**
- **-A**
- **-C**
- **-E**
- **-Q**
- **-S**
- **-T**
- **-X**
- **+dpv**

1

AUTHOR

ld was developed by AT&T, the University of California, Berkeley, and HP.

FILES

/usr/lib/lib*	32 bit system archive and shared libraries
/usr/lib/pa20_64/lib*	64 bit system archive and shared libraries
/usr/ccs/lib*	32 bit development archive and shared libraries
/opt/langtools/lib/pa20_64	64 bit development object files, archive and shared libraries
a.out	output file
/usr/lib/dld.sl	32 bit dynamic loader
/usr/lib/pa20_64/dld.sl	64 bit dynamic loader
/opt/langtools/lib/end.o	for use with the 32 bit **dde** debugger

/opt/langtools/lib/pa20_64/end.o	for use with the 64 bit **dde** debugger
/usr/ccs/lib/crt0.o	32 bit run-time startup file
/opt/langtools/lib/crt0.o	Identical to **/usr/ccs/lib/crt0.o**
/opt/langtools/lib/pa20_64/crt0.o	64 bit run-time startup file
/usr/ccs/lib/dyncall.o	32 bit only. Used with **-A** option links
/opt/langtools/lib/mcrt0.o	32 bit run-time startup with profiling (see *prof*(1))
/opt/langtools/lib/pa20_64/mcrt0.o	64 bit run-time startup with profiling
/usr/lib/milli.a	32 bit millicode library automatically searched by **ld**
/usr/lib/pa20_64/milli.a	64 bit millicode library automatically searched by **ld**
/opt/langtools/lib/gcrt0.o	run-time start-up with profiling (see *gprof*(1))
/opt/langtools/lib/pa20_64/gcrt0.o	64 bit run-time start-up with profiling (see *gprof*(1))
/opt/langtools/lib/icrt0.o	32 bit run-time start-up with profiling (see discussion of profile based optimization above.)
/opt/langtools/lib/pa20_64/icrt0.o	64 bit run-time start-up with profiling (see discussion of profile based optimization above.)
/usr/lib/nls/$LANG/ld.cat	message catalog
/var/tmp/ld*	temporary files
flow.data	file containing profile data generated by running an instrumented executable
/usr/ccs/bin/fdp	program for creating the procedure link order from a profile database file created by an instrumented executable; forked by the **-P** option
/usr/ccs/lbin/uccom	PA-RISC code generator for the C language
/opt/fortran/lbin/uf77pass1	PA-RISC code generator for the FORTRAN language
/opt/langtools/lbin/ucomp	Default PA-RISC code generator

l

SEE ALSO

Profiling and Debugging Tools:

adb(1)	absolute debugger
gprof(1)	display call graph profile data
prof(1)	display profile data
dde(1)	C, C++, FORTRAN, and Pascal symbolic debugger

System Tools:

ar(1)	create archived libraries
CC(1)	invoke the HP-UX C++ compiler
cc(1)	invoke the HP-UX C compiler
chatr(1)	change program's internal attributes
exec(2)	execute a file
f77(1)	invoke the HP-UX FORTRAN compiler
fastbind(1)	invoke the fastbind tool
nm(1)	print name list of object file
pc(1)	invoke the HP-UX Pascal compiler
strip(1)	strip symbol and line number information from an object file

Miscellaneous:

a.out(4)	assembler, compiler, and linker output

ar(4)	archive format
crt0(3)	execution startup routine
dld.sl(5)	dynamic loader
end(3C)	symbol of the last locations in program

Texts and Tutorials:

HP-UX Linker and Libraries Online User Guide
(See the **+help** option)

HP-UX Linker and Libraries User's Guide
(See *manuals*(5) for ordering information)

STANDARDS CONFORMANCE

ld: SVID2, SVID3, XPG2, XPG4

NAME

ldd - list dynamic dependencies of executable files or shared libraries

SYNOPSIS

ldd [**-d**] [**-r**] [**-s**] [**-v**] *filename*...

DESCRIPTION

ldd is a command that can list the dynamic dependencies of incomplete executable files or shared libraries. **ldd** command will work only on 64-bit executables or shared libraries.

ldd lists verbose information about dynamic dependencies and symbol references. If the object file is an executable file, **ldd** will list all shared libraries that would be loaded as a result of executing the file. If it is a shared library, **ldd** will list all shared libraries that would be loaded as a result of loading the library.

ldd uses the same algorithm as the dynamic loader **/usr/lib/pa20_64/dld.sl** to locate the shared libraries.

Options

ldd recognizes the following options:

-d Check reference to data symbols.

-r Check reference to data and code symbols.

-s Displays the search path used to locate the shared libraries.

-v Display all dependency relationships.

EXTERNAL INFLUENCES

Environment Variables

The following internationalization variables affect the execution of **ldd**:

LANG
Determines the locale category for native language, local customs and coded character set in the absence of **LC_ALL** and other **LC_*** environment variables. If **LANG** is not specified or is set to the empty string, a default of **C** (see *lang*(5)) is used instead of **LANG**.

LC_ALL
Determines the values for all locale categories and has precedence over **LANG** and other **LC_*** environment variables.

LC_MESSAGES
Determines the locale that should be used to affect the format and contents of diagnostic messages written to standard error.

LC_NUMERIC
Determines the locale category for numeric formatting.

LC_CTYPE
Determines the locale category for character handling functions.

NLSPATH
Determines the location of message catalogs for the processing of **LC_MESSAGES**.

If any internationalization variable contains an invalid setting, **ldd** behaves as if all internationalization variables are set to **C**. See *environ*(5).

DIAGNOSTICS

ldd prints the record of shared library path names to stdout. The optional list of symbol resolution problems are printed to stderr.

ldd returns zero when the operation is successful. A non-zero return code indicates that an error occurred.

EXAMPLES

By default **ldd** will print a simple dynamic path information. This consists of the dependencies recorded in the executable (or the shared library) followed by the physical location where these libraries are found.

```
ldd a.out
    ./libx.sl =>    ./libx.sl
    libc.2 =>       /lib/pa20_64/libc.2
    libdl.1 =>      /lib/pa20_64/libdl.1
```

The **-v** option will cause **ldd** to print dependency relationship along with the dynamic path information.

```
ldd -v a.out
  find library=./libx.sl; required by a.out
    ./libx.sl =>    ./libx.sl
  find library=libc.2; required by a.out
    libc.2 =>       /lib/pa20_64/libc.2
  find library=libdl.1; required by /lib/pa20_64/libc.2
    libdl.1 =>      /lib/pa20_64/libdl.1
```

The **-r** option to **ldd** causes it to analyze all symbol references and print information about unsatisfied code and data symbols.

```
ldd -r a.out
    ./libx.sl =>    ./libx.sl
    libc.2 =>       /lib/pa20_64/libc.2
    libdl.1 =>      /lib/pa20_64/libdl.1
    symbol not found: val1 (./libx.sl)
    symbol not found: count (./libx.sl)
    symbol not found: func1 (./libx.sl)
```

WARNINGS

ldd does not list shared libraries explicitly loaded using *dlopen*(3X) or *shl_load*(3X).

FILES

a.out	output file
/usr/lib/pa20_64/dld.sl	64-bit PARISC dynamic loader
/usr/ccs/lib/pa20_64/lddstub	Dummy executable loaded to check the dependencies of shared libraries.
/usr/lib/nls/$LANG/ldd.cat	message catalog

SEE ALSO

System Tools:

ld(1)	invoke the link editor

Miscellaneous:

a.out(4)	assembler, compiler, and linker output
dld.sl(5)	dynamic loader

Texts and Tutorials:

HP-UX Linker and Libraries User's Guide

NAME

leave - remind you when you have to leave

SYNOPSIS

leave [*hhmm*]

DESCRIPTION

The **leave** command waits until the specified time, then reminds you to leave. You are reminded 5 minutes and 1 minute before the actual time, at the time, and every minute thereafter. When you log off, **leave** exits.

The time of day is in the form *hhmm*, where *hh* is a time in hours (which can range from 0 through 11 or 0 through 24 hours), and *mm* is the number of minutes after the specified hour. If the value of *hh* is greater than 11 (24-hour clock time), the specified value is reduced by 12 to a new value in the range of 0 through 11, thus ensuring that the alarm time is always set to activate within the next 12 hours. For example, if *hhmm* is 1350 and the current time is 4:00 PM (1600), the 1350 value is changed to 150 and the alarm is set for 1:50 AM, nine hours and 50 minutes later. On the other hand, if it is 9:00 AM and *hhmm* is specified as 2200 (10:00 PM), the value used is converted to 1000 and the alarm is set for one hour later instead of 13 hours as specified.

If no argument is provided, **leave** prompts with

```
When do you have to leave?
```

A reply of newline causes **leave** to exit; otherwise the reply is assumed to be a time. This form is suitable for inclusion in a **.login** or **.profile** file.

The **leave** command ignores interrupts, quits, and terminate signals. To get rid of it you should either log off or use **kill -9** giving its process ID.

EXAMPLES

The command

```
leave 1204
```

sends an alarm (a beep) to your terminal to remind you that you have to leave at 12:04 and reminds you that you are late at one minute intervals after 12:04.

WARNINGS

The **leave** command checks to see if a user has logged out by checking the **/etc/utmp** file every 100 seconds. If a user logs out and logs back in to the same tty before **leave** makes its periodic check, **leave** may not know that the user has logged out.

AUTHOR

leave was developed by the University of California, Berkeley.

FILES

/etc/utmp

SEE ALSO

calendar(1).

NAME

lifcp - copy to or from LIF files

SYNOPSIS

lifcp [**-T***xxx*] [**-L***xxx*] [**-v***xxx*] [**-a**] [**-b**] [**-i** *xxx*] [**-r**] [**-t**] *file1 file2*

lifcp [**-T***xxx*] [**-L***xxx*] [**-v***xxx*] [**-a**] [**-b**] [**-i***xxx*] [**-r**] [**-t**] [*file1 file2* ...] *directory*

DESCRIPTION

lifcp copies a LIF file to an HP-UX file, an HP-UX file to a LIF file, or a LIF file to another LIF file. It also copies a list of (HP-UX/LIF) files to a (LIF/HP-UX) directory. The last name on the argument list is the destination file or directory.

Options can be used singly or combined in any order before the file names. The space between option and argument is optional.

-T*xxx* Used only when copying files to a LIF volume. This option forces the file type of the LIF directory entry to be set to the argument given. The argument can be decimal, octal or hex, using standard "C" notation.

-L*xxx* Used only when copying files to a LIF volume. This option will set the "last volume flag" to *xxx* (0 or 1). The default "last volume flag" is **1**.

-v*xxx* Used only when copying files to a LIF volume. This option sets the "volume number" to *xxx*. The default "volume number" is one.

-a This option forces a ASCII mode of copying regardless of the file type. When copying in ASCII mode from HP-UX to LIF the default file type is BINARY (1). (For details on available modes of copying refer to *lif*(4)). This option has no effect when copying from LIF to LIF.

1

-b This option forces a BINARY mode of copying regardless of the file type. When copying in BINARY mode from HP-UX to LIF the default file type is BINARY (-2). (For details on available modes of copying refer to *lif*(4)). This option has no effect when copying from LIF to LIF.

-i*xxx* Used only when copying files to a LIF volume. This option sets the "implementation" field of the LIF directory entry to the argument given. The argument value can be decimal, octal or hex, using standard "C" notation. The "implementation" field can only be set for file types -2001 to -100000 (octal). The "implementation" field is set to zero for all interchange file types and for file types -2 to -200 (octal). Note that the "implementation" value controls the attributes of the LIF file with regard to protection and record sizes. **lifls -l** or **lifls -i** can be used to determine the "implementation" value of a file.

-r Forces RAW mode copying regardless of file type. When copying in RAW mode from HP-UX to LIF the default file type is BIN (-23951). **-T** option overrides the default file type. (various modes of copying are explained in *lif*(4).) -r option has no effect in LIF to LIF copy operations.

-t causes HP-UX file names to be translated to a name acceptable by a LIF utility; that is, all lowercase letters are converted to uppercase and all other characters except numerics are changed to an underscore (_). If the HP-UX file name starts with a nonletter, the file name is preceded by the capital letter X. Thus, for example, if two files named colon (:) and semicolon (;), were copied, both of them would be translated to X_. File names are truncated to a maximum of 10 characters. When copying a LIF file to an HP-UX or LIF file, **-t** has no effect. Omitting **-t** causes an error to be generated if an improper name is used.

The default copying modes when copying from LIF to HP-UX are summarized in the following table:

File Type	Default Copying Mode
ASCII	ASCII
BINARY	BINARY
BIN	RAW
other	RAW

When copying from HP-UX to LIF, the default copying mode is ASCII and an ASCII file is created.

When copying from LIF to LIF, if no options are specified, then all the LIF directory fields and file contents are duplicated from source to destination.

A LIF file name is recognized by the embedded colon (:) delimiter (see *lif*(4) for LIF file naming conventions). A LIF directory is recognized by a trailing colon. If an HP-UX file name containing a colon is used, the colon must be escaped with two backslash characters (\\) (the shell removes one of them).

The file name **-** (dash) is interpreted to mean standard input or standard output, depending on its position in the argument list. This is particularly useful if the data requires nonstandard translation. When copying from standard input, if no other name can be found, the name "STDIN" is used.

LIF file naming conventions are known only to the LIF utilities. Since file name expansion is done by the shell, this mechanism cannot be used for expanding LIF file names.

Do not mount the special file while using *lifcp*.

DIAGNOSTICS

lifcp returns exit code 0 if the file is copied successfully. Otherwise it prints a diagnostic and returns nonzero.

1

EXAMPLES

Copy HP-UX file **abc** to LIF file **CDE** on LIF volume **lifvol** which is actually an HP-UX file initialized to be a LIF volume:

lifcp abc lifvol:CDE

Copy all the HP-UX files in the current directory to the LIF volume **lifvol** which is present in the parent directory. File names are translated to appropriate LIF file names.

lifcp -t * ../lifvol:

Copy all the HP-UX object files in the current directory to the LIF volume *lifvol*. Copying mode is RAW and LIF file types are set to -5555.

lifcp -r -T -5555 -t *.o lifvol:

Copy all the object files in the current directory to the LIF volume **lifvol**. Copying mode is BINARY and LIF BINARY files are created.

lifcp -r -T 0xffffe961 -i 0x20200080 bdat lifvol:BDAT

Copy a BDAT file, without a password, from a BASIC WorkStation to an HP-UX LIF volume **lifvol**. Note that **-i** controls protection and record size attributes. The file type for a BDAT file is -5791 (or 0xffffe961) and its record size is 256 bytes per record.

lifcp -b *.o lifvol:

Copy all files in the current directory to the LIF volume **lifvol** in the **root** directory. Copying mode is RAW and LIF file types are set to BIN.

lifcp -r -t * /lifvol:

Copy file **abc:** to LIF file **CDE** in **lifvol**.

lifcp abc\\: lifvol:CDE

Copy files **abc** and **def** to LIF files **ABC** and **DEF** within **lifvol**.

lifcp -t abc def lifvol:

Copy LIF file **ABC** within **lifvol** to file **ABC** within current directory.

lifcp lifvol:ABC .

Copy standard input to LIF file **A_FILE** on LIF volume **/dev/dsk/c0t6d0**.

lifcp - /dev/dsk/c0t6d0:A_FILE

Copy LIF file **ABC** in **lifvol** to LIF file **CDE** on **/dev/dsk/c0t6d0 .**

lifcp lifvol:ABC /dev/dsk/c0t6d0:CDE

Copy the output of *pr* to the LIF file **ABC**.

pr abc | lifcp - lifvol:ABC

Copy the output of *pr* to the LIF volume **lifvol**. LIF file **STDIN** is created since no files names are specified.

pr abc | lifcp - lifvol:

Copy LIF file **ABC** in **lifvol** to standard output.

lifcp lifvol:ABC -

Copy all files in current directory to LIF files of the same name on LIF volume **lifvol** (may cause errors if file names in the current directory do not obey LIF naming conventions!).

lifcp * ../lifvol:

DEPENDENCIES

The following option is also supported:

-K*nnn* forces each file copied in to begin on a *nnn* × 1024-byte boundary from the beginning of the volume. This is useful when files are used for Series 700/800 boot media. This option has no effect when copying from a LIF volume.

AUTHOR

lifcp was developed by the Hewlett-Packard Company.

SEE ALSO

lifinit(1), lifls(1), lifrename(1), lifrm(1), lif(4).

NAME

lifinit - write LIF volume header on file

SYNOPSIS

lifinit [**-v***nnn*] [**-d***nnn*] [**-n** *string*] [**-s***nnn*] [**-l***nnn*] [**-e***nnn*] *file*

DESCRIPTION

lifinit writes a LIF volume header on a volume or file.

Options

lifinit recognizes the following options and command-line arguments which can appear in any order:

-v*nnn* Sets volume size to *nnn* bytes. If *nnn* is not a multiple of 256, it is rounded down to the next such multiple.

-d*nnn* Sets directory size to *nnn* file entries. If *nnn* is not an integer multiple of 8, it is rounded up to next such multiple.

-n *string* Sets the volume name to be *string*. If the **-n** option is not specified, the volume name is set to the last component of the path name specified by *file*. A legal LIF volume name is 6 characters long and is limited to uppercase letters (A-Z), digits (0-9) and the underscore character (_). The first character (if any) must be a letter. The utility automatically performs translation to create legal LIF volume names. Therefore, all lowercase letters are converted to uppercase, and all other characters except numeric and underscore are replaced with a capital letter **X**. If the volume name does not start with a letter, the volume name is preceded by a capital letter **X**. The volume name is also right-padded with spaces or truncated as needed to be six characters long. If **-n** is used with no *string*, the default volume name is set to 6 spaces.

1

-s*nnn* set the initial system load (ISL) start address to *nnn* in the volume label. This is useful when building boot media for Series 700/800 systems.

-l*nnn* specifies the length in bytes of the ISL code in the LIF volume.

-e*nnn* set the ISL entry point to *nnn* bytes from the beginning of the ISL. For example, specifying **-e3272** means that the ISL entry point is 3272 (decimal) bytes from the beginning of the ISL object module.

-K*nnn* forces the directory start location to be the nearest multiple of *nnn* × 1024 bytes from the beginning of the volume. This is necessary for booting Series 700/800 systems from LIF media.

If *file* does not exist, a regular HP-UX disk file is created and initialized.

The default values for volume size are 256 kilobytes for regular files, and the actual capacity of the device for device files.

The default directory size is a function of the volume size. A percentage of the volume size is allocated to the volume directory as follows:

Volume Size	Directory Size
< 2MB	~1.3%
> 2MB	~0.5%

Each directory entry occupies 32 bytes of storage. The actual directory space is subject to the rounding rules stated above.

Do not mount the special file while using **lifinit**.

RETURN VALUE

lifinit returns exit code 0 if the volume is initialized successfully. Otherwise it prints a diagnostic message and returns nonzero.

EXAMPLES

Initialize file **x** to be a LIF volume containing 500 000 bytes with 10 directory file entries:

```
lifinit -v500000 -d10 x
```

Initialize device **/dev/rdsk/c0t6d0** as a LIF volume using default initialization conditions (device must not be a mounted file system device):

```
lifinit /dev/rdsk/c0t6d0
```

WARNINGS

To prevent media corruption, do not terminate *lifinit* once it has started executing.

AUTHOR

`lifinit` was developed by HP.

SEE ALSO

lifcp(1), lifls(1), lifrename(1), lifrm(1), lif(4).

NAME

lifls - list contents of a LIF directory

SYNOPSIS

lifls [**option**] *name*

DESCRIPTION

lifls lists the contents of a LIF directory on standard output. The default output format lists file names in multiple columns (similar to *ls*(1), except unsorted) if standard output is a character special file. If standard output is not a tty device, the output format is one file name per line. *name* is a path name to an HP-UX file containing a LIF volume and optional file name. If *name* is a volume name, the entire volume is listed. If *name* is of the form *volume*:*file*, only the file is listed. The following options are available, and only one option should be specified with a given command:

-l List in long format, giving volume name, volume size, directory start, directory size, file type, file size, file start, "implementation" field (in hex), date created, last volume, and volume number.

-C Force multiple column output format regardless of standard output type.

-L Return the content of the "last volume flag" in decimal.

-i Return the content of the "implementation" field in hex.

-v Return the content of the "volume number" in decimal.

-b *blist*
Report only on files using block numbers specified on the command line in *blist*, a comma separated list of block numbers in DEV_BSIZE units.

Do not mount the special file while using **lifls**.

DIAGNOSTICS

lifls returns zero if the directory was listed successfully. Otherwise it prints a diagnostic and returns nonzero.

EXAMPLES

```
lifls -C /dev/rdsk/c0t6d0
```

AUTHOR

lifls was developed by HP.

SEE ALSO

lifcp(1), lifinit(1), lifrename(1), lifrm(1), lif(4).

NAME

lifrename - rename LIF files

SYNOPSIS

lifrename *oldfile newfile*

DESCRIPTION

oldfile is a full LIF file specifier (see *lif*(4) for details) for the file to be renamed (e.g. **liffile:A_FILE**). *newfile* is new name to be given to the file (only the file name portion). This operation does not include copy or delete. Old file names must match the name of the file to be renamed, even if that file name is not a legal LIF name.

Do not mount the special file while using **lifrename**.

DIAGNOSTICS

lifrename returns zero if the file name is changed successfully. Otherwise it prints a diagnostic and returns nonzero.

EXAMPLES

```
lifrename liffile:A_FILE B_FILE
lifrename /dev/dsk/c0t6d0:ABC CDE
```

AUTHOR

lifrename was developed by HP.

SEE ALSO

lifcp(1), lifinit(1), lifls(1), lifrm(1), lif(4).

NAME

lifrm - remove a LIF file

SYNOPSIS

lifrm *file1* ... *filen*

DESCRIPTION

lifrm removes one or more entries from a LIF volume. File name specifiers are as described in *lif*(4).

Do not mount the special file while using **lifrm**.

DIAGNOSTICS

lifrm returns zero if the file is removed successfully. Otherwise it prints a diagnostic and returns nonzero.

EXAMPLES

```
lifrm liffile:MAN
lifrm /dev/rdsk/c0t6d0:F
```

AUTHOR

lifrm was developed by HP.

SEE ALSO

lifcp(1), lifinit(1), lifls(1), lifrename(1), lif(4).

NAME

line - read one line from user input

SYNOPSIS

line [**-t** *timeout*]

DESCRIPTION

line copies one line (up to a new-line) from the standard input and writes it on the standard output. It returns an exit code of 1 on EOF and always prints at least a new-line. It is often used within shell files to read from the user's terminal.

Options

line recognizes the following command-line option:

-t *timeout* Timeout after *timeout* seconds where *timeout* is an integer value (if a non-integer value is specified, it is converted to an integer; i.e., rounded down). A blank is required between **-t** and the *timeout* argument. This option is not documented in POSIX and other industry standards, and should not be used in portable applications.

EXTERNAL INFLUENCES

International Code Set Support

Single- and multi-byte character code sets are supported.

EXAMPLES

The following lines in a shell script prompt for a file name and display information about the file:

```
echo 'Enter file name: \c'
reply=`line`
ls -l $reply
```

To limit the response time to 10 seconds, use the form:

```
reply=`line -t 10`
```

then test for no response. If no response occurs before timeout expires, a default behavior should be provided.

WARNINGS

This command is likely to be withdrawn from X/Open standards. Applications using this command might not be portable to other vendors' systems. As an alternative **read** is recommended.

SEE ALSO

sh(1), read(2).

STANDARDS CONFORMANCE

line: SVID2, SVID3, XPG2, XPG3

NAME

listusers - display user login data

SYNOPSIS

listusers [**-g** *groups*] [**-l** *logins*]

DESCRIPTION

The **listusers** command displays data concerning user logins. The output shows the user login and the **/etc/passwd** comment field value (e.g., user name, etc.). The default displays data about all user logins.

Options

The **listusers** command supports the following options:

-g *groups* Display all users belonging to *groups*, sorted by login. A comma separated list specifies multiple groups.

-l *logins* Display the requested *logins*. A comma separated list specifies multiple logins.

A user login has a UID of 100 or greater.

When the **-l** and **-g** options are combined, a user login is only displayed once, even though the login may belong to multiple specified groups.

EXAMPLES

List all user logins.

```
listusers
```

List all user logins in the group **cmds** and the users **bob**, **john** and **otto**, removing all duplicates.

```
listusers -g cmds -l bob,john,otto
```

FILES

/etc/passwd
/etc/group

SEE ALSO

passwd(1), logins(1M), passwd(1M), group(4), passwd(4).

STANDARDS COMPLIANCE

listusers: SVID3

NAME

ln - link files and directories

SYNOPSIS

ln [**-f**] [**-i**] [**-s**] *file1 new_file*

ln [**-f**] [**-i**] [**-s**] *file1* [*file2* ...] *dest_directory*

ln [**-f**] [**-i**] [**-s**] *directory1* [*directory2* ...] *dest_directory*

DESCRIPTION

The **ln** command links:

- *file1* to a new or existing *new_file*,
- *file1* to a new or existing file named *file1* in existing *dest_directory*,
- *file1*, *file2*, ... to new or existing files of the same name in existing *dest_directory*,
- *directory1*, *directory2*, ... to new directories of the same name in existing *dest_directory*,
- or it creates symbolic links between files or between directories.

If links are to *dest_directory*, corresponding file or directory names in that directory are linked to *file1*, *file2*, ..., or *directory1*, *directory2*, ..., etc., as appropriate. If two or more existing files or directories (excluding destination file name *new_file*) are specified, the destination must be a directory. If *new_file* already exists as a regular file (or link to another file), its contents (or the existing link) and its ACL are destroyed only if the **-f** option is specified. The ACL on the *new_file* after the link is the same as that of the *source_file* file.

If the **-f** and **-i** options are specified and the link being created is the name of an existing link or ordinary file and the access permissions of the file forbid writing, **ln** asks permission to overwrite the file. If the access permissions of the directory forbid writing, **ln** aborts and returns with the error message:

cannot unlink *new_file*

(even if the file is an ordinary file and not a link to another file). When asking for permission to overwrite an existing file or link, **ln** prints the mode (see *chmod*(2) and *Access Control Lists* below), followed by the first letters of the words **yes** and **no** in the current native language, prompting for a response, and reading one line from the standard input. If the response is affirmative and is permissible, the operation occurs; if not, the command proceeds to the next source file, if any.

Hard links are created with the same ownerships and permissions as the file or directory to which they are linked. If ownership or permissions are changed on a link or file, the same changes appear on corresponding hard links. The **ln** command does not permit hard links to a directory.

Symbolic links are created with the ownership of the creator and the permissions are of the creator's current umask. Once created, the symbolic link ownership and permissions will not change, since the mode and ownership of the symbolic link is ignored by the system.

If *file1* is a file and *new_file* is a link to an existing file or an existing file with other links, *new_file* is disassociated from the existing file and links and linked to *file1*. When **ln** creates a link to a new or existing file name, ownerships and permissions are always identical to those for the file to which it is linked. If **chown**, **chgrp**, or **chmod** is used to change ownership or permissions of a file or link, the change applies to the file and all associated links. The last modification time and last access time of the file and all associated links are identical (see *chown*(1) and *chmod*(1)).

For a discussion of symbolic links, see *symlink*(4).

Options

The **ln** command recognizes the following options:

-f Force existing destination path names to be removed to allow the link.

-i Write a prompt to the standard error output requesting confirmation for each link that would overwrite an existing file. This option takes effect only if used in conjunction with the **-f** option.

-s Cause **ln** to create symbolic links instead of the usual hard links. A symbolic link contains the name of the file to which it is linked. The referenced file is used when an **open()** operation is performed on the link (see *open*(2)). A **stat()** on a symbolic link returns the linked-

l

to file; an **lstat()** must be performed to obtain information about the link (see *stat*(2)). A **readlink()** call can be used to read the contents of the symbolic link (see *readlink*(2)). Symbolic links may span file systems and refer to directories.

Access Control Lists (ACLs)

If optional ACL entries are associated with *new_file*, **ln** displays a plus sign (**+**) after the access mode when asking permission to overwrite the file.

If *new_file* is a new file, it inherits the access control list of *file1*, altered to reflect any difference in ownership between the two files (see *acl*(5)).

EXTERNAL INFLUENCES

Environment Variables

LC_CTYPE determines the interpretation of text as single byte and/or multibyte characters.

LANG and **LC_CTYPE** determine the local language equivalent of **y** (for yes/no queries).

LANG determines the language in which messages are displayed.

If **LC_CTYPE** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of **C** (see *lang*(5)) is used instead of **LANG**. If any internationalization variable contains an invalid setting, **ln** behaves as if all internationalization variables are set to **C**. See *environ*(5).

International Code Set Support

Single byte and multibyte character code sets are supported.

EXAMPLES

The following command creates **file1** and **file2** in **dest_dir**, which are linked back to the original files **file1** and **file2**:

```
ln -f file1 file2 dest_dir
```

If **file1** and/or **file2** exists in the destination directory, it is removed and replaced by a link to **file1** or **file2**, respectively. If existing file **file1** or **file2** is a link to another file or a file with links, the existing file remains. Only the link is broken and replaced by a new link to **file1** or **file2**.

WARNINGS

ln does not create hard links across file systems.

DEPENDENCIES

NFS

Access control lists of networked files are summarized (as returned in **st_mode** by **stat()**), but not copied to the new file. When using **ln** on such files, a **+** is not printed after the mode value when asking for permission to overwrite a file.

AUTHOR

ln was developed by AT&T, the University of California, Berkeley and HP.

SEE ALSO

cp(1), cpio(1), mv(1), rm(1), link(1M), readlink(2), stat(2), symlink(2), symlink(4), acl(5).

STANDARDS CONFORMANCE

ln: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

NAME

locale - get locale-specific (NLS) information

SYNOPSIS

locale [**-a**|**-m**]

locale [**-ck**] *name* ...

DESCRIPTION

The **locale** command displays information about the current locale or about available locales.

When invoked without arguments, **locale** displays the name and actual or implied value of each of the locale-related environment variables in the order shown below, one per line:

```
LANG
LC_CTYPE
LC_COLLATE
LC_MONETARY
LC_NUMERIC
LC_TIME
LC_MESSAGES
LC_ALL
```

An actual value is the value the variable actually has in the user's environment. An implied value is derived from the value of another variable. Implied values are displayed enclosed in double quotes, while actual values are unquoted.

The determination of implied values is that if the variable **LC_ALL** is present and has a non-null value, that is the actual value for **LC_ALL**, and all of the other variables take its value as an implied value. If **LC_ALL** is not set, all of the **LC_*** variables that are set are shown with their value as an actual value. Any that have no value are shown with the value of the **LANG** environment variable as their implied value. **LC_ALL** is displayed as **LC_ALL=\n** if it has no value.

The **locale** command can take multiple arguments, which may be locale category names, locale keywords, or the special word **charmap** (see *localedef*(1M) for a description of locale keywords and charmaps). If an argument is a keyword, the value associated with that keyword in the current environment is displayed and possibly other information, depending on selected options. If an argument is a category name (i.e., **LC_***), the values of all keywords defined in that category are displayed. If an argument is the special word **charmap**, the charmap file (if any) that was used in the definition of the current locale is displayed.

Non-printable characters are printed as hexadecimal values in the form,

\x*hh*

except that if a different escape character has been defined for the locale, it is displayed instead of the "\".

Options

The following options are available:

-a List all available locales. These are the possible meaningful values that can be assigned to **LANG** or any of the **LC_*** variables on the system. They are dependent upon which locales have been installed on the system. By default, the locales in **/usr/lib/nls/loc/locales** are listed.

-c Display names of locale categories that have been selected either explicitly or by giving a keyword contained therein. This option may be used with the **-k** option.

-k Display names of keywords that have been selected either explicitly or by providing their containing category as an argument. Keyword names and values are displayed as:

*<keyword>***=***<value>*

Without the **-k** option, only the values are displayed. This option can be used with the **-c** option.

-m Display a list of available charmap files on the system. See *localedef*(1M) for a definition of charmap files and their usage.

EXTERNAL INFLUENCES

Environment Variables

LANG provides a default value for the internationalization variables that are unset or null. If **LANG** is unset or null, the default value of "C" (see *lang*(5)) is used. If any of the internationalization variables contains an invalid setting, **locale** will behave as if all internationalization variables are set to "C". See *environ*(5).

LC_ALL, when set to a non-empty string value, overrides the values of all other internationalization variables.

LC_CTYPE determines the interpretation of text as single and/or multi-byte characters, the classification of characters as printable, and the characters matched by character class expressions in regular expressions.

LC_MESSAGES determines the locale that should be used to affect the format and content of diagnostic messages written to standard error, and informative messages written to standard output.

NLSPATH determines the location of message catalog for the processing of **LC_MESSAGES**.

International Code Set Support

Single- and multi-byte character code sets are supported.

RETURN VALUE

The **locale** command exits with one of the following values:

0 All requested information was found and displayed successfully.

>0 An error occurred in either finding or displaying the information.

EXAMPLES

If the locale environment variables are set as:

```
LANG=fr_FR.iso88591
LC_COLLATE=C
```

the command:

```
locale
```

gives the following output:

```
LANG=fr_FR.iso88591
LC_CTYPE="fr_FR.iso88591"
LC_COLLATE=C
LC_MONETARY="fr_FR.iso88591"
LC_NUMERIC="fr_FR.iso88591"
LC_TIME="fr_FR.iso88591"
LC_MESSAGES="fr_FR.iso88591"
LC_ALL=
```

The command:

```
LC_ALL=POSIX locale -ck decimal_point
```

produces:

```
LC_NUMERIC decimal_point="."
```

If **LANG** is set to **POSIX** and no other locale variables are set, the command:

```
locale LC_NUMERIC
```

produces:

```
"."
""

""
```

which correspond to the keywords *decimal_point*, *thousands_sep*, *grouping*, and *alt_digit*.

SEE ALSO

localedef(1M), localeconv(3C), nl_langinfo(3C), setlocale(3C), charmap(4), localedef(4), environ(5), lang(5).

STANDARDS CONFORMANCE

`locale`: XPG4, POSIX.2

NAME

lock - reserve a terminal

SYNOPSIS

lock

DESCRIPTION

lock requests a password from the user, then prints **LOCKED** on the terminal and refuses to relinquish the terminal until the password is repeated. If the user forgets the password, the only recourse is to log in elsewhere and kill the lock process.

NAME

logger - make entries in the system log

SYNOPSIS

logger [**-t** *tag*] [**-p** *pri*] [**-i**] [**-f** *file*] [*message* ...]

DESCRIPTION

The **logger** command provides a program interface to the **syslog()** system log module (see *syslog*(3C)).

A message can be given on the command line, which is logged immediately, or a file is read and each line is logged. If no *file* or *message* is specified, the contents of the standard input are logged.

Options

The **logger** command recognizes the following command-line options and arguments:

-t *tag*	Mark every line in the log with the specified *tag*. The default is the value returned by **getlogin()** (see *getlogin*(3C)). If **getlogin()** returns NULL, **syslog** is the default.
-p *pri*	Enter the message with the specified priority. The priority can be specified numerically or as a *facility***.***level* pair. For example, **-p local3.info** logs the message or messages as **info**rmational level in the **local3** facility. The default is **user.notice**.
-i	Log the process ID of the logger process with each line.
-f *file*	Log the contents of the specified file.
message	The message to log; if not specified, the file specified by the **-f** option or standard input is logged.

EXTERNAL INFLUENCES

Environment Variables

LC_MESSAGES determines the language in which messages are displayed.

If **LC_MESSAGES** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**.

If any internationalization variable contains an invalid setting, **logger** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported.

EXAMPLES

Send the message **System rebooted** to the **syslogd** daemon:

```
logger System rebooted
```

Send output from the **users** command (see *users*(1) to the **syslogd** daemon, marked as level **info** and facility **local0**. The message is tagged with the string **USERS**:

```
users | logger -p local0.info -t USERS
```

Send the message **System going down immediately!!!** to the **syslog** daemon, at the **emerg** level and **user** facility:

```
logger -p user.emerg "System going down immediately!!!"
```

WARNINGS

The **logger** command has no effect if the **syslogd** daemon (see *syslogd*(1M)) is not running on the system.

Messages written in locales other than the POSIX/C locale are not supported.

AUTHOR

logger was developed by the University of California, Berkeley.

SEE ALSO

syslogd(1M), getlogin(3C), syslog(3C).

STANDARDS CONFORMANCE

logger: XPG4, POSIX.2

NAME

login - sign on; start terminal session

SYNOPSIS

login [*name* [*env-var*] ...]

DESCRIPTION

The **login** command is used at the beginning of each terminal session to properly identify a prospective user. **login** can be invoked as a user command or by the system as an incoming connection is established. **login** can also be invoked by the system when a previous user shell terminates but the terminal does not disconnect.

If **login** is invoked as a command, it must replace the initial command interpreter (the user's login shell). This is accomplished with the shell command

```
exec login
```

The user's login name is requested, if it is not specified on the command line, and the corresponding password is obtained, if required, with the following prompts:

```
login:
Password:
```

Terminal echo is turned off (where possible) during password entry to prevent written records of the password. If the account does not have a password, and the authentication profile for the account requires one, **login** invokes **pam_chauthtok()** to establish one for the account. On a trusted system, **login** displays the last successful and unsuccessful login times and terminal devices.

As a security precaution, some installations use an option that requires a second "dialup" password. This occurs only for dialup connections, and is requested with the prompt:

```
dialup password:
```

Both passwords must be correct for a successful login (see *dialups*(4) for details on dialup security).

If password aging is activated, the user's password may have expired. **pam_chauthtok()** is invoked to change the password. In an untrusted environment, the user is required to re-login after a successful password change (see *passwd*(1)).

After three unsuccessful login attempts, a **HANGUP** signal is issued. If a login is not successfully completed within a certain period of time (for example, one minute), the terminal is silently disconnected.

Ater a successful login, the accounting files are updated, user and group IDs, group access list, and working directory are initialized, and the user's command interpreter (shell) is determined from corresponding user entries in the files **/etc/passwd** and **/etc/logingroup** (see *passwd*(4) and *group*(4)). If **/etc/passwd** does not specify a shell for the user name, **/usr/bin/sh** is used by default. **login** then forks the appropriate shell by using the last component of the shell path name preceded by a **-** (for example, **-sh** or **-ksh**). When the command interpreter is invoked with its name preceded by a minus in this manner, the shell performs its own initialization, including execution of profile, login, or other initialization scripts.

For example, if the user login shell is the Bourne, Korn, or POSIX shell (see *sh-bourne*(1), *ksh*(1), or *sh-posix*(1), respectively), the shell executes the profile files **/etc/profile** and **$HOME/.profile** if they exist (and possibly others as well). Depending on what these profile files contain, messages regarding mail in the user's mail file or any messages the user may have received since the user's last login may be displayed.

If the command name field is *****, a **chroot()** to the directory named in the directory field of the entry is performed. At that point, **login** is re-executed at the new level, which must have its own root structure, including a **/usr/bin/login** command and an **/etc/passwd** file.

For the normal user, the basic environment variables (see *environ*(5)) are initialized to:

HOME=*login_directory*
LOGNAME=*login_name*
MAIL=/var/mail/*login_name*
PATH=:/usr/bin
SHELL=*login_shell*

1

login_directory, *login_name*, and *login_shell* are taken from the corresponding fields of the **passwd** file entry (see *passwd*(4)).

For superuser, **PATH** is set to:

```
PATH=:/usr/sbin:/usr/bin:/sbin
```

In the case of a remote login, the environment variable **TERM** is also set to the remote user's terminal type.

The environment can be expanded or modified by supplying additional arguments to **login**, either at execution time or when **login** requests the user's login name. The arguments can take either the form *value* or *varname*=*value*, where *varname* is a new or existing environment variable name and *value* is a value to be assigned to the variable.

An argument in the first form (without an equals sign) is placed in the environment as if it were entered in the form

L*n***=***value*

where *n* is a number starting at 0 that is incremented each time a new variable name is required.

An argument in the second form (with an equals sign) is placed into the environment without modification.

If the variable name (**L***n* or *varname*) already appears in the environment, the new value replaces the older one.

There are two exceptions. The variables **PATH** and **SHELL** cannot be changed. This prevents users logged in with restricted shell environments from spawning secondary shells that are not restricted.

Both **login** and **getty** understand simple single-character quoting conventions. Typing a backslash in front of a character quotes it and allows the inclusion of such things as spaces and tabs.

1

If **/var/adm/btmp** is present, all unsuccessful login attempts are logged to that file. This feature is disabled if the file is not present. The **lastb** command, (see *last*(1)), displays a summary of bad login attempts for users with read access to **btmp**.

If the **/etc/securetty** file is present, login security is in effect. Only user **root** is allowed to log in successfully on the ttys listed in this file. Restricted ttys are listed by device name, one per line. Valid tty names are dependent on the installation. An example is

```
console
tty01
ttya1
etc.
```

Note that this feature does not inhibit a normal user from using the **su** command (see *su*(1)).

HP-UX Smart Card Login

If the user account is configured to use a Smart Card, the user password is stored in the card. This password has characteristics identical to a normal password stored on the system.

In order to login using a Smart Card account, the card must be inserted into the Smart Card reader. The user is prompted for a PIN (personal identification number) instead of a password during authentication. The prompts are:

```
login:
Enter PIN:
```

The password is retrieved automatically from the Smart Card when a valid PIN is entered. Therefore, it is not necessary to know the password, only the PIN.

The card is locked if an incorrect PIN is entered three consecutive times. It may be unlocked only by the card issuer.

SECURITY FEATURES

On a trusted system, **login** prohibits a user from logging in if any of the following is true:

- The password for the account has expired and the user cannot successfully change the password.
- The password lifetime for the account has passed.
- The time between the last login and the current time exceeds the time allowed for login intervals.

- The administrative lock on the account has been set.
- The maximum number of unsuccessful login attempts for the account has been exceeded.
- The maximum number of unsuccessful login attempts for the terminal has been exceeded.
- The administrative lock on the terminal has been set.
- The terminal has an authorized user list and the user is not on it.
- The terminal has time of day restrictions and the current time is not within the allowable period.

On a trusted system, **login** allows superuser to log in on the console unless **/etc/securetty** exists and does not contain **console**.

EXTERNAL INFLUENCES

Environment Variables

HOME	User's home directory.
MAIL	Where to look for mail.
PATH	Path to be searched for commands.
SHELL	Which command interpreter is being used.
TERM	User's terminal type.
varname	User-specified named variables.
L*n*	User-specified unnamed variables.

DIAGNOSTICS

The following diagnostics appear if the associated condition occurs:

.rhosts is a soft link

The personal equivalence file is a symbolic link.

Bad .rhosts ownership

The personal equivalence file is not owned by the local user or by a user with appropriate privileges.

Bad group id

setgid() failed (see *setuid*(2)).

Bad user id

setuid() failed (see *setuid*(2)).

Cannot open password file

Consult system administrator.

Locuser too long

The indicated string was too long for **login**'s internal buffer.

Login incorrect

User name and password cannot be matched.

No /usr/bin/login or /etc/login on root

Attempted to log in to a subdirectory root that does not have a subroot login command. That is, the **passwd** file entry had shell path *****, but the system cannot find a **login** command under the given home directory.

No directory

Consult system administrator.

No Root Directory

Attempted to log in to a subdirectory root that does not exist. That is, the **passwd** file entry had shell path *****, but the system cannot **chroot()** to the given home directory.

No shell

The user shell (**/usr/bin/sh** if shell name is null in **/etc/passwd**) could not be started with the **exec** command. Consult system administrator.

No utmp entry. You must exec "login" from the lowest level "sh"

Attempted to execute **login** as a command without using the shell's **exec** internal command or from other than the initial shell. The current shell is terminated.

Remuser too long

The indicated string was too long for **login**'s internal buffer.

Terminal type too long

The indicated string was too long for **login**'s internal buffer.

Unable to change to directory *name*

Cannot **chdir** to the user's home directory.

Your password has expired. Choose a new one

Password aging is enabled and the user's password has expired.

WARNINGS

If **/etc/group** is linked to **/etc/logingroup**, and group membership for the user trying to log in is managed by the Network Information Service (NIS), and no NIS server is able to respond, **login** waits until a server does respond.

DEPENDENCIES

Pluggable Authentication Modules (PAM)

PAM is an Open Group standard for user authentication, password modification, and validation of accounts. In particular, **pam_authenticate()** is invoked to perform all functions related to **login**. This includes retrieving the password, validating the account, and displaying error messages. **pam_chauthtok()** is invoked during password expiration or establishment.

HP Process Resource Manager

If the optional HP Process Resource Manager (PRM) software is installed and configured, the login shell is launched in the user's initial process resource group. If the user's initial group is not defined, the shell runs in the user default group (**PRMID=1**). See *prmconfig*(1) for a description of how to configure HP PRM, and *prmconf*(4) for a description of how the user's initial process resource group is determined.

AUTHOR

login was developed by AT&T and HP.

FILES

$HOME/.profile	Personal profile (individual user initialization)
$HOME/.rhosts	Personal equivalence file for the remote login server
/etc/d_passwd	Dialup security encrypted passwords
/etc/dialups	Lines which require dialup security
/etc/hosts.equiv	System list of equivalent hosts allowing logins without passwords
/etc/logingroup	Group file — defines group access lists
/etc/motd	Message-of-the-day
/etc/passwd	Password file — defines users, passwords, and primary groups
/etc/profile	System profile (initialization for all users)
/etc/securetty	List of valid ttys for root login
/etc/utmp	Users currently logged in
/tcb/files/auth/*/*	The trusted system password database
/var/adm/btmp	History of bad login attempts
/var/adm/wtmp	History of logins, logouts, and date changes
/var/mail/*login_name*	Mailbox for user *login_name*

SEE ALSO

csh(1), groups(1), ksh(1), last(1), mail(1), newgrp(1), passwd(1), sh(1), sh-bourne(1), sh-posix(1), su(1), getty(1M), initgroups(3C), dialups(4), group(4), passwd(4), profile(4), utmp(4), environ(5).

HP Process Resource Manager

prmconfig(1), prmconf(4) in *HP Process Resource Manager Users Guide.*

Pluggable Authentication Modules (PAM)
pam_acct_mgmt(3), pam_authenticate(3), pam_chauthtok(3).

HP-UX Smart Card Login
scpin(1), scsync(1).

NAME

logname - get login name

SYNOPSIS

logname

DESCRIPTION

logname writes the user's login name to standard output. The login name is equivalent to that returned by **getlogin()** (see *getlogin*(3C)).

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which diagnostic messages are displayed.

FILES

/etc/profile

SEE ALSO

env(1), login(1), getlogin(3C), logname(3C), environ(5).

STANDARDS CONFORMANCE

logname: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

NAME

lorder - find ordering relation for an object library

SYNOPSIS

lorder [*files*]

DESCRIPTION

The input consists of one or more object or archive library *files* (see *ar*(1)) placed on the command line or read from standard input. The standard output is a list of pairs of object file names, meaning that the first file of the pair refers to external identifiers defined in the second. Output can be processed by **tsort** to find an ordering of a library suitable for one-pass access by **ld** (see *tsort*(1) and *ld*(1)). Note that the link editor **ld** is capable of multiple passes over an archive in the archive format and does not require that **lorder** be used when building an archive. Using the **lorder** command may, however, allow for a slightly more efficient access of the archive during the link edit process.

The symbol table maintained by **ar** allows **ld** to randomly access symbols and files in the archive, making the use of **lorder** unnecessary when building archive libraries (see *ar*(1)).

EXTERNAL INFLUENCES

Environment Variables

The following internationalization variables affect the execution of **lorder**:

LANG
Determines the locale category for native language, local customs and coded character set in the absence of **LC_ALL** and other **LC_*** environment variables. If **LANG** is not specified or is set to the empty string, a default of **C** (see *lang*(5)) is used instead of **LANG**.

LC_ALL
Determines the values for all locale categories and has precedence over **LANG** and other **LC_*** environment variables.

LC_COLLATE
Determines the locale category for character collation.

LC_CTYPE
Determines the locale category for character handling functions.

LC_MESSAGES
Determines the locale that should be used to affect the format and contents of diagnostic messages written to standard error.

LC_NUMERIC
Determines the locale category for numeric formatting.

NLSPATH
Determines the location of message catalogues for the processing of **LC_MESSAGES**.

If any internationalization variable contains an invalid setting, **lorder** behaves as if all internationalization variables are set to **C**. See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported.

EXAMPLES

Build a new library from existing **.o** files:

```
ar cr library `lorder *.o | tsort`
```

When creating libraries with so many objects that the shell cannot properly handle the ***.o** expansion, the following technique may prove useful:

```
ls | grep '.o$' | lorder | tsort | xargs ar cq library
```

WARNINGS

Object files whose names do not end with **.o** are overlooked, even when contained in library archives. Their global symbols and references are attributed to some other file.

FILES

`/var/tmp/*symref`	temporary files
`/var/tmp/*symdef`	

SEE ALSO

System Tools:

ar(1)	create archived libraries
ld(1)	invoke the link editor

Miscellaneous:

tsort(1)	produce an ordered list of items (topological sort)

STANDARDS CONFORMANCE

`lorder`: SVID2, SVID3, XPG2, XPG4

NAME

lp, lpalt, cancel - print/alter/cancel requests on an LP printer or plotter

SYNOPSIS

lp [**-c**] [**-d***dest*] [**-m**] [**-n***number*] [**-o***option*] [**-p***priority*] [**-s**] [**-t***title*] [**-w**] [*file* ...]

lpalt *id* [**-d***dest*] [**-i**] [**-m**] [**-n***number*] [**-o***option*] [**-p***priority*] [**-s**] [**-t***title*] [**-w**]

cancel [*id* ...] [*printer* ...] [**-a**] [**-e**] [**-i**] [**-u***user*]

DESCRIPTION

The **lp** command queues files for printing. The **lpalt** command changes information in a queued request. The **cancel** command deletes a queued request.

lp Command

The **lp** command arranges for the named files, *file* ..., and associated information (collectively called a *request*) to be queued for output to a printer or plotter in the LP (line printer) subsystem. The process is called printing, regardless of the actual output device.

lp associates a unique identifier with each request and writes it to standard output, using the message:

request id is *dest***-***sequence* **(***fileinfo***)**

The request ID is *dest***-***sequence*, which can be used later to alter, cancel, or find the status of the request (see **lpalt** and **cancel** below, and *lpstat*(1)).

For example, in the following message,

```
request id is pr47lf8e-2410 (1 file)
```

the request ID is **pr47lf8e-2410**.

1

lp Options and Arguments

lp recognizes the following options and arguments. The keyletter options can be specified in any order. The *file* ... names must be last. Blanks are not permitted between a keyletter and its argument.

file ... Print each named file. If no file names are specified, standard input is assumed. The hyphen symbol (**-**) also specifies standard input and can be intermixed on the command line with file names. Files are printed in the same order in which they are specified.

-c Copy the named files to LP subsystem spooling directories.

Normally, the files are linked into a spool directory. The ownership and mode of the linked files remain unchanged. If the **-c** option is given, or linking is not possible (perhaps because the printer is not physically attached to the local system or cluster), the files are copied into the spool directories. The ownership and mode of the copies are set to allow read access to owner **lp** and group **bin** only.

If the files are linked rather than copied, any changes made to the named files after the request is made but before it is printed will be reflected in the printed output. Standard input is always copied instead of linked.

-d*dest* Select *dest* as the printer or class of printers that is to do the printing. If *dest* is a printer, the request will be printed only on that specific printer. If *dest* is a class, the request will be printed on the first available printer that is a member of the class. Under certain conditions (printer unavailability, file space limitation, etc.), requests for a specific *dest* might not be accepted (see *accept*(1M) and *lpadmin*(1M)).

If the **-d** option is omitted, *dest* is taken from the environment variable **LPDEST**. If that variable is unset or empty, the default queue is used, if one has been defined. If there is no default queue, or **LPDEST** is set but invalid, **lp** issues an error message and the request is not queued. Printer and class names and the default queue are defined by your LP subsystem administrator (see *lpadmin*(1M) and *lpstat*(1)).

-m Send a mail message (see *mail*(1)). to the user after the request has been printed. By default, no mail is sent upon normal completion of the print request.

-n*number* Print *number* copies of the output. The default is 1.

-o*option* Specify a printer-dependent *option*. You can specify several printer options by repeating the **-o** option. For information about the options that are available for a printer supported on your system, see the interface script for the printer name in the **/etc/lp/interface** directory.

-p*priority* Set the priority of the print request. *priority* must be in the range 0 (lowest priority) to 7 (highest priority). The priority is used to select the next spooled file for the targeted printer or class of printers. If the priority is less than the *fence*, the minimum priority set for the printer, the print request is deferred until the fence is lowered or the priority is raised. The default for a printer queue is the default priority set by the **lpadmin** or **lpfence** command (see *lpadmin*(1M) and *lpsched*(1M)). The default for a class queue is the highest default priority among printers in the class.

-s Suppress standard output messages from **lp** such as "**request id is** ...". Error messages are still displayed on standard error.

-t*title* Print *title* on the banner page of the output.

-w Write a message to the user's terminal after the request has been printed. If the user is not logged in or (for remote printing) if **rlpdaemon** (see *rlpdaemon*(1M)) is not running on the user's local system, mail will be sent instead.

lpalt Command

The **lpalt** command alters a request made by a previous **lp** command, if it is not currently printing. (To requeue a currently printing request, use the **disable** command (see *enable*(1)) to stop the printer.)

lpalt Options

lpalt recognizes the following options and arguments, which can be specified in any order. Blanks are not permitted between a keyletter and its argument.

id Specifies the request to be altered. *id* is a request ID returned by **lp** or **lpalt**.

-d*dest* Requeue the request to the named printer or class *dest*. A new unique request ID is written to standard output.

-i Alter only local requests.

-m Send mail upon normal completion of the print request.

-n*number* Change the number of copies to *number*.

-o*option* Specify a printer-dependent *option*. You can specify several printer options by repeating the **-o** option. All **-o** options from previous **lp** and **lpalt** commands for this request ID are deleted.

-p*priority* Change the request's priority to *priority*.

-s Suppress standard output messages from **lpalt** such as "**request id is** ...". Error messages are still displayed on standard error.

-t*title* Change the *title* on the banner page of the output.

-w Write a message to the user's terminal after the request has been printed. If the user is not logged in or (for remote printing) if **rlpdaemon** (see *rlpdaemon*(1M)). is not running on the user's local system, mail will be sent instead.

cancel Command

The **cancel** command cancels requests that were made with the **lp** command, even if they are currently printing. At least one *id* or *printer* must be specified.

The cancellation of a request that is currently printing frees the printer to print its next available request.

cancel Options and Arguments

cancel recognizes the following options and arguments, which can be specified in any order. Blanks are not permitted between a keyletter and its argument.

id ... Specifies one or more requests. *id* is a request ID returned by **lp** or **lpalt**.

printer ... Specifies one or more printers. *printer* is the name of a printer, not a class. Either cancel the request that is currently printing on each *printer*, or, if an **-a**, **-e**, or **-u** option is specified, specify the printer on which to perform the corresponding operation.

-a	Remove all requests the user owns on each *printer*. The owner is determined by the user's login name and the host name of the machine where the **lp** command was invoked.
-e	Empty the spool queue of all requests for each *printer*. Only users with appropriate privileges can use this option.
-i	Cancel only local requests.
-u*user*	Remove any requests belonging to *user*. You can repeat the **-u** option to specify more users. Only users with appropriate privileges can use this option.

Printing Overview

A printer can print requests from one or two destination queues: its own private queue and an optional class queue, which can serve one or more printers. The destination queues are set up with the **lpadmin** command. The **lp** command places a printing request into a printer or class destination queue as directed by a user. The **lpsched** scheduler directs the requests from the destination queues to the printers. The **accept** and **reject** commands control whether **lp** can place requests in the destination queues. The **enable** and **disable** commands control whether **lpsched** can send a queued request to a printer. If a printer has two queues and one queue is rejecting requests, users can still direct requests to the other destination queue and have the requests printed. **lpstat** reports the current status of the destination queues and the scheduler. See *enable*(1), *lpstat*(1), *accept*(1M), and *lpadmin*(1M).

EXTERNAL INFLUENCES

Environment Variables

LANG determines the locale to use for the locale categories when both **LC_ALL** and the corresponding environment variable (beginning with **LC_**) do not specify a locale. If **LANG** is not set or is set to the empty string, a default of "C" (see *lang*(5)) is used.

LC_ALL determines the locale to use to override any values for locale categories specified by the setting of **LANG** or any environment variables beginning with **LC_**.

LC_CTYPE determines the locale for interpretation of sequences of bytes of text data as characters (e.g., single- verses multibyte characters in arguments and input files).

LC_MESSAGES determines the language in which messages are displayed.

LPDEST determines the output device or destination. If the **LPDEST** environment variable is not set, the **PRINTER** environment variable is used. The **-d** *dest* option takes precedence over **LPDEST**. Results are undefined when **-d** is not specified and **LPDEST** contains a value that is not a valid device or destination name.

PRINTER determines the output device or destination. If the **LPDEST** and **PRINTER** environment variables are not set, an unspecified output device is used. The **-d** *dest* option and the **LPDEST** environment variable takes precedence over **PRINTER**. Results are undefined when **-d** is not specified, **LPDEST** is unset, and **PRINTER** contains a value that is not a valid device or destination name.

If any internationalization variable contains an invalid setting, the commands behave as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multibyte character code sets are supported.

RETURN VALUE

Exit values are:

0	Successful completion.
>0	Error condition occurred.

EXAMPLES

For a HP2934A printer named **lp2**, configured with an interface script that defines the **-c** option to cause the printer to print in a compressed mode, use the following command to print **myfile** with compressed print on **lp2**:

```
lp -dlp2 -oc myfile
```

lp can be used at the end of a pipeline to print the results of a previous command. It is commonly used with the **pr** command (see *pr*(1)) to print formatted output. For a default printer, to format file **.profile** into pages and print three copies of it:

1

```
pr .profile | lp -n3
```

WARNINGS

A remote print request can be altered or canceled only by the user who requested it, and only from the system from which the the original **lp** command was issued.

If the restrict cancel feature (see *lpadmin*(1M)) is enabled for the specified printer, a user can only alter or cancel requests owned by the user.

For a remote system, **lpalt** cannot change *dest* and *priority*.

FILES

/etc/lp	Directory of spooler configuration data
/etc/lp/interface	Directory of active LP device interface scripts
/usr/lib/lp	Directory of model and font file directories
/var/adm/lp	Directory of spooler log files
/var/spool/lp	Directory of LP spooling files and directories

SEE ALSO

enable(1), lpstat(1), mail(1), slp(1), accept(1M), lpadmin(1M), lpana(1M), lpsched(1M), rcancel(1M), rlp(1M), rlpdaemon(1M), rlpstat(1M).

STANDARDS CONFORMANCE

lp: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

cancel: SVID2, SVID3, XPG4

NAME

lpfilter, divpage, fontdl, lprpp, plotdvr, printstat, reverse - filters invoked by lp interface scripts

SYNOPSIS

/usr/lbin/divpage [**-p** | **-l**] [**-h** | **-q**] [**-n***FontID*] *filename*

/usr/lbin/fontdl [**-n***FontID*] [**-l**] [**-p**] *filename*

/usr/lbin/lprpp [**-i**] [**-o**] [**-e**] [**-l***nn*] [**-n**] [**-p**]

/usr/lbin/plotdvr **-l***request_id* **-u***username* [**-e**] [**-f**] [**-i**] *filename*

/usr/sbin/printstat **-l***request_id* **-u***username* *filename*

/usr/sbin/reverse [**-l** *page_length*]

Remarks

The structure of these filters is currently under review. They may become obsolete or be restructured in a future release.

DESCRIPTION

Various filters are used by the **lp** subsystem to obtain specialized behavior for specific types of devices or data. This entry describes currently supported filters.

A number of these filters use a specified *username* and *filename* to determine the location of the user who originated the print message.

The *filename* is used to determine the *hostname* of the system where the request originated, and must have the form [*dirname*]**/d?A***nnnhostname* or [*dirname*]**/dA***nnnnhostname*, where *dirname* is not a path name, but only the name of the basename's parent directory. *filename* meets this requirement when it is set to **$6** in the interface script for the printer.

divpage

Provides capabilities for printing multiple pages per sheet and selection of built-in fonts.

Options:

-p	Set mode to portrait (default).
-l	Set mode to landscape.
-h	Print half pages (default).
-q	Print quarter pages.
-n*FontID*	Use font number *FontID*. Default is 0. Causes the string **^{E}C(** *FontID***X** to be sent to the printer.

fontdl

fontdl downloads the font contained in *filename* to a printer connected to standard output.

Options:

-n*FontID*	Specifies the ID number by which the font will be referenced. Default is 0.
-l	Specify landscape mode. Default is portrait.
-p	Specifies proportional spacing. Default is fixed.

lprpp

This is a filter that converts backspace overstrike to line overprint with horizontal print positioning to enhance bold print. This functionality is required on printers such as the LaserJet, which cannot produce bold print by overstriking.

Options:

-i	Converts <ANYCHAR> to <ANYCHAR><BACKSPACE>_ to italicize ANYCHAR. Also properly italicizes overstruck (bold) characters. Does *not* work correctly for "hashed-overstrike" such as: <ANYCHAR><BACKSPACE><DIFFERENTCHAR><BACKSPACE>_

1

-o	Prints only the odd numbered pages. Used with **-e** for double-sided printing.
-e	Print only the even numbered pages. Used with **-o** for double sided printing.
-l*nn*	Specifies the page length, in lines. Default is 60 unless *-n* or *-p* is selected, in which case it is 66.
-n	Specifies nroff mode for printing output of the *nroff* command. Prints 66 lines per page with the first line appearing on logical line 4 of the printer.
-p	Specifies pr mode for printing output from the *pr* command. Prints 66 lines per page with the first line appearing on logical line 3 of the printer.

plotdvr

HP-GL plotter filter. This filter scans the data for "PG" commands (paper feed). When this data is encountered, the filter strips it from the data stream and informs the requesting user of the need to change paper in the plotter.

Options:

-l*request_id*	Specifies the printer request ID and is used in various messages regarding the plot request.
-u*username*	The requesting user's login name, used to communicate with the user regarding the request.
-e	Specifies the use of escape sequences, rather than HP-GL commands, to determine plotter status.
-f	Plot without stopping for paper changes. The "PG" commands are not stripped from the data stream and the user is not notified of them. This option is used on plotters capable of automatic page feed.
-i	Prevents initialization of the plotter.

l

printstat

Interrogates an RS232 printer as to its status, and does not return until the printer is ready. If the printer is *off-line*, *out of paper*, or *disconnected*, the submitter of the print request is notified of this condition periodically until it is corrected. When the printer is ready to print, the command exits.

Standard input and standard output must both be connected to the serial printer device.

This program uses the *send-status* command **^{E}C?^{D}1?** to determine status. Not all serial printers respond to this command. Only the following configurations support this command:

Printer	Comments
LaserJet	Not supported
LaserJetII	Supported
LaserJetIID	Requires HP 26013A module
LaserJetIIP	Not supported
LaserJetIII	Requires HP 26013A module
LaserJet2000	Not supported

Options:

-l*request-id*	Print request ID used in various communications with the user.
-u*username*	The requesting user's login name, used to communicate with the user regarding the request.

reverse

Prints the data appearing on the standard input in reverse page order to the standard output. This command can handle up to 2000 pages.

Options:

-l*page_length*

Specifies the page length, in lines. Default is 66.

DIAGNOSTICS

Error and diagnostic messages appear on the printed output, on the user's terminal, or are mailed to the user, depending on circumstances.

WARNINGS

There is little consistency in the interface to these filters.

SEE ALSO

lp(1), lpadmin(1).

Managing Systems and Workgroups.

NAME

lpstat - report line printer status information

SYNOPSIS

lpstat [**-drst**] [**-a**[*list*]] [**-c**[*list*]] [**-o**[*list*]] [**-p**[*list*]] [**-u**[*list*]] [**-v**[*list*]] [**ID**...]

DESCRIPTION

The **lpstat** utility writes to standard output information about the current status of the line printer system.

If no arguments are given, **lpstat** writes the status of all requests made to **lp** by the user that are still in the output queue.

OPTIONS

The **lpstat** utility supports the XBD specification, Section 10.2, Utility Syntax Guidelines, except the option-arguments are optional and cannot be presented as separate arguments.

Some of the options below can be followed by an optional list that can be in one of two forms: a list of items separated from one another by a comma, or a quoted list of items separated from one another by a comma or one or more blank characters, or combinations of both. See EXAMPLES.

The omission of a list following such options causes all information relevant to the option to be written to standard output; for example:

```
lpstat -o
```

writes the status of all output requests that are still in the output queue.

-a[*list*] Write the acceptance status of destinations for output requests. The *list* argument is a list of intermixed printer names and class names.

-c[*list*] Write the class names and their members. The *list* argument is a list of class names.

-d Write the system default destination for output requests.

-o[*list*] Write the status of output requests. The *list* argument is a list of intermixed printer names, class names and request IDs.

-p[*list*] Write the status of printers. The *list* argument is a list of printer names.

-r Write the status of the line printer request scheduler.

-s Write a status summary, including the status of the line printer scheduler, the system default destination, a list of class names and their members and a list of printers and their associated devices.

-t Write all status information.

-u[*list*] Write the status of output requests for users. The *list* argument is a list of login names.

-v[*list*] Write the names of printers and the pathnames of the devices associated with them. The list argument is a list of printer names.

OPERANDS

The following operand is supported:

ID A request ID, as returned by **lp**.

STDIN

Not used.

INPUT FILES

None.

ENVIRONMENT VARIABLES

The following environment variables affect the execution of **lpstat**:

LANG Provide a default value for the internationalisation variables that are unset or null. If **LANG** is unset or null, the corresponding value from the implementation-specific default locale will be used. If any of the internationalisation variables contains an invalid setting, the utility will behave as

	if none of the variables had been defined.
LC_ALL	If set to a non-empty string value, override the values of all the other internationalisation variables.
LC_CTYPE	Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single- as opposed to multi-byte characters in arguments).
LC_MESSAGES	Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error, and informative messages written to standard output.
LC_TIME	Determine the format of date and time strings output when displaying line printer status information with the **-a**, **-o**, **-p**, **-t**, or **-u** options.
NLSPATH	Determine the location of message catalogues for the processing of **LC_MESSAGES.**
TZ	Determine the timezone used with date and time strings.

ASYNCHRONOUS EVENTS
Default.

STDOUT
The standard output is a text file containing the information described in OPTIONS, in an unspecified format.

STDERR
Used only for diagnostic messages.

OUTPUT FILES
None.

EXTENDED DESCRIPTION
None.

EXIT STATUS
The following exit values are returned:

0	Successful completion.
>0	An error occurred.

CONSEQUENCES OF ERRORS
Default.

APPLICATION USAGE
The **lpstat** utility cannot reliably determine the status of print requests in all conceivable circumstances. When the printer is under the control of another operating system or resides on a remote system across a network, it need not be possible to determine the status of the print job after it has left the control of the local operating system. Even on local printers, spooling hardware in the printer may make it appear that the print job has been completed long before the final page is printed.

EXAMPLES
1. Obtain the status of two printers, the pathnames of two printers, a list of all class names and the status of the request named HiPri-33:

```
lpstat -plaser1,laser4 -v"laser2 laser3" -c HiPri-33
```

2. Obtain user print job status using the obsolescent mixed blank and comma form:

```
lpstat -u"ddg,gmv, maw"
```

FUTURE DIRECTIONS
A version of **lpstat** that fully supports the XBD specification, Section 10.2, Utility Syntax Guidelines may be introduced in a future issue.

1

SEE ALSO

cancel(1), lp(1).

CHANGE HISTORY

First released in Issue 2.

Issue 3

The operation of this utility in an 8-bit transparent manner has been noted.

The operation of this utility in an internationalised environment has been described.

Issue 4

Format reorganised.

Exceptions to Utility Syntax Guidelines conformance noted.

Internationalised environment variable support mandated.

STANDARDS CONFORMANCE

`lpstat`: SVID2, SVID3, XPG2, XPG3, XPG4

HP-UX EXTENSIONS

DESCRIPTION

Any arguments that are not *options* are assumed to be request *ids* (as returned by **lp**). **lpstat** prints the status of such requests. *options* can appear in any order and can be repeated and intermixed with other arguments.

-i	Inhibit the reporting of remote status.
-o[*list*]	Also see the **-i** option.
-t	Print all status information. Same as specifying **-r**, **-s**, **-a**, **-p**, **-o**. See the **-i** option.

Security Restriction

Only users who have the **lp** subsystem authorization or the **printqueue** secondary subsystem authorization can view the entire queue. Unauthorized users can view only their own jobs whose sensitivity levels are dominated by the user's current sensitivity level.

The **allowmacaccess** privilege allows viewing jobs at higher sensitivity levels.

EXAMPLES

Check whether your job is queued:

```
lpstat
```

Check the relative position of a queued job:

```
lpstat -t
```

Verify that the job scheduler is running:

```
lpstat -r
```

FILES

/var/spool/lp/*
/var/adm/lp/*
/etc/lp/*
/usr/lib/lp/*

SEE ALSO

enable(1), lp(1), rlpstat(1M).

STANDARDS CONFORMANCE

lpstat: SVID2, SVID3, XPG2, XPG3, XPG4

l

NAME

ls, lc, l, ll, lsf, lsr, lsx - list contents of directories

SYNOPSIS

ls [**-abcdefgilmnopqrstuxACFLR1**] [*names*]

lc [**-abcdefgilmnopqrstuxACFLR1**] [*names*]

l [*ls_options*] [*names*]
ll [*ls_options*] [*names*]
lsf [*ls_options*] [*names*]
lsr [*ls_options*] [*names*]
lsx [*ls_options*] [*names*]

DESCRIPTION

For each directory argument, the **ls** command lists the contents of the directory. For each file argument, **ls** repeats its name and any other information requested. The output is sorted in ascending collation order by default (see Environment Variables below). When no argument is given, the current directory is listed. When several arguments are given, the arguments are first sorted appropriately, but file arguments appear before directories and their contents.

If you are a user with appropriate privileges, all files except **.** and **..** are listed by default.

There are three major listing formats. The format chosen depends on whether the output is going to a login device (determined by whether output device file is a **tty** device), and can also be controlled by option flags. The default format for a login device is to list the contents of directories in multicolumn format, with entries sorted vertically by column. (When individual file names (as opposed to directory names) appear in the argument list, those file names are always sorted across the page rather than down the page in columns because individual file names can be arbitrarily long.) If the standard output is not a login device, the default format is to list one entry per line. The **-C** and **-x** options enable multicolumn formats, and the **-m** option enables stream output format in which files are listed across the page, separated by commas. In order to determine output formats for the **-C**, **-x**, and **-m** options, **ls** uses an environment variable, **COLUMNS**, to determine the number of character positions available on each output line. If this variable is not set, the **terminfo** database is used to determine the number of columns, based on the environment variable **TERM**. If this information cannot be obtained, 80 columns is assumed.

The command **lc** functions the same as **ls** except that the **lc** default output is columnar, even if output is redirected.

Options

ls recognizes the following options:

-a List all entries; usually entries whose names begin with a period (**.**) are not listed.

-b List nonprinting characters in the octal **\***ddd* notation.

-c Use time of last modification of the inode (file created, mode changed, etc.) for sorting (**-t**) or printing (**-l** (ell)).

-d If an argument is a directory, list only its name (not its contents); often used with **-l** (ell) to get the status of a directory.

-e List the extent attributes of the file. If any of the files has a extent attribute, this option lists the extent size, space reserved and allocation flags. This option must be used with the **-l** (ell) option.

-f Interpret each argument as a directory and list the name found in each slot. This option disables **-l** (ell), **-r**, **-s**, and **-t**, and enables **-a**; the order is the order in which entries appear in the directory.

-g Same as **-l** (ell), except that only the group is printed (owner is omitted). If both **-l** (ell) and **-g** are specified, the owner is not printed.

-i For each file, list the inode number in the first column of the report. When used in multicolumn output, the number precedes the file name in each column.

-l (ell) List in long format, giving mode, number of links, owner, group, size in bytes, and time of last modification for each file (see further DESCRIPTION and Access Control Lists below). If the time of last modification is greater than six months ago, or any time in the future, the year is substituted for the hour and minute of the modification time. If the file is a special file, the size field contains the major and minor device numbers rather than a size. If the file is a symbolic link, the filename is printed, followed by **->** and the pathname of the referenced file.

-m Stream output format.

-n The same as **-l**, (ell) except that the owner's UID and group's GID numbers are printed, rather than the associated character strings.

-o The same as **-l**, (ell) except that only the owner is printed (group is omitted). (If both **-l** (ell) and **-o** are specified, the group is not printed).

-p Put a slash (**/**) after each file name if that file is a directory.

-q List nonprinting characters in file names as the character (**?**).

-r Reverse the order of sort to get reverse (descending) collation or oldest first, as appropriate.

-s List size in blocks, including indirect blocks, for each entry. The first entry listed is the total number of blocks in the directory. When used in multicolumn output, the number of blocks precedes the file name in each column. The number of indirect blocks in a file is filesystem dependent.

-t Sort by time modified (latest first) before sorting alphabetically.

-u Use time of last access instead of last modification for sorting (**-t** option) or printing (**-l** (ell) option).

-x List multicolumn output with entries sorted across rather than down the page.

-A The same as **-a**, except that the current directory **.** and parent directory **..** are not listed. For a user with appropriate privileges, this flag defaults to on, and is turned off by **-A**.

-C List multicolumn output with entries sorted down the columns.

-F After each file name, put one of:

- A slash (**/**) if the file is a directory or a symbolic link to a directory.
- An asterisk (*****) if the file is executable;
- An at-sign (**@**) if the file is a symbolic link to a file;
- A vertical bar (**|**) if the file is a fifo.

-L If the argument is a symbolic link, list the file or directory to which the link refers rather than the link itself.

-R Recursively list subdirectories encountered.

-1 (one) List the file names in single column format regardless of the output device. This forces single column format to the user's terminal.

Specifying more than one of the options in the following mutually exclusive pairs is not considered an error: **-C** and **-l** (ell), **-m** and **-l** (ell), **-x** and **-l** (ell), **-C** and **-1** (one), and **-c** and **-u**.

ls is known by several shorthand-version names for the various formats:

l	is equivalent to **ls -m**
ll	is equivalent to **ls -l** (ell)
lsf	is equivalent to **ls -F**
lsr	is equivalent to **ls -R**
lsx	is equivalent to **ls -x**

The shorthand notations are implemented as links to **ls**. Option arguments to the shorthand versions behave exactly as if the long form above had been used with the additional arguments.

Mode Bits Interpretation (-l option)

The mode printed in listings produced by the **-l** (ell) option consists of 10 characters, for example, **-rwxr-xr-x**.

The first character indicates the entry type:

b	Block special file
c	Character special file
d	Directory

1

`l`	Symbolic link
`n`	Network special file
`p`	Fifo (also called a "named pipe") special file
`s`	Socket
`-`	Ordinary file

The next 9 characters are interpreted as three sets of three characters each which identify access and execution permissions for the owner, group, and others categories, as described in *chmod*(1). The **-** indicates the permission is not granted. The various permissions can be put together in any combination, except that the **x**, **s**, **S**, **t**, and **T** characters are mutually exclusive, as implied below.

`-r--------`	Read by owner
`--w-------`	Write by owner
`---x------`	Execute (or search directory) by owner; do not set user ID on execution
`---s------`	Execute/search by owner; set user ID on execution
`---S------`	No execute/search by owner; set user ID on execution
`----r-----`	Read by group
`-----w----`	Write by group
`------x---`	Execute/search by group; do not set group ID on execution
`------s---`	Execute/search by group; set group ID on execution
`------S---`	No execute/search by group; set group ID on execution
`-------r--`	Read by others
`--------w-`	Write by others
`---------x`	Execute/search by others; do not set sticky bit on execution
`---------t`	Execute/search by others; set sticky bit on execution
`---------T`	No execute/search by others; set sticky bit on execution

The mode characters are interpreted as follows:

`-`	Deny all permissions in the corresponding position.
`r`	Grant read permission to the corresponding user class.
`w`	Grant write permission to the corresponding user class.
`x`	Grant execute (or search in directory) permission to the corresponding user class.
`s`	Grant execute (search) permission to the corresponding user class. Execute the file as if by the owner (set user ID, SUID) or group (set group ID, SGID), as indicated by position.
`S`	Deny execute (search) permission to the corresponding user class. Execute the file as if by the owner (set user ID, SUID) or group (set group ID, SGID), as indicated by position.
`t`	Grant execute (search) permission to others. The "sticky" (save text image) bit is set (see the description of **S_ISVTX** in *chmod*(2)).
`T`	Deny execute (search directory) permission to others. The "sticky" (save text image) bit is set.

When an option is specified that results in a listing of directory and/or file sizes in bytes or blocks (such as the **-s** or **-l** (ell) option), a total count of blocks, including indirect blocks, is also printed at the beginning of the listing.

Access Control Lists (ACLs)

If a file has optional ACL entries, the **-l** (ell) option displays a plus sign (**+**) after the file's permissions. The permissions shown are a summary representation of the file's access control list, as returned by **stat()** in the **st_mode** field (see *stat*(2)). To list the contents of an access control list, use the **lsacl** command (see *lsacl*(1) and *acl*(5)).

EXTERNAL INFLUENCES

Environment Variables

If the **COLUMNS** variable is set, **ls** uses the width provided in determining positioning of columnar output.

LANG determines the locale to use for the locale categories when both **LC_ALL** and the corresponding environment variable (beginning with **LC_**) do not specify a locale. If **LANG** is not set or is null, it defaults to **C** (see *lang*(5)).

LC_COLLATE determines the order in which the output is sorted.

LC_CTYPE determines which characters are classified as nonprinting for the **-b** and **-q** options, and the interpretation of single- and/or multibyte characters within file names.

LC_TIME determines the date and time strings output by the **-g**, **-l** (ell), **-n**, and **-o** options.

LC_MESSAGES determines the language in which messages (other than the date and time strings) are displayed.

If any internationalization variable contains an invalid setting, they all default to **C** (see *environ*(5)).

International Code Set Support

Single- and multibyte character code sets are supported.

RETURN VALUE

ls exits with one of the following values:

0 All input files were listed successfully.

>0 **ls** was aborted because errors occurred when accessing files. The following conditions cause an error:

- Specified file not found.
- User has no permission to read the directory.
- Process could not get enough memory.
- Invalid option specified.

EXAMPLES

Print a long listing of all the files in the current working directory (including the file sizes). List the most recently modified (youngest) file first, followed by the next older file, and so forth, to the oldest. Files whose names begin with a **.** are also printed.

```
ls -alst
```

WARNINGS

Setting options based on whether the output is a login (*tty*) device is undesirable because **ls -s** is very different from **ls -s | lp**. On the other hand, not using this setting makes old shell scripts that used **ls** almost inevitably fail.

Nonprinting characters in file names (without the **-b** or **-q** option) may cause columnar output to be misaligned.

DEPENDENCIES

NFS

The **-l** (ell) option does not display a plus sign (**+**) after the access permission bits of networked files to represent existence of optional access control list entries.

AUTHOR

ls was developed by AT&T, the University of California, Berkeley and HP.

FILES

/etc/group	For group IDs for **-l** (ell) and **-g**.
/etc/passwd	For user IDs for **-l** (ell) and **-o**.
/usr/share/lib/terminfo/?/*	For terminal information.

SEE ALSO

chmod(1), find(1), lsacl(1), stat(2), acl(5).

STANDARDS CONFORMANCE

ls: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

NAME

lsacl - list access control lists (ACLs) of files

SYNOPSIS

/usr/bin/lsacl [**-l**] *file*...

DESCRIPTION

lsacl lists access control lists (ACLs) of one or more files in symbolic, "short" form, one file's ACL per line of output, followed by the file name; see *acl*(5) for ACL syntax.

Options

lsacl recognizes the following option:

-l Print ACLs in long form. Each file's ACL can be more than one line long, and is always preceded by file name, colon, and newline. Consecutive file names are separated by blank lines.

If a hyphen (**-**) is given as a file name argument, **lsacl** prints the ACL for the standard input. By default, it prints the file name as **-**. For the **-l** option it prints a file name of **<stdin>**.

Unlike **ls**, **lsacl** cannot list ACLs of files in subdirectories automatically or list the ACL entries of the files in the current directory by default. A good way to do the latter is:

```
lsacl *
```

or

```
lsacl .* *
```

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages are displayed.

If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**. If any internationalization variable contains an invalid setting, **lsacl** behaves as if all internationalization variables are set to "C". See *environ*(5).

RETURN VALUE

If **lsacl** succeeds, it returns zero. If invoked incorrectly, it returns a value of **1**. If **lsacl** is unable to read the ACL for a specified file, it prints an error message to standard error, continues, and later returns a value of **2**.

EXAMPLES

List the ACL for the file **dir/file1**:

```
lsacl dir/file1
```

List ACLs for all files in the current directory, in long form.

```
lsacl -l .* *
```

List ACLs for all files under **mydir**:

```
find mydir -print | sort | xargs lsacl
```

DEPENDENCIES

lsacl will fail when the target file resides on a file system which does not support ACLs.

NFS:

lsacl is not supported on remote files.

AUTHOR

lsacl was developed by HP.

SEE ALSO

chacl(1), getaccess(1), ls(1), getacl(2), acl(5).

NAME

m4 - macro processor

SYNOPSIS

m4 [*options*] [*file* ...]

DESCRIPTION

m4 is a macro processor intended as a front end for Ratfor, C, and other languages. Each of the argument files is processed in order; if there are no files, or if a file name is **-**, standard input is read. The processed text is written to standard output.

Options

m4 recognizes the following options:

-e Operate interactively. Interrupts are ignored and the output is unbuffered. Using this mode may be very difficult.

-s Enable line sync output for the C preprocessor (**#**line ...)

-B*int* Change the size of the push-back and argument collection buffers from the default of 4,096.

-H*int* Change the size of the symbol table hash array from the default of 199. The size should be prime.

-S*int* Change the size of the call stack from the default of 100 slots. Macros take three slots, and nonmacro arguments take one.

-T*int* Change the size of the token buffer from the default of 512 bytes.

To be effective, the options listed above must appear before any file names and before any **-D** or **-U** options.

-D*name*[**=***val*]
Define *name* as *val* or as null if *val* is omitted.

-U*name* Undefine *name*.

Macro Calls

Macro calls have the form:

*name***(***arg1***,** *arg2***, ... ,***argn***)**

The left parenthesis (**(**) must immediately follow the name of the macro. If the name of a defined macro is not followed by a **(**, it is deemed to be a call of that macro with no arguments. Potential macro names consist of alphabetic letters, digits, and underscore (_); the first character cannot be a digit.

Leading unquoted blanks, tabs, and newlines are ignored while collecting arguments. Left and right single quotes (**‘** and **’**) are used to quote strings. The value of a quoted string is the string stripped of the quotes.

When a macro name is recognized, its arguments are collected by searching for a matching right parenthesis. If fewer arguments are supplied than are in the macro definition, the trailing arguments are taken to be null. Macro evaluation proceeds normally during the collection of the arguments, and any commas or right parentheses which happen to turn up within the value of a nested call are as effective as those in the original input text. After argument collection, the value of the macro is pushed back onto the input stream and rescanned.

Built-In Macro Names

m4 makes available the following built-in macros. They can be redefined, but, once this is done, the original meaning is lost. Their values are null unless otherwise stated.

changecom Change left and right comment markers from the default **#** and newline. With no arguments, the comment mechanism is effectively disabled. With one argument, the left marker becomes the argument and the right marker becomes newline. With two arguments, both markers are affected. Comment markers may be up to five characters long.

changequote Change quote symbols to the first and second arguments. The symbols may be up to five characters long. **changequote** without arguments restores the original values (i.e., **‘** and **’**).

decr	Returns the value of its argument decremented by 1.
define	The second argument is installed as the value of the macro whose name is the first argument. Each occurrence of **$***n* in the replacement text, where *n* is a digit, is replaced by the *n*th argument. Argument 0 is the name of the macro; missing arguments are replaced by the null string; **$#** is replaced by the number of arguments; **$*** is replaced by a list of all the arguments separated by commas; **$@** is equivalent to **$***, but each argument is quoted (with the current quotes).
defn	Returns the quoted definition of its arguments. It is useful for renaming macros, especially built-ins.
divert	**m4** maintains 10 output streams, numbered 0 to 9. The final output is the concatenation of the streams in numerical order; initially, stream 0 is the current stream. The **divert** macro changes the current output stream to its (digit-string) argument. Output diverted to a stream other than 0 through 9 is discarded.
divnum	Returns the value of the current output stream.
dnl	Reads and discards characters up to and including the next newline.
dumpdef	Prints current names and definitions, for the named items, or for all if no arguments are given.
errprint	Prints its argument on the diagnostic output file.
eval	Evaluates its argument as an arithmetic expression, using 32-bit arithmetic. Operators include **+**, **-**, *****, **/**, **%**, ****** (exponentiation), bitwise **&**, **|**, **^**, and **~**, relationals, and parentheses. Octal and hexadecimal numbers may be specified as in C. The second argument specifies the radix for the result; the default is 10. The third argument may be used to specify the minimum number of digits in the result.
hpux	Is a predefined object with a null value.
ifdef	If the first argument is defined, the value is the second argument; otherwise the third. If there is no third argument, the value is null. The word **unix** is predefined on HP-UX system versions of **m4**.
ifelse	Has three or more arguments. If the first argument is the same string as the second, then the value is the third argument. If not, and if there are more than four arguments, the process is repeated with arguments 4, 5, 6 and 7. Otherwise, the value is either the fourth string, or, if it is not present, null.
include	Returns the contents of the file named in the argument.
incr	Returns the value of its argument incremented by 1. The value of the argument is calculated by interpreting an initial digit-string as a decimal number.
index	Returns the position in its first argument where the second argument begins (zero origin), or −1 if the second argument does not occur.
len	Returns the number of characters in its argument.
m4exit	Causes immediate exit from **m4**. Argument 1, if given, is the exit code; the default is 0.
m4wrap	Argument 1 is pushed back at final EOF; for example: **m4wrap(‘cleanup()’)**
maketemp	Fills in a string of **XXXXX** in its argument with the current process ID.
popdef	Removes current definition of its arguments, exposing the previous one, if any.
pushdef	Similar to **define**, but saves any previous definition.
shift	Returns all but its first argument. The other arguments are quoted and pushed back with commas in between. The quoting nullifies the effect of the extra scan that will subsequently be performed.
sinclude	Identical to **include**, except that it says nothing if the file is inaccessible.
substr	Returns a substring of its first argument. The second argument is a zero-origin number selecting the first character; the third argument indicates the length of the substring. A missing third argument is taken to be large enough to extend to the end of the first string.

syscmd	Executes the HP-UX system command given in the first argument. No value is returned.
sysval	Is the return code from the last call to **syscmd**.
traceoff	Turns off trace globally and for any macros specified. Macros specifically traced by **traceon** can be untraced only by specific calls to **traceoff**.
traceon	With no arguments, turns on tracing for all macros (including built-ins). Otherwise, turns on tracing for named macros.
translit	Transliterates the characters in its first argument from the set given by the second argument to the set given by the third. No abbreviations are permitted.
undefine	Removes the definition of the macro named in its argument.
undivert	Causes immediate output of text from diversions named as arguments, or all diversions if no argument. Text may be undiverted into another diversion. Undiverting discards the diverted text.

(XPG4 only.) It is an error to specify an argument containing any non-numeric character for the built-in-macros: **decr**, **divert**, **incr**, **m4exit**, **substr**, **undivert**, and **eval**.

SEE ALSO

cc(1), cpp(1), ratfor(1).

STANDARDS CONFORMANCE

m4: SVID2, SVID3, XPG2, XPG3, XPG4

NAME

hp9000s200, hp9000s300, hp9000s500, hp9000s800, pdp11, u3b, u3b2, u3b5, u3b10, u370, vax - provide truth value about processor type

SYNOPSIS

```
hp9000s200
hp9000s300
hp9000s400
hp9000s500
hp9000s700
hp9000s800
hp-mc680x0
hp-pa
pdp11
u3b
u3b2
u3b5
u3b10
u370
vax
```

DESCRIPTION

The following commands return a true value (exit code 0) if the a processor type matches the command name. Otherwise a false value (exit code non-zero) is returned. These commands are commonly used within **make** makefiles and shell procedures to improve portability of applications (see *make*(1)).

Command	True for:	Command	True for:
hp9000s200	Series 200	**pdp11**	PDP-11/45 or PDP-11/70
hp9000s300	Series 300	**u3b**	3B20 computer
hp9000s400	Series 400	**u3b2**	3B2 computer
hp9000s500	Series 500	**u3b5**	3B5 computer
hp9000s700	Series 700	**u3b10**	3B10 computer
hp9000s800	Series 800/700	**u370**	IBM System/370 computer
hp-mc680x0	Series 200, 300, or 400	**vax**	VAX-11/750 or VAX-11/780
hp-pa	Series 700 or 800		

EXAMPLES

Given a shell script that must behave differently when run on an HP 9000 Series 700 or 800 system, select the correct code segment to be executed:

```
if hp9000s800
then
    # system is Series 700 or 800.
        if hp9000s700
        then

            # System is Series 700

                 Series 700 code fragment goes here
        else
            # System is Series 800

                 Series 800 code fragment goes here

        fi
fi
```

WARNINGS

hp9000s800 always returns true on both Series 800 and Series 700 systems. Therefore, when using this command in scripts to determine hardware type, always use both **hp9000s800** and **hp9000s700** in the appropriate sequence to ensure correct results (see EXAMPLES).

machid(1) will no longer provide support for future machines beyond the Series 800 and Series 700 systems. Decisions should be based on the hardware and software configuration information returned by *getconf*(1).

m

SEE ALSO

getconf(1), make(1), sh(1), test(1), true(1).

NAME

mail, rmail - send mail to users or read mail

SYNOPSIS

mail [**+**] [**-epqr**] [**-f** *file*]

mail [**-dt**] *person* ...

rmail [**-dt**] *person* ...

Remarks:

See *mailx*(1) and *elm*(1) for an enhanced user mail interface.

DESCRIPTION

The **mail** command, when used without arguments, prints the user's mail, message-by-message, in last-in, first-out order.

For each message, **mail** prints a **?** prompt and reads a line from the standard input to determine the disposition of the message. Commands that automatically proceed to the next message exit from **mail** if **mail** already on the last message.

Commands

mail supports the following commands:

<new-line>	Go on to next message. Exit if already on last message.
+	Same as <new-line>.
n	Same as <new-line>.
d	Delete message and go on to next message.
p	Print message again.
-	Go back to previous message.
s [*files*]	Save message in the named *files* (default is **mbox**), mark the message for deletion from the user's *mailfile*, and proceed to next message.
y [*files*]	Same as **s** [*files*].
w [*files*]	Save message without its header (the "From ..." line), in the named *files* (default is **mbox**), mark the message for deletion, and go on to next message.
m *person* ...	Mail the message to each named *person*, mark the message for deletion, and go on to next message.
q	Put undeleted mail back in the **mailfile** and stop.
EOT (Ctrl-D)	Same as **q**.
x	Abort. Leave original **mailfile** unchanged and stop.
!*command*	Escape to the command interpreter and execute *command*.
?	Print a command summary.
*****	Same as **?**.

Command-Line Options

The following command-line options alter printing of the mail:

+	Cause messages to be printed in first-in, first-out order.
-e	Suppresses printing of mail and returns the exit value: 0 = Mail present 1 = No mail 2 = Other error
-p	Prints all mail without prompting for disposition.
-q	Causes **mail** to terminate if an interrupt is received. Normally an interrupt only causes the termination of the printing of the current message.

m

-r	Same as **+**.
-f *file*	Causes **mail** to use *file* (for example, **mbox**) instead of the default *mailfile*.
-t	Causes the outbound message to be preceded by each *person* the mail is sent to. A *person* is usually a user name recognized by **login** (see *login*(1)). If a *person* being sent mail is not recognized, or if **mail** is interrupted during input, the file **dead.letter** will be saved to allow editing and resending. Note that **dead.letter** is regarded as a temporary file in that it is recreated every time needed, erasing the previous contents of **dead.letter**.
-d	Causes **mail** to deliver mail directly. This isolates **mail** from making routing decisions, and allows it to be used as a local delivery agent. Typically this option is used by auto-routing facilities when they deliver mail locally.

When *person*s are named, **mail** takes the standard input up to an end-of-file (or up to a line consisting of just a **.**) and adds it to each *person*'s *mailfile*. The message is preceded by the sender's name and a postmark.

To denote a recipient on a remote system, prefix *person* by the system name and exclamation mark (see *uucp*(1)). Everything after the first exclamation mark in *person* is interpreted by the remote system. In particular, if *person* contains additional exclamation marks, it can denote a sequence of machines through which the message is to be sent on the way to its ultimate destination. For example, specifying **a!b!cde** as a recipient's name causes the message to be sent to user **b!cde** on system **a**. System **a** then interprets that destination as a request to send the message to user **cde** on system **b**. This might be useful, for instance, if the sending system can access system **a** but not system **b**. **mail** does not use **uucp** if the remote system is the local system name (i.e., localsystem!user).

The **mailfile** can be manipulated in two ways to alter the function of **mail**. The *other* permissions of the file can be read-write, read-only, or neither read nor write to allow different levels of privacy. If changed to other than the default, the file is preserved, even when empty, to perpetuate the desired permissions. The file can also contain the first line:

Forward to *person*

which causes all mail sent to the owner of the **mailfile** to be forwarded to *person*. This is especially useful for forwarding all of a person's mail to a given machine in a multiple-machine environment. In order for forwarding to work properly the **mailfile** should have "mail" as group ID, and the group permission should be read-write.

rmail only permits the sending of mail. **uucp** uses **rmail** as a security precaution.

When a user logs in, the command **mail -e** can be used to detect the presence of mail, if any, and so indicate. When terminating, **mail** produces a notification message if new mail arrived while **mail** was running.

EXTERNAL INFLUENCES

Environment Variables

LC_TIME determines the format and contents of the displayed date and time strings.

If **LC_TIME** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**. If any internationalization variable contains an invalid setting, **mail** behaves as if all internationalization variables are set to "C". See *environ*(5).

When set, the **TMPDIR** environment variable specifies a directory to be used for temporary files, overriding the default directory **/tmp**.

International Code Set Support

Between HP-UX systems, single- and multi-byte character code sets are supported within mail text. Headers are restricted to characters from the 7-bit USASCII code set (see *ascii*(5)).

WARNINGS

Conditions sometimes result in a failure to remove a lock file.

After an interrupt, the next message may not be printed. To force printing, type a **p**.

Lines that look like postmarks in the message (that is, “From ...”) are preceded by **>**.

mail treats a line consisting solely of a dot (**.**) as the end of the message, except when the **rmail -d** command is used.

The maximum allowable line length in mail messages is **BUFSIZ** bytes as defined in **/usr/include/stdio.h**. If line length exceeds this limit, **mail** truncates the line starting at beginning-of-line, and uses only the trailing **BUFSIZ** characters.

Using two separate mail programs to access the same mail file simultaneously (usually inadvertently from two separate windows) can cause unpredictable results.

FILES

/var/mail/*.lock	Lock for mail directory
dead.letter	Unmailable text
/tmp/ma*	Temporary file
$MAIL	Variable containing path name of **mailfile**
$HOME/mbox	Saved mail
/etc/passwd	To identify sender and locate persons
/var/mail	Directory for incoming mail (mode **775**, group ID **mail**)
/var/mail/*user*	Incoming mail for *user*; that is, the **mailfile** (mode **660**, group ID **mail**)

SEE ALSO

login(1), mailx(1), uucp(1), write(1).

STANDARDS CONFORMANCE

mail: SVID2, SVID3, XPG2, XPG3

rmail: SVID2, SVID3

m

NAME

mailfrom - summarize mail folders by subject and sender

SYNOPSIS

mailfrom [**-hnQqStv**] [**-s** *status*] [*folder*|*username*]...

DESCRIPTION

The **mailfrom** command reads one or more mail folders and outputs one line per message in the form:

from [*subject*]

where *from* is the name of the person the message is from, and *subject* is the subject of the message, if present. If **mailfrom** determines that the message is from you, the *from* portion will read **To** *user*, where *user* is the user the message was sent to. This happens when you receive a copy of a letter you sent.

The default folder is your incoming mailbox, **/var/mail/***yourloginname*. See the Operands subsection below.

Options

mailfrom recognizes the following options:

-h Print a brief help message summarizing the options.

-n Number the messages using the same numbering scheme used by **readmail**.

-Q Very quiet mode. Only error messages are produced. This option is useful in shell scripts, where only the success or failure of the program is important, and output is not desired.

-q Quiet mode. Output only a one-line summary for each mailbox or folder.

-S Add a summary of the number of messages by message status in each mailbox or folder. To get the summary only, use this with the **-q** option. The summary has the form:

```
Folder contains:
  New messages: n
  Unread messages: u
  Read messages: r
```

If an item count, *n*, *r*, or *u* is zero, the line is omitted.

-s *status* Only display headers from messages with the given status. *status* can be one of **new**, **old**, **read**, or **unread**. **old** and **unread** are equivalent. The **-s** option can be repeated to print header information from more than one category, for example, only new and unread messages. The values can be abbreviated to their first letters. The default is all messages.

-t Tidy mode. If the *from* field is long enough to displace the *subject* field from its normal start column, move the subject down onto the next line.

-v Verbose mode. Print a descriptive header before listing the contents of each mailbox or folder.

Operands

mailfrom recognizes the following optional operands:

folder|*username*
A file name or the name of a mail user on your system. You can use the **=***filename* format to specify a folder in your mail directory, defined by the **maildir** string variable in your **elmrc** configuration file.

mailfrom searches for the value as a file name relative to your current directory. Then, if the file name is not an absolute path, it searches for the value relative to the incoming mailbox directory, **/var/mail**. The first file found is selected. You must have read access to the file.

EXIT STATUS

mailfrom returns the following values:

0	Messages matching *status* are present.
1	No messages matching *status* are present, but there are some messages.
2	There are no messages at all.
3	An error occurred.

If multiple mailboxes or folders are specified, the exit status only applies to the last one examined. This can be used in scripts to determine what kind of mail a user has.

EXAMPLES

Display header information from all the messages in your mailbox.

```
mailfrom
```

Display header information from all new messages in your mailbox.

```
mailfrom -s new
```

Assuming you have the proper file permissions to read **guest**'s mail, print out header information from all new and unread messages in **guest**'s incoming mailbox.

```
mailfrom -s new -s unread guest
```

Print only a summary of how many new, unread, and read messages are in your incoming mailbox.

```
mailfrom -q -S
```

FILES

$HOME/.elm/elmrc	Your **elm** configuration file.
/var/mail	Directory of incoming mailboxes.

AUTHOR

mailfrom was developed by HP.

SEE ALSO

elm(1), mail(1), mailx(1), readmail(1).

NAME

mailq - prints the mail queue

SYNOPSIS

mailq [**-v**]

DESCRIPTION

mailq prints a summary of the mail messages queued for future delivery.

The first line printed for each message shows the internal identifier used on this host for the message, the size of the message in bytes, the date and time the message was accepted into the queue, and the envelope sender of the message. The second line shows the error message that caused this message to be retained in the queue; it will not be present if the message is being processed for the first time. The following lines show message recipients, one per line.

mailq is identical to **sendmail -bp**.

The options are as follows:

Options

-v Print verbose information. This adds the priority of the message and a single character indicator ("+" or blank) indicating whether a warning message has been sent on the first line of the message. Additionally, extra lines may be intermixed with the recipients indicating the "controlling user" information; this shows who will own any programs that are executed on behalf of this message and the name of the alias this command expanded from, if any.

RETURN VALUE

The **mailq** utility exits 0 on success, and >0 if an error occurs.

SEE ALSO

sendmail(1M).

HISTORY

The **mailq** command appeared in 4.0BSD.

NAME
mailstats - print mail traffic statistics

SYNOPSIS
mailstats

DESCRIPTION
mailstats reads and interprets the **sendmail** statistics file, **/etc/mail/sendmail.st**, then prints out the mail traffic statistics. If **/etc/mail/sendmail.st** exists, **sendmail** collects statistics about your mail traffic and stores them in the statistics file. This file does not grow.

Statistics are gathered on a per-mailer basis for each mailer defined in the **sendmail** configuration file. Statistics are kept on the number of messages and the number of bytes for all inbound and outbound traffic.

To show the definitions of the mailers (delivery agents) reported on by mailstats, use the command:

```
grep '^M' /etc/mail/sendmail.cf
```

The first one listed is mailer 0, the second is mailer 1, etc.

To clear the statistics file, execute, as root:

```
cp /dev/null /etc/mail/sendmail.st
```

The statistics file is then invalid until some mail is actually transferred.

DIAGNOSTICS
mailstats generates error messages if the statistics file is not accessible or if the size of the statistics file has changed. Error messages are:

mailstats: file size changed
Either **/etc/mail/sendmail.st** is zero length, meaning that no mail has been transferred since it was cleared out, or its size has changed. Since the size of this file is supposed to remain constant, any change in size means that the file is invalid.

mailstats: /etc/mail/sendmail.st: No such file or directory
The statistics file does not exist.

mailstats: /etc/mail/sendmail.st: Permission denied
The statistics file's permissions are set so that you cannot read it.

EXAMPLES
Here is a typical example of **mailstats** output:

```
Statistics from Fri Dec  7 12:34:47 1990
M msgsfr  bytes_from  msgsto  bytes_to
0    216        318K      68       76K
3     80        108K      75      124K
4      0          0K      60       64K
```

This example shows that mailers 0, 3 and 4 have handled the given amounts of mail traffic since Friday, December 6th. Specifically, **sendmail** has sent 60 messages containing 64 kilobytes via mailer number 4, but has not received any inbound messages via mailer number 4.

AUTHOR
mailstats was developed by the University of California, Berkeley.

FILES

/etc/mail/sendmail.st	mail traffic statistics file
/etc/mail/sendmail.cf	sendmail configuration file

SEE ALSO
sendmail(1M).

m

NAME

mailx - interactive message processing system

SYNOPSIS

Send mode:

mailx [**-FUm**] [**-s** *subject*] [**-r** *address*] [**-h** *number*] *address* ...

Receive mode:

mailx -e

mailx [**-UHiNn**] [**-u** *user*]

mailx -f [**-UHiNn**] [*filename*]

Obsolescent:

mailx [**-f** *filename*] [**-UHiNn**]

DESCRIPTION

mailx provides a comfortable, flexible environment for sending and receiving messages electronically. When reading mail, **mailx** provides commands to facilitate saving, deleting, and responding to messages. When sending mail, **mailx** allows editing, reviewing and other modification of the message as it is created.

Incoming mail for each user is stored in a standard file called the **system mailbox** for that user. When using **mailx** to read messages, the system mailbox is used unless an alternate mailbox file is specified by using the **-f** option with or without a specific filename. As incoming messages are read from the system mailbox, they are marked to be moved to a secondary file for storage (unless specific action is taken) so that the messages need not be seen again. This secondary file is called the *mbox* and is usually located in the user's **HOME** directory (see **MBOX** description under ENVIRONMENT VARIABLES below for a description of this file and other environment variables used by **mailx**). Messages remain in this file until specifically removed.

Command-line options start with a hyphen (**-**), and any other arguments are assumed to be destinations (recipients). If no recipients are specified, **mailx** attempts to read messages from the system mailbox.

Recipient addresses specified on the command line must total less than 1024 characters in length. You may declare an **alias** or **group** (see COMMANDS below) to specify a recipient address or list of addresses of up to 8191 characters, and use that alias or group name (though each address in the list must still be less than 1024 characters). If you wish to specify a list of recipient addresses of greater length than this, have your system administrator declare an alias or group in the system alias file **/etc/mail/aliases** and use that alias name instead.

Options

mailx recognizes the following command-line options:

-e	Test for presence of mail. **mailx** prints nothing and exits with a successful return code if there is mail to read. Sometimes used in login scripts such as **$HOME/.profile** to check for mail during login.
-f	Read messages from *filename* instead of the user's system mailbox. If *filename* is not specified, the secondary *mbox* is used.
-F	Record the message in a file named after the first recipient. Overrides the **record** environment variable, if set.
-h *number*	The number of network "hops" made so far. This is provided for network software to prevent infinite delivery loops.
-H	Print header summary only.
-i	Ignore interrupts. Also see the description of the **ignore** environment below.
-n	Do not initialize from the system default **mailx.rc** file.
-m	Do not add MIME header lines *Mime Version, Content Type & Content Encoding* to the header information while sending mail.
-N	Do not print initial header summary.

m

-r *address*	Pass *address* to network delivery software. All tilde commands are disabled.
-s *subject*	Set the Subject header field to *subject*.
-u *user*	Read *user*'s *mailbox*. Can be used only if read access to *user*'s mailbox is not read protected.
-U	Convert UUCP-style addresses to Internet standards. Overrides the **conv** environment variable.
-d	Turn on debugging output. Neither particularly interesting nor recommended.

When reading mail, **mailx** operates in **command mode**. A header summary of the first several messages is displayed, followed by a prompt indicating that **mailx** can accept regular commands (see COMMANDS below). When sending mail, **mailx** operates in **input mode**. If no subject is specified on the command line, a prompt for the subject is printed. As the message is typed, **mailx** reads the message and stores it in a temporary file. Commands can be entered by beginning a line with the tilde (~) escape character followed by a single command letter and optional arguments. See TILDE ESCAPES for a summary of these commands.

The behavior of **mailx** at any given time is governed by a set of **environment variables**; flags and valued parameters that are set and cleared by using the **se**t and **uns**et commands. See ENVIRONMENT VARIABLES below for a summary of these parameters.

Recipients listed on the command line can be of three types: login names, shell commands, or alias groups. Login names can be any network address, including mixed network addressing. If the recipient name begins with a pipe symbol (**|**), the rest of the name is assumed to be a shell command to pipe the message through. This provides an automatic interface with any program that reads the standard input, such as **lp** (see *lp*(1)) for recording outgoing mail on paper. Alias groups are set by the **a**lias command (see COMMANDS below) and are lists of recipients of any type.

m

Regular commands are of the form

[*command*] [*msglist*] [*arguments*]

If no command is specified in command mode, **p**rint is assumed. In input mode, commands are recognized by the escape character (tilde unless redefined by the **escape** environment variable), and lines not treated as commands are treated as input for the message.

Each message is assigned a sequential number, and there is always the notion of a **current message**, marked by a **>** in the header summary. Many commands take an optional list of messages (*msglist*) to operate on, which defaults to the current message. A *msglist* is a list of message specifications separated by spaces. The message list can include:

n	Message number *n*.
.	The current message.
^	The first undeleted message.
$	The last message.
*****	All messages.
*n***-***m*	An inclusive range of message numbers, *n* through *m*, where *n* is less than *m*.
user	All messages from *user*.
/*string*	All messages with *string* in the subject line (uppercase-lowercase differences are ignored).
:*c*	All messages of type *c*, where *c* is one of:

d	deleted messages
n	new messages
o	old messages
r	read messages
u	unread messages

Note that the context of the command determines whether this type of message specification makes sense.

Other arguments are usually arbitrary strings whose usage depends on the command involved. File names, where expected, are expanded using normal shell conventions (see *sh*(1)). Special characters are recognized by certain commands, and are documented with the commands below.

At start-up time, **mailx** reads commands from a system-wide file (**/usr/share/lib/mailx.rc**) to initialize certain parameters, then from a private start-up file (**$HOME/.mailrc**) for personalized variables. Most regular commands are legal inside start-up files, the most common use being to set up initial display options and alias lists. The following commands are not legal in the start-up file: **!**, **C**opy, **e**dit, **fo**llowup, **F**ollowup, **ho**ld, **m**ail, **pre**serve, **r**eply, **R**eply, **sh**ell, and **v**isual. Any errors in the start-up file cause the remaining lines in the file to be ignored.

COMMANDS

The following is a complete list of **mailx** commands:

! *command* — Escape to the shell. See the description of the **SHELL** environment variable below.

*comment* — Null command (comment). Useful in **.mailrc** files.

= — Print the current message number.

? — Print a summary of commands.

<new-line> — Advance to next message and **p**rint. If this is the first command entered, the first unread message is printed. (To read the current message, use **p**rint.)

alias *alias name*...
group *alias name*...
: Declare an alias for the given names. The names are substituted when *alias* is used as a recipient. Useful in the **.mailrc** file.

alternates *name*...
: Declares a list of alternate names for your login. When responding to a message, these names are removed from the list of recipients for the response. With no arguments, **alt**ernates prints the current list of alternate names. See also **allnet** under ENVIRONMENT VARIABLES.

cd [*directory*]
chdir [*directory*] — Change directory. If *directory* is not specified, **$HOME** is used.

copy [*filename*]
copy [*msglist*] *filename*
: Copy messages to the file without marking the messages as saved. Otherwise equivalent to the **s**ave command.

Copy [*msglist*] — Save the specified messages in a file whose name is derived from the author of the message to be saved, without marking the messages as saved. Otherwise equivalent to the **S**ave command.

delete [*msglist*] — Delete messages from the *mailbox*. If **autoprint** is set, the next message after the last one deleted is printed (see ENVIRONMENT VARIABLES). See also **dp**.

discard [*header-field* ...]
ignore [*header-field* ...]
: Suppresses printing of the specified header fields when displaying messages on the screen. Examples of header fields to ignore are "status" and "cc." The fields are included when the message is saved. The **P**rint and **T**ype commands override this command.

dp[*msglist*]
dt[*msglist*] — Delete the specified messages from the mailbox and print the next message after the last one deleted. Roughly equivalent to a **d**elete command followed by a **p**rint command.

echo *string* ... — Echo the given string or strings (similar to **echo** – see *echo*(1)).

edit [*msglist*] — Edit the given messages. The messages are placed in a temporary file and the **EDITOR** variable is used to get the name of the editor (see ENVIRONMENT VARIABLES). Default editor is *ed* (see *ed*(1)).

exit

xit
: Exit from **mailx**, without changing the mailbox. No messages are saved in the *mbox* (see also **q**uit).

file [*filename*]
folder [*filename*]
: Quit from the current file of messages and read in the specified file. Several special characters are recognized when used as file names, and substitutions are made as follows:

%	the current mailbox.
%*user*	the mailbox for *user*.
#	the previous file.
&	the current *mbox*.

Default file is the current mailbox.

folders
: Print the names of the files in the directory set by the **folder** variable (see ENVIRONMENT VARIABLES).

followup [*message*]
: Respond to a message and record the response in a file whose name is derived from the author of the message. Overrides the **record** variable, if set. See also the **F**ollowup, **S**ave, and **C**opy commands and **outfolder** (see ENVIRONMENT VARIABLES).

Followup [*msglist*]
: Respond to the first message in the *msglist*, sending the message to the author of each message in the *msglist*. The subject line is extracted from the first message and the response is recorded in a file whose name is derived from the author of the first message. See also the **fo**llowup, **S**ave, and **C**opy commands and **outfolder** (see ENVIRONMENT VARIABLES).

from [*msglist*]
: Print the header summary for the specified messages.

group *alias name ...*
alias *alias name ...*
: Declare an alias for the given names. The names are substituted when *alias* is used as a recipient. Useful in the **.mailrc** file.

headers [*message*]
: Prints the page of headers which includes the message specified. The **screen** variable sets the number of headers per page (see ENVIRONMENT VARIABLES). See also the **z** command.

help
: Prints a summary of commands.

hold [*msglist*]
preserve [*msglist*]
: Holds the specified messages in the *mailbox*.

if **s**|**r**
  mail-commands
else
  mail-commands
endif
: Conditional execution, where **s** executes the accompanying *mail-commands*, up to an **el**se or **en**dif if the program is in send mode, and **r** causes the accompanying *mail-commands* to be executed only in receive mode. Intended for use in **.mailrc** files.

ignore *header-field ...*
discard *header-field ...*
: Suppresses printing of the specified header fields when displaying messages on the screen. Examples of header fields to ignore are **status** and **cc**. All fields are included when the message is saved. The **P**rint and **T**ype commands override this command.

list
: Prints all commands available. No explanation is given.

mail *name ...*
: Mail a message to the specified users.

mbox [*msglist*]
: Arrange for the given messages to end up in the standard *mbox* save file when **mailx** terminates normally. See **MBOX** description under ENVIRONMENT VARIABLES for a description of this file. See also the **ex**it and **q**uit commands.

next [*message*]
: Go to next message matching *message*. A *msglist* can be specified, but in this case the first valid message in the list is the only one used. This is useful for jumping to the next message from a specific user since the name would be interpreted as a command

in the absence of a real command. See the discussion of *msglist*s above for a description of possible message specifications.

pipe [*msglist*] [*command*]
| [*msglist*] [*command*]
Pipe messages in *msglist* through the specified *command*. Each message is treated as if it were read. If *msglist* is not specified, the current message is used. If *command* is not specified, the command specified by the current value of the **cmd** variable is used. If *msglist* is specified, *command* must also be specified. If the **page** variable is set, a form feed character is inserted after each message (see ENVIRONMENT VARIABLES).

preserve [*msglist*]
hold [*msglist*]
Preserve the specified messages in the *mailbox*.

Print [*msglist*]
Type [*msglist*]
Print the specified messages on the screen, including all header fields. Overrides suppression of fields by the **ig**nore command.

print [*msglist*]
type [*msglist*]
Print the specified messages. If **crt** is set, messages longer than the number of lines specified by the **crt** variable are paged through the command specified by the **PAGER** variable. The default command is **pg** (see *pg*(1)), but many users prefer **more** (see *more*(1) – see ENVIRONMENT VARIABLES).

quit
Exit from **mailx**, storing messages that were read in *mbox* and unread messages in the user's system mailbox. Messages that have been explicitly saved in a file are deleted.

Reply [*msglist*]
Respond [*msglist*]
Send a response to the author of each message in the *msglist*. The subject line is taken from the first message. If **record** is set to a file name, the response is saved at the end of that file (see ENVIRONMENT VARIABLES).

reply [*message*]
respond [*message*]
Reply to the specified message, including all other recipients of the message. If **record** is set to a file name, the response is saved at the end of that file (see ENVIRONMENT VARIABLES).

Save [*msglist*]
Save the specified messages in a file whose name is derived from the author of the first message. The name of the file is based on the author's name with all network addressing stripped off. See also the **C**opy, **fo**llowup, and **F**ollowup commands and **outfolder** (see ENVIRONMENT VARIABLES).

save [*filename*]
save [*msglist*] *filename*
Save the specified messages in the given file. The file is created if it does not exist. The message is deleted from the *mailbox* when **mailx** terminates unless **keepsave** is set (see ENVIRONMENT VARIABLES and the **ex**it and **q**uit commands).

set
set *name*
set *name*=*string*
set *name*=*number*
Define a variable called *name*. The variable can be given a null, string, or numeric value. **Se**t by itself prints all defined variables and their values (see ENVIRONMENT VARIABLES for detailed descriptions of the **mailx** variables).

shell
Invoke an interactive shell (see **SHELL** under ENVIRONMENT VARIABLES).

size [*msglist*]
Print the size in characters of the specified messages.

source *filename*
Read commands from the given file and return to command mode.

top [*msglist*]
Print the top few lines of the specified messages. If the **toplines** variable is set, it is interpreted as the number of lines to print (see ENVIRONMENT VARIABLES). The default is 5.

touch [*msglist*]
Touch the specified messages. If any message in *msglist* is not specifically saved in a file, it is placed in the *mbox* upon normal termination. See **ex**it and **q**uit.

Type [*msglist*]

m

Print [*msglist*]	Print the specified messages on the screen, including all header fields. Overrides suppression of fields by the **ig**nore command.
type [*msglist*] **p**rint [*msglist*]	Print the specified messages. If **crt** is set, messages longer than the number of lines specified by the **crt** variable are paged through the command specified by the **PAGER** variable. The default command is *pg*(1) but many users prefer *more*(1) (see ENVIRONMENT VARIABLES).
unalias *alias*	Discard the specified *alias* names.
undelete [*msglist*]	Restore the specified deleted messages. Restores only messages that were deleted in the current mail session. If **autoprint** is set, the last message of those restored is printed (see ENVIRONMENT VARIABLES).
unset *name***...**	Cause the specified variables to be erased. If the variable was a shell variable imported from the execution environment, it cannot be erased.
version	Prints the current version and release date.
visual [*msglist*]	Edit the given messages with a screen editor. The messages are placed in a temporary file and the **VISUAL** variable is used to get the name of the editor (see ENVIRONMENT VARIABLES).
write [*msglist*] *filename*	Write the given messages on the specified file, except for the header (the "From ..." line) and trailing blank line. Otherwise equivalent to the **s**ave command.
xit **ex**it	Exit from **mailx**, without changing the *mailbox*. No messages are saved in the *mbox* (see also **q**uit).
z[**+**|**-**]	Scroll the header display forward or backward one screen-full. The number of headers displayed is set by the **screen** variable (see ENVIRONMENT VARIABLES).

m

TILDE ESCAPES

The following commands can be used only when in input mode, by beginning a line with the tilde escape character (**˜**). See **escape** (ENVIRONMENT VARIABLES) for changing this special character.

˜! *command*	Escape to the shell.
˜.	Simulate end of file (terminate message input).
˜: *mail-command* **˜_** *mail-command*	Perform the command-level request. Valid only when sending a message while reading mail.
˜?	Print a summary of tilde escapes.
˜A	Insert the autograph string **Sign** into the message (see ENVIRONMENT VARIABLES).
˜a	Insert the autograph string **sign** into the message (see ENVIRONMENT VARIABLES).
˜b *name ...*	Add *name* to the blind carbon copy (Bcc) list.
˜c *name ...*	Add *name* to the carbon copy (Cc) list.
˜d	Read in the **dead.letter** file. See **DEAD** (under ENVIRONMENT VARIABLES) for a description of this file.
˜e	Invoke the editor on the partial message. Also see the **EDITOR** environment variable description below.
˜f [*msglist*]	Forward the specified messages. The messages are inserted into the message without alteration.
˜h	Prompt for Subject line and To, Cc, and Bcc lists. If the field is displayed with an initial value, it can be edited as if you had just typed it.
˜i *string*	Insert the value of the named variable into the text of the message. For example, **˜A** is equivalent to **˜i Sign**.
˜m [*msglist*]	Insert the specified messages into the letter, shifting the new text to the right one tab stop. Valid only when sending a message while reading mail.

˜p	Print the message being entered.
˜q	Quit (terminate) input mode by simulating an interrupt. If the body of the message is not null, the partial message is saved in **dead.letter**. See the description of the **DEAD** environment variable below for a description of this file.
˜R *name ...*	Add *name* to the Reply-To list.
˜r *filename* **˜<** *filename* **˜<!** *command*	Read in the specified file. If the argument begins with an exclamation point (**!**), the rest of the string is assumed to be an arbitrary shell command and is executed, with the standard output inserted into the message.
˜s *string ...*	Set the subject line to *string*.
˜t *name ...*	Add the given *name*s to the To list.
˜v	Invoke a preferred screen editor on the partial message. Also see the **VISUAL** environment variable description below.
˜w *filename*	Write the partial message onto the given file, without the header.
˜x	Exit as with **˜q** except the message is not saved in *dead.letter*.
˜| *command*	Pipe the body of the message through the given *command*. If *command* returns a successful exit status, the output of the command replaces the message.

ENVIRONMENT VARIABLES

The following variables are internal **mailx** program variables. They can be imported from the execution environment or set by the **se**t command at any time. The **uns**et command can be used to erase variables.

allnet	All network names whose login names match are treated as identical. This causes the *msglist* message specifications to behave similarly. Default is **noallnet**. See also the **alt**ernates command and the **metoo** variable.
append	Upon termination, append messages to the end of the *mbox* file instead of inserting them at the beginning of the file. Default is **noappend.**
askbcc	Prompt for the Bcc list after the message is entered. Default is **noaskbcc.**
askcc	Prompt for the Cc list after the message is entered. Default is **noaskcc**.
asksub	Prompt for a subject if it is not specified on the command line with the **-s** option. Enabled by default.
autoprint	Enable automatic printing of messages after **d**elete and **u**ndelete commands. Default is **noautoprint**.
bang	Enable special-case treatment of exclamation points (**!**) in shell escape command lines as in *vi*(1). Default is **nobang**.
charset=*charset*	Set the default character-set. If none is specified, **mailx** will attempt to use the value of **LANG** to look up the system default for the user's locale. If that is unsuccessful, the default value of *us-ascii* will be used.
cmd=*command*	Set the default command for the **pi**pe command. No default value.
conv=*conversion*	Convert UUCP addresses to the specified address style. The only valid conversion currently supported is *internet*, which requires a mail delivery program conforming to the RFC822 standard for electronic mail addressing. Conversion is disabled by default. See also **sendmail** and the **-U** command-line option.
crt=*number*	Pipe messages having more than *number* lines through the command specified by the value of the **PAGER** variable **pg** by default (see *pg*(1)). Disabled by default.
DEAD=*filename*	The name of the file in which to save partial letters in case of untimely interrupt or delivery errors. Default is **$HOME/dead.letter.**
debug	Enable verbose diagnostics for debugging. Messages are not delivered. Default is **nodebug**.
dot	When processing input from a terminal, interpret an ASCII period character on a line by itself as end-of-file. Default is **nodot**.

EDITOR=*command* The command to run when the **e**dit or **~e** command is used. Default is **ed** (see *ed*(1)).

encoding=*encoding*
Set the default encoding to be used when 8-bit characters are present. Allowable values are *quoted-printable*, *base64* and *8bit*. The short-hand *q-p* is also acceptable for quoted-printable. The default value will be determined based upon the value of **charset.** A value of *8bit* means not to encode.

escape=*c* Substitute *c* for the **~** escape character.

folder=*directory* The directory for saving standard mail files. User specified file names beginning with a plus (+) are expanded by preceding the file name with this directory name to obtain the real file name. If *directory* does not start with a slash (**/**), **$HOME** is used as a prefix. There is no default for the **folder** variable. See also **outfolder** below.

header Enable printing of the header summary when entering **mailx**. Enabled by default.

hold Preserve all messages that are read in the system mailbox instead of putting them in the standard *mbox* save file. Default is **nohold**.

ignore Ignore interrupts while entering messages. Useful when communicating over noisy dial-up lines. Default is **noignore**.

ignoreeof Ignore end-of-file during message input. Input must be terminated by a period (**.**) on a line by itself or by the **~.** command. Default is **noignoreeof**. See also **dot** above.

keep When the *mailbox* is empty, truncate it to zero length instead of removing it. Disabled by default.

keepsave Keep messages that have been saved in other files in the system mailbox instead of deleting them. Default is **nokeepsave**.

MBOX=*filename* The name of the file to save messages which have been read. The **x**it command overrides this function, as does saving the message explicitly in another file. Default is **$HOME/mbox**.

metoo Usually, when a group (alias) containing the sender is expanded, the sender is removed from the expansion. Setting this option causes the sender to be included in the group. Default is **nometoo**.

LISTER=*command* The command (and options) to use when listing contents of the **folder** directory. The default is *ls*(1).

onehop When responding to a message that was originally sent to several recipients, the other recipient addresses are normally forced to be relative to the originating author's machine for the response. This flag disables alteration of the recipients' addresses, improving efficiency in a network where all machines can send directly to all other machines (i.e., one hop away).

outfolder Cause the files used to record outgoing messages to be located in the directory specified by the **folder** variable. Default is **nooutfolder**. See **folder** above and the **S**ave, **C**opy, **fo**llowup, and **F**ollowup commands.

page Used with the **pi**pe command to insert a form feed after each message sent through the pipe. Default is **nopage**.

PAGER= *command* The command to use as a filter for paginating output. This can also be used to specify the pager command-line options (for example, **set PAGER=**Default is **pg** (see *pg*(1)), but many users prefer **more** (see *more*(1)).

prompt= string
Set the *command mode* prompt to *string*. Default is **?**.

quiet Refrain from printing the opening message and version when entering **mailx**. Default is **noquiet**.

record=*filename* Record all outgoing mail in *filename*. Disabled by default. See also **outfolder** above.

save	Enable saving of messages in **dead.letter** on interrupt or delivery error. See **DEAD** for a description of this file. Enabled by default.
screen=*number*	Set the number of lines in a screen-full of headers for the **h**eaders command.
sendmail=*command*	Alternate command for delivering messages. Default is **mail** (see *mail*(1)).
sendwait	Wait for background mailer to finish before returning. Default is **nosendwait**.
SHELL=*command*	The name of a preferred command interpreter. Default is the user's login program (see *passwd*(4), *shells*(4), and *chsh*(1)). Note: in the unusual case that a user's login program is a script file from which **mailx** is executed, rather than a shell, then **mailx** requires that the user explicitly set **SHELL=***/usr/bin/sh* in his or her **$HOME/.mailrc** file.
showto	When displaying the header summary and the message is from you, print the recipient's name instead of the author's name.
sign=*string*	The variable that is inserted into the text of a message when the **~a** (autograph) command is given. No default (see also **~i** under ILDE ESCAPES).
Sign=*string*	The variable inserted into the text of a message when the **~A** command is given. No default (see also **~i** (TILDE ESCAPES)).
SMARTMAILER	When **SMARTMAILER** is set, various commands use the **From:** line instead of the default **From** line.
toplines=*number*	The number of lines of header to print with the **to**p command. Default is 5.
VISUAL=*command*	The name of a preferred screen editor. Default is **vi** (see *vi*(1)).

EXTERNAL INFLUENCES

Environment Variables

The following are environment variables taken from the execution environment and are not alterable within **mailx**.

HOME	The user's home directory. This is usually the current directory immediately after login.
MAILRC	The name of the mailer start-up file. Default is **$HOME/.mailrc.**
LC_COLLATE **LC_CTYPE**	**LC_COLLATE** and **LC_CTYPE** influence **mailx** when the command interpreter (see SHELL below) is invoked. To determine the behavior of **LC_COLLATE** and **LC_CTYPE**, see the corresponding shell manual entry for the applicable command interpreter
LC_TIME	**LC_TIME** determines the format and contents of the date and time strings displayed. If **LC_TIME** is not specified in the environment, or is set to the empty string, the value of **LANG** is used as a default. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG.** If any internationalization variable contains an invalid setting, **mailx** behaves as if all internationalization variables are set to "C". See *environ*(5).
TMPDIR	When set, the **TMPDIR** environment variable specifies a directory to be used for temporary files, overriding the default directory **/tmp**.

International Code Set Support

Single- and multi-byte character code sets are supported within mail text. Headers are restricted to characters from the 7-bit USASCII character code set (see *ascii*(5)).

WARNINGS

Where *command* is shown as valid, arguments are not always allowed. Experimentation is recommended.

Internal variables imported from the execution environment cannot be **uns**et.

The full internet addressing is not fully supported by **mailx**. The new internationalization standards need some time to settle down.

mail(1), the standard mail delivery program, treats a line consisting solely of a dot (**.**) as the end of the message.

Using two separate mail programs to access the same mail file simultaneously (usually inadvertently from two separate windows) can cause unpredictable results.

FILES

/var/mail/	Post office directory (mode 775, group ID **mail**)
/var/mail/*user*	System mailbox for *user* (mode 660, owned by *user*,*group* ID **mail**)
$HOME/.mailrc	Personal start-up file
/usr/share/lib/mailx.rc	Global start-up file
$HOME/mbox	Secondary storage file
/tmp/R[**emqsx**]*****	Temporary files

SEE ALSO

mail(1), pg(1), ls(1).

mailx tutorial in *Mail Systems Users Guide*.

STANDARDS CONFORMANCE

mailx: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

NAME

make - maintain, update, and regenerate groups of programs

SYNOPSIS

make [**-f** *makefile*] [**-bBdeiknpPqrsStuw**] [*macro_name*=*value*] [*names*]

DESCRIPTION

Makefile Structure

A makefile can contain four different kinds of lines: target lines, shell command lines, macro definitions, and include lines.

TARGET LINES

Target lines consist of a blank-separated, non-null list of targets, followed by a colon (**:**) or double colon (**::**), followed by a (possibly null) list of prerequisite files called **dependents**. Pattern Matching Notation (see *regexp*(5)) is supported for the generation of file names as dependents.

SHELL COMMAND LINES

Text following a semicolon (**;**) on a target line, and all following lines that begin with a tab are shell commands to be executed to update the target (see the Environment section below about SHELL). The first line that does not begin with a tab or **#** begins a new target definition, macro definition, or include line. Shell commands can be continued across lines by using a <backslash><new-line> sequence.

Target lines with their associated command lines are called *rules*.

MACROS

Lines of the form *string1* **=** *string2* are macro definitions. Macros can be defined anywhere in the makefile, but are usually grouped together at the beginning. *string1* is the macro name; *string2* is the macro value. *string2* is defined as all characters up to a comment character or an unescaped new-line. Spaces and tabs immediately to the left and right of the **=** are ignored. Subsequent appearances of **$(***string1***)** anywhere in the makefile (except in comments) are replaced by *string2*. The parentheses are optional if a single character macro name is used and there is no substitute sequence. An optional substitute sequence, **$(***string1* [**:***subst1***=**[*subst2*]]**)** can be specified, which causes all nonoverlapping occurrences of *subst1* at the end of substrings in the value of *string1* to be replaced by *subst2*. Substrings in a macro value are delimited by blanks, tabs, new-line characters, and beginnings of lines. For example, if

```
OBJS = file1.o file2.o file3.o
```

then

```
$(OBJS:.o=.c)
```

evaluates to

```
file1.c file2.c file3.c
```

Macro values can contain references to other macros (see WARNINGS):

```
ONE =1
TWELVE = $(ONE)2
```

The value of **$(TWELVE)** is set to **$(ONE)2** but when it is used in a target, command, or include line, it is expanded to **12**. If the value of **ONE** is subsequently changed by another definition further down in the makefile or on the command line, any references to **$(TWELVE)** reflect this change.

Macro definitions can also be specified on the command line and override any definitions in the makefile.

(XPG4 only. Macros on the command line are added to the **MAKEFLAGS** environment variable. Macros defined in the **MAKEFLAGS** environment variable, but without any command line macro, adds the macro to the environment overwriting any existing environment variable of the same name.)

Certain macros are automatically defined for **make** (see Built-in Macros). See the Environment section for a discussion of the order in which macro definitions are treated.

The value assigned to a macro can be overridden by a conditional macro definition. A conditional macro definition takes on the form *target* **:=** *string1* **=** *string2*. When the target line associated with target is being processed, the macro value specified in the conditional macro definition is in effect. If *string1* is previously defined, the new value of *string1* will override the previous definition. The new value of string1 takes effect when target or any dependents of target are being processed.

INCLUDE LINES

If the string **include** appears as the first seven letters of a line in a makefile, and is followed by one or more space or tab characters, the rest of the line is assumed to be a file name and is read and processed by the current invocation of **make** as another makefile after any macros in the filename have been expanded.

The default behaviour of **make** is to use **.DEFAULT** built-in target, if target does not have explicit commands associated with it and **.DEFAULT** target is defined. See the Built-In Targets Section.

General Description

make executes commands previously placed in a makefile to update one or more target names. Target names are typically names of programs. If no **-f** option is specified, the filenames **makefile**, **Makefile**, **s.makefile**, **SCCS/s.makefile**, **s.Makefile** and **SCCS/s.Makefile** are tried in that order. If **-f -** is specified, the standard input is used. More than one **-f** option can be specified. The makefile arguments are processed in the order specified. A space between the **-f** and the filename **must** be present, and multiple makefile names must each have their own **-f** option preceding them. The contents of a makefile override the built-in rules and macros if they are present.

If no target names are specified on the command line, **make** updates the first target in the (first) makefile that is not an inference rule. A target is updated only if it depends on files that are newer than the target. Missing files are deemed to be out-of-date. All dependents of a target are recursively updated, if necessary, before the target is updated. This effects a depth-first update of the dependency tree for the target.

If a target does not have any dependents specified after the separator on the target line (*explicit dependents*), any shell commands associated with that target are executed if the target is out-of-date.

A target line can have either a single or double colon between the target name or names and any explicit dependent names. A target name can appear on more than one target line, but all of those lines must be of the same (single- or double-colon) type. For the usual single-colon case, at most one of these target lines can have explicit commands associated with it. If the target is out-of-date with any of its dependents on any of the lines, the explicit commands are executed, if they are specified, or else a default rule can be executed. For the double-colon case, explicit commands can be associated with more than one of the target lines containing the target name; if the target is out-of-date with any of the dependents on a particular line, the commands for that line are executed. A built-in rule may also be executed.

m

Target lines and their associated shell command lines are also referred to as **rules**. Hash marks (**#**) and new-line characters surround comments anywhere in the makefile except in rules. Comments in the rules depend on the setting of the **SHELL** macro.

The following makefile says that **pgm** depends on two files: **a.o** and **b.o**, and that they in turn depend on their corresponding source files (**a.c** and **b.c**) and a common file **incl.h**:

```
OBJS = a.o b.o

pgm: $(OBJS)
    cc $(OBJS) -o pgm

a.o: incl.h a.c
    cc -c a.c

b.o: incl.h b.c
    cc -c b.c
```

Command lines are executed one at a time, each by its own shell. Each command line can have one or more of the following prefixes: **-**, **@**, or **+**. These prefixes are explained below.

Commands returning non-zero status normally terminate **make**. The **-i** option or the presence of the special target **.IGNORE** in the makefile cause **make** to continue executing the makefile regardless of how many command lines cause errors, although the error messages are still printed on standard output. If **-** is present at the beginning of a command line, any error returned by that line is printed to standard output but **make** does not terminate. The prefix **-** can be used to selectively ignore errors in a makefile. If the **-k** option is specified and a command line returns an error status, work is abandoned on the current target, but continues on other branches that do not depend on that target. If the **-k** option is present in the **MAKEFLAGS** environment variable, processing can be returned to the default by specifying the **-S** option.

The **-n** option specifies printing of a command line without execution. However, if the command line has the string **$(MAKE)** or **${MAKE}** in it or **+** as a prefix, the line is always executed (see discussion of the **MAKEFLAGS** macro under Environment). The **-t** (touch) option updates the modified date of a file without executing any commands.

A command line is normally printed before it is executed, but if the line has a **@** at the beginning, printing is suppressed. The **-s** option or the presence of the special target **.SILENT:** in the makefile suppresses printing of all command lines. The **@** can be used to selectively turn off printing. Everything printed by **make** (except the initial tab) is passed directly to the shell without alteration. Thus,

```
echo a\
b
```

produces

```
ab
```

just as the shell would.

The **-b** option allows old makefiles (those written for the old version of **make**) to run without errors. The old version of **make** assumed that if a target did not have any explicit commands associated with it, the user intended the command to be null, and would not execute any **.DEFAULT** rule that might have been defined. The current version of **make** operates in this mode by default. However, the current version of **make** provides a **-B** option which turns this mode off so that if a target does not have explicit commands associated with it and a **.DEFAULT** rule is defined, the **.DEFAULT** rule is executed. Note that the **-b** and **-B** options have no effect on the search and possible location and execution of an appropriate inference rule for the target. The search for a built-in inference rule other than **.DEFAULT** is always performed.

The signals **SIGINT**, **SIGQUIT**, **SIGHUP**, and **SIGTERM** (see *signal*(5)) cause the target to be deleted unless the target depends on the special name **.PRECIOUS**.

Options

The following is a brief description of all options and some special names. Options can occur in any order. They can be specified separately, or together with one **-**, except for the **-f** option.

-b	Compatibility mode for old (Version 7) makefiles. This option is turned on by default.
-B	Turn off compatibility mode for old (Version 7) makefiles.
-d	Debug mode. Print out detailed information on files and times examined. (This is very verbose and is intended for debugging the **make** command itself.)
-e	Environment variables override assignments within makefiles .
-f *makefile*	Description file name, referred to as the makefile. A file name of **-** denotes the standard input. The contents of the makefile override the built-in rules and macros if they are present. Note that the space between **-f** and *makefile* **must** be present. Multiple instances of this option are allowable (except for **-f -**), and are processed in the order specified.
-i	Ignore error codes returned by invoked commands. This mode is also entered if the special target name **.IGNORE** appears in the makefile.
-k	When a command returns nonzero status, abandon work on the current entry, but continue on other branches that do not depend on that target. This is the opposite of **-S**. If both **-k** and **-S** are specified, the last one specified is used.
-n	No execute mode. Print commands, but do not execute them. Even lines beginning with an **@** are printed. However, lines that contain the string **$(MAKE)** or **${MAKE}** or that have **+** as a prefix to the command are executed.
-p	Write to standard output the complete set of macro definitions and target descriptions.
-P	Update in parallel more than one target at a time. The number of targets updated concurrently is determined by the environment variable **PARALLEL** and the presence of **.MUTEX** directives in make file.
-q	Question. The **make** command returns a zero or non-zero status code, depending on whether the target file is or is not up-to-date. Targets are not updated with this option.
-r	Clear suffix list and do not use the built-in rules.
-s	Silent mode. Command lines are not printed to standard output before their execution. This mode is also entered if the special target name **.SILENT** appears in the makefile.
-S	Terminate if an error occurs while executing the commands to bring a target up-to-date. This is the default and the opposite of **-k.** If both **-k** and **-S** are specified, the last one

m

given is used. This enables overriding the presence of the **k** flag in the **MAKEFLAGS** environment variable.

-t Touch the target files (causing them to be up-to-date) rather than issue the usual commands.

-u Unconditionally **make** the target, ignoring all timestamps.

-w Suppress warning messages. Fatal messages will not be affected.

[*macro_name*=*value*]
Zero or more command line macro definitions can be specified. See the Macros section.

[*names*] Zero or more target names that appear in the makefile. Each target so specified is updated by **make**. If no names are specified, **make** updates the first target in the makefile that is not an inference rule.

Parallel Make

If **make** is invoked with the -P option, it tries to build more than one target at a time, in parallel. (This is done by using the standard UNIX system process mechanism which enables multiple processes to run simultaneously.) For the makefile shown in the example in the previous section, it would create processes to build **a.o** and **b.o** in parallel. After these processes were complete, it would build **pgm**.

The number of targets **make** will try to build in parallel is determined by the value of the environment variable **PARALLEL**. If -P is invoked, but **PARALLEL** is not set, then **make** will try to build no more than two targets in parallel.

You can use the **.MUTEX** directive to serialize the updating of some specified targets. This is useful when two or more targets modify a common output file, such as when inserting modules into an archive or when creating an intermediate file with the same name, as is done by lex and yacc. If the makefile in the previous section contained a **.MUTEX** directive of the form

```
.MUTEX: a.o b.o
```

it would prevent make from building a.o and b.o in parallel.

Environment

All variables defined in the environment (see *environ*(5)) are read by **make** and are treated and processed as macro definitions, with the exception of the **SHELL** environment variable which is always ignored. The value of the **SHELL** environment variable will not be used as a macro and will not be modified by defining the **SHELL** macro in a makefile or on the command line. Variables with no definition or empty string definitions are included by **make**.

There are four possible sources of macro definitions which are read in the following order: internal (default), current environment, the makefile(s), and command line. Because of this order of processing, macro assignments in a makefile override environment variables. The **-e** option allows the environment to override the macro assignments in a makefile. Command-line macro definitions always override any other definitions.

The **MAKEFLAGS** environment variable is processed by **make** on the assumption that it contains any legal input option (except **-f**, **-p**, and **-d**) defined for the command line. The **MAKEFLAGS** variable can also be specified in the makefile.

(XPG4 only. **MAKEFLAGS** in the makefile replaces the **MAKEFLAGS** environment variable. Command line options have precedence over **MAKEFLAGS** environment variable.)

If **MAKEFLAGS** is not defined in either of these places, **make** constructs the variable for itself, puts the options specified on the command line and any default options into it, and passes it on to invocations of commands. Thus, **MAKEFLAGS** always contains the current input options. This proves very useful for recursive **make**s. Even when the **-n** option is specified, command lines containing the string **$(MAKE)** or **${MAKE}** are executed; hence, one can perform a **make -n** recursively on an entire software system to see what would have been executed. This is possible because the **-n** is put into **MAKEFLAGS** and passed to the recursive invocations of **$(MAKE)** or **${MAKE}**. This is one way of debugging all of the makefiles for a software project without actually executing any of the commands.

Each of the commands in the rules is given to a shell to be executed. The shell used is the shell command interpreter (see *sh*(1)), or the one specified in the makefile by the **SHELL** macro. To ensure the same shell is used each time a makefile is executed, the line:

```
SHELL=/usr/bin/sh
```

or a suitable equivalent should be put in the macro definition section of the makefile.

Suffixes

Target and/or dependent names often have suffixes. Knowledge about certain suffixes is built into **make** and used to identify appropriate inference rules to be applied to update a target (see the section on Inference Rules). The current default list of suffixes is:

```
.o .c .c~ .C .C~ .cxx .cxx~ .cpp .cpp~ .cc .cc~
.y .y~ .l .l~ .L .L~ .Y .Y~ .s .s~ .sh .sh~
.h .h~ .H .H~  .p .p~ .f .f~ .r .r~
```

These suffixes are defined as the dependents of the special built-in target **.SUFFIXES**. This is done automatically by **make**.

Additional suffixes can be specified in a makefile as the dependents list for **.SUFFIXES**. These additional values are added to the default values. Multiple suffix lists accumulate. The order of the suffix list is significant (see the Inference Rules section). If the user wishes to change the order of the suffixes, he must first define **.SUFFIXES** with a null dependent list, which clears the current value for **.SUFFIXES**, and then define **.SUFFIXES** with the suffixes in the desired order. The list of suffixes built into **make** on any machine can be displayed by:

```
make -fp - 2>/dev/null </dev/null
```

The list of built-in suffixes incorporated with the definitions in a given makefile called **mymakefile** can be displayed by:

```
make -fp mymakefile 2>/dev/null </dev/null
```

Inference Rules

Certain target or dependent names (such as those ending with **.o**) have inferable dependents such as **.c** and **.s**, etc. If no update commands for such a name appear in the makefile, and if an inferable dependent file exists, that dependent file is compiled to update the target. In this case, **make** has inference rules that allow building files from other files by examining the suffixes and determining an appropriate inference rule to use. There are currently default inference rules defined for:

Single Suffix Rules

```
.c .c~
.C .C~ .cxx .cxx~ .cpp .cpp~ .cc .cc~
.sh .sh~
.p .p~
.f .f~
.r .r~
```

Double Suffix Rules

```
.c.o .c~.o .c~.c .c.a .c~.a
.C.o .C~.o .C~.C .C.a .C~.a
.cxx.o .cxx~.o .cxx~.cxx .cxx.a .cxx~.a
.cpp.o .cpp~.o .cpp~.cpp .cpp.a .cpp~.a
.cc.o .cc~.o .cc~.cc .cc.a .cc~.a
.s.o .s~.o .s~.a
.p.o .p~.o .p~.p .p.a .p~.a
.f.o .f~.o .f~.f .f.a .f~.a
.r.o .r~.o .r~.r .r.a .r~.a
.y.o .y~.o .y.c .y~.c
.l.o .l~.o .l.c
.h~.h  .H~.H .hxx~.hxx .hpp~.hpp
.C.o .C~.o .C.a .C~.a
.L.C .L.o .L~.C .L~.L .L~.o
.Y.C .Y.o .Y~.C .Y~.Y .Y~.o
```

Double suffix inference rules (**.c.o**) define how to build **x.o** from **x.c**. Single suffix inference rules (**.c**) define how to build **x** from **x.c**. In effect, the first suffix is null. Single suffix rules are useful for building targets from only one source file; e.g., shell procedures and simple C programs.

m

A tilde in the above rules refers to an SCCS file (see *sccsfile*(4)). Thus, the rule **.c~.o** would transform an SCCS C source file into an object file (**.o**). Since the **s.** of the SCCS files is a prefix, it is incompatible with **make**'s suffix point-of-view. Hence, the tilde is a way of changing any file reference into an SCCS file reference.

A rule to create a file with suffix **.o** from a file with suffix **.c** is specified as an entry with **.c.o** as the target and no dependents. Shell commands associated with the target define the rule for making a **.o** file from a **.c** file. Any target name that has no slashes in it and starts with a dot is identified as an inference (**implicit**) rule instead of a target (**explicit**) rule. Targets with one dot are single suffix inference rules; targets with two dots are double suffix inference rules. Users can, in a makefile, define additional inference rules and either redefine or cancel default inference rules.

The default inference rule for changing a **.c** file into a **.o** file might look like this:

```
.c.o:
    $(CC) $(CFLAGS) -c $<
```

and the default inference rule for changing a **yacc** file to a C object file might look like this:

```
    .y.o:
        $(YACC) $(YFLAGS) $<
        $(CC) $(CFLAGS) -c y.tab.c
        rm y.tab.c
        mv y.tab.o $@
```

Certain macros are used in the default inference rules to permit the inclusion of optional matter in any resulting commands. For example, **CFLAGS**, **LDFLAGS**, and **YFLAGS** are used for compiler options to *cc*(1), *lex*(1), and *yacc*(1), respectively. **LDFLAGS** is commonly used to designate linker/loader options. These macros are automatically defined by **make** but can be redefined by the user in the makefile.

The macro **LIBS** is, by convention, used to specify the order of inclusion of any special libraries during the linking phase of compilation. To specify a particular order of inclusion for a particular set of libraries, the existing single suffix rule for a **.c** file,

m

```
    $(CC) $(CFLAGS) $< $(LDFLAGS) -o $@
```

can be redefined as

```
    $(CC) $(CFLAGS) $< $(LDFLAGS) -o $@ $(LIBS)
```

as well as defining **LIBS** in the makefile.

There are also some special built-in macros used in the inference rules (**@**, **<**). See the Built-in Macros section.

If a target does not have explicit dependents, or if a dependent does not also have a target that matches it with associated explicit rules, **make** looks for the first inference rule that matches both the target's (dependent's) suffix (which may be null) and a file which matches the other suffix of the rule. Since it conducts this search by going through the list of **.SUFFIXES** values front to back, the order in which **.SUFFIXES** is defined is significant.

To print out the rules compiled into the **make** on any machine, type:

```
    make -fp - 2>/dev/null </dev/null
```

Since **make** defines an inference rule **.c.o**, the example in the General Description section can be rewritten more simply:

```
    OBJS = a.o b.o
        pgm: $(OBJS)
            cc $(OBJS) -o pgm
        $(OBJS): incl.h
```

Libraries

If a target or dependent name contains parentheses, it is assumed to be an archive library, the string within parentheses referring to a member within the library. Thus **lib(file.o)** and **$(LIB)(file.o)** both refer to an archive library that contains **file.o** (this assumes the **LIB** macro has been previously defined). The expression **$(LIB)(file1.o file2.o)** is not valid. Rules pertaining to archive libraries have the form **.***xx***.a** where *xx* is the suffix from which the archive member is to be made. An unfortunate byproduct of the current implementation requires the *xx* to be different from the suffix of the archive member. Thus, one cannot have **lib(file.o)** depend upon **file.o** explicitly.

The most common use of the archive interface follows. Here, we assume the source files are all C type source:

```
lib: lib(file1.o) lib(file2.o) lib(file3.o)
     @echo lib is now up-to-date
.c.a:
     $(CC) -c $(CFLAGS) $<
     ar rv $@ $*.o
     rm -f $*.o
```

(See the section on Built-in Macros for an explanation of the **<**, **@**, and ***** symbols.) In fact, the **.c.a** rule listed above is built into **make** and is unnecessary in this example. This rule is applied to each dependent of **lib** in turn. The following example accomplishes this more efficiently:

```
lib:  lib(file1.o) lib(file2.o) lib(file3.o)
     $(CC) -c $(CFLAGS) $(?:.o=.c)
     ar rv lib $?
     rm $?
     @echo lib is now up-to-date
.c.a:;
```

Here substitution in the macros is used. The **$?** list is defined to be the set of object file names (inside **lib**) whose C source files are out-of-date. The substitution sequence translates the **.o** to **.c**. (Unfortunately, one cannot as yet transform to **.c~**; however, this may become possible in the future.) Note also, the disabling of the **.c.a** rule, which would have created and archived each object file, one by one. This particular construct speeds up archive library maintenance considerably, but becomes very cumbersome if the archive library contains a mix of assembly programs and C programs.

Archive members containing the definition of a symbol are designated by double parentheses around the symbol name, **lib((***entry_name***))**, but are otherwise handled as described above.

m

Built-In Targets

make has knowledge about some special targets. These must be specified in the makefile to take effect (with the exception of **.SUFFIXES**, which is automatically set by **make** but can be changed by the user).

.DEFAULT	If a file must be made but there are no explicit commands or relevant built-in rules for it, the commands associated with the target name **.DEFAULT** are used if **.DEFAULT** has been defined in the makefile. **.DEFAULT** does not have any explicit dependents.
.PRECIOUS	Dependents of this target are not removed when **QUIT**, **INTERRUPT**, **TERMINATE**, or **HANGUP** are received.
.SILENT	Same effect as the **-s** option. No dependents or explicit commands need to be specified.
.IGNORE	Same effect as the **-i** option. No dependents or explicit commands need to be specified.
.SUFFIXES	The explicit dependents of **.SUFFIXES** are added to the built-in list of known suffixes and are used in conjunction with the inference rules. If **.SUFFIXES** does not have any dependents, the list of known suffixes is cleared. There are no commands associated with **.SUFFIXES**.
.MUTEX	Serialize the updating of specified targets (See the *Parallel Make* Section).

Built-in Macros

There are five internally maintained macros that are useful for writing rules for building targets. In order to clearly define the meaning of these macros, some clarification of the terms **target** and **dependent** is necessary. When **make** updates a target, it may actually generate a series of targets to update. Before any rule (either explicit or implicit) is applied to the target to update it, recursion takes place on each dependent of the target. The dependent, upon recursion, becomes a target itself, and may have or generate its own dependents, which in turn are recursed upon until a target is found that has no dependents, at which point the recursion stops. Not all targets processed by **make** appear as explicit targets in the makefile; some of them are explicit dependents from the makefile while others are implicit dependents generated as **make** recursively updates the target. For instance, when the following makefile is executed:

```
pgm:  a.o b.o
    cc a.o b.o -o pgm
```

the following series of targets to be made is generated:

--- pgm
with two dependents and an explicit rule to follow

--- a.o
(recursively) with an implicit dependent of **a.c** which matches the implicit rule **.c.o**

--- a.c
(recursively) with no implicit dependents and no implicit rules. This stops the recursion and simply returns the last modification time of the file **a.c.**

--- b.o
(recursively) with an implicit dependent of **b.c** which matches the implicit rule **.c.o**

--- b.c
(recursively) with no implicit dependents and no implicit rules. This stops the recursion and merely returns the last modification time of the file **b.c**.

In the definitions below, the word *target* refers to a target specified in the makefile, an explicit dependent specified in the makefile which becomes the target when **make** recurses on it, or an implicit dependent (generated as a result of locating an inference rule and file that match the suffix of the target) which becomes the target when **make** recurses on it. The word dependent refers to an explicit dependent specified in the makefile for a particular target, or an implicit dependent generated as a result of locating an appropriate inference rule and corresponding file that matches the suffix of the target.

It may be helpful to think of target rules as user specified rules for a particular target name, and inference rules as user or **make** specified rules for a particular class of target names. It may also be helpful to remember that the value of the target name and its corresponding dependent names change as **make** recurses on both explicit and implicit dependents, and that inference rules are only applied to implicit dependents or to explicit dependents which do not have target rules defined for them in the makefile.

$@** The **$@ macro is the full target name of the current target, or the archive filename part of a library archive target. It is evaluated for both target and inference rules.

$% The **$%** macro is only evaluated when the current target is an archive library member of the form **libname(***member***.o)** or **libname((***entry***))**. In these cases, **$@** evaluates to **libname** and **$%** evaluates to **member.o** or the object file containing the symbol **entry**. **$%** is evaluated for both target and inference rules.

$?** The **$? macro is the list of dependents that are out-of-date with respect to the current target; essentially, those modules that have been rebuilt. It is evaluated for both target and inference rules, but is usually only used in target rules. **$?** evaluates to one name only in an inference rule, but may evaluate to more than one name in a target rule.

$<** In an inference rule, **$< evaluates to the source file name that corresponds to the implicit rule which matches the suffix of the target being made. In other words, it is the file that is out-of-date with respect to the target. In the **.DEFAULT** rule, the **$<** macro evaluates to the current target name. **$<** is evaluated only for inference rules. Thus, in the **.c.o** rule, the **$<** macro would evaluate to the **.c** file. An example for making optimized **.o** files from **.c** files is:

```
.c.o:
    cc -c -O $*.c
```

or:

```
.c.o:
    cc -c -O $<
```

$*** The macro **$* is the current target name with the suffix deleted. It is evaluated only for inference rules.

These five macros can have alternative forms. When an uppercase **D** or **F** is appended to any of the five macros, the meaning is changed to “directory part” for **D** and “file part” for **F**. Thus, **$(@D)** refers to the directory part of the string **$@**. If there is no directory part, **./** is generated. When the **$?** macro contains more than one dependent name, the **$(?D)** expands to a list of directory name parts and the **$(?F)** expands to a list of the filename parts.

In addition to the built-in macros listed above, other commonly used macros are defined by **make**. These macros are used in the default inference rules, and can be displayed with the **-p** option. These macros can be used in target rules in the makefile. They can also be redefined in the makefile.

$$@** The **$$@ macro has meaning *only* on dependency lines. Macros of this form are called **dynamic dependencies** because they are evaluated at the time the dependency is actually processed. **$$@** evaluates to exactly the same thing as **$@** does on a command line; i.e., the current target name. This macro is useful for building large numbers of executable files, each of which has only one source file. For instance, the following HP-UX commands could all be built using the same rule:

```
CMDS = cat echo cmp chown
$(CMDS) : $$@.c
    $(CC) -O $? -o $@
```

If this makefile is invoked with **make cat echo cmp chown**, **make** builds each target in turn using the generic rule, with **$$@** evaluating to **cat** while **cat** is the target, to **echo** when the target is **echo**, and so forth.

The dynamic dependency macro can also take the F form, **$$(@F)** which represents the filename part of **$$@**. This is useful if the targets contain pathnames. For example:

```
INCDIR = /usr/include
INCLUDES = $(INCDIR)/stdio.h \
        $(INCDIR)/pwd.h   \
        $(INCDIR)/dir.h   \
        $(INCDIR)/a.out.h
$(INCLUDES) : $$(@F)
    cp $? $@
    chmod 0444 $@
```

Special Macros

The VPATH macro allows make to search a colon separated list of directories for dependents. Lines of the form **VPATH**= *path1:path2* ... causes make to first search the current directory for a dependent and if the dependent is not found, make searches *path1* and continues until the directories specified in the VPATH macro are exhausted.

EXTERNAL INFLUENCES

Environment Variables

LANG provides a default value for the internationalization variables that are unset or null. If **LANG** is unset or null, the default value of "C" (see *lang*(5)) is used. If any of the internationalization variables contains an invalid setting, **make** will behave as if all internationalization variables are set to "C". See *environ*(5).

LC_ALL If set to a non-empty string value, overrides the values of all the other internationalization variables.

LC_CTYPE determines the interpretation of text as single and/or multi-byte characters, the classification of characters as printable, and the characters matched by character class expressions in regular expressions.

LC_MESSAGES determines the locale that should be used to affect the format and contents of diagnostic messages written to standard error and informative messages written to standard output.

NLSPATH determines the location of message catalogues for the processing of **LC_MESSAGES**.

PROJECTDIR provides a directory to be used to search for SCCS files not found in the current directory. In all of the following cases, the search for SCCS files will be made in the directory **SCCS** in the identified directory. If the value of **PROJECTDIR** begins with a slash, it is considered an absolute pathname; otherwise, the home directory of a user of that name is examined for a subdirectory **src** or **source**. If such a directory is found, it is used. Otherwise, the value is used as a relative pathname.

If **PROJECTDIR** is not set or has a null value, the current directory is searched first, followed by a search in the **SCCS** directory in the current directory.

The setting of **PROJECTDIR** affects all files listed in the remainder of this utility description for files with a component named **SCCS**.

International Code Set Support

Single and multi-byte character code sets are supported.

RETURN VALUES

make returns a 0 upon successful completion or a value greater than 0 if an error occurred. If the **-q** option is specified, **make** returns 0 if the target was up-to-date and a value greater than 0 if the target was not up-to-date.

EXAMPLES

The following example creates an executable file from a C source code file without a makefile, if program.c exists in the current directory:

```
make program
```

The following example shows more than one makefile specified and some command line macros defined, and updates the first target in module1:

```
make -f module1 -f module2 RELEASE=1.0 CFLAGS=-g
```

The following example updates two targets in a default makefile currently residing in the current directory:

```
make clobber prog
```

The following example updates the prog target in a specified makefile, allows environment variables to override any common variables in the makefile, clears the built-in suffix list and ignore the built-in rules, and outputs exhaustive debugging information:

```
make -erd -f module1 prog
```

WARNINGS

Be wary of any file (such as an include file) whose access, modification, and last change times cannot be altered by the **make**-ing process. For example, if a program depends on an include file that in turn depends on another include file, and if one or both of these files are out-of-date, **make** tries to update these files each time it is run, thus unnecessarily re-**make**ing up-to-date files that are dependent on the include file. The solution is to manually update these files with the **touch** command before running **make** (see *touch*(1)).

Some commands return non-zero status inappropriately; use **-i** to overcome the difficulty.

File names with the characters **= : @ $** do not work.

Built-in commands that are directly executed by the shell such as **cd** (see *cd*(1)), are ineffectual across new-lines in **make**.

The syntax **(lib(file1.o file2.o file3.o)** is illegal.

You cannot build **lib(file.o)** from **file.o**.

The macro **$(a:.o=.c˜)** does not work.

Expanded target lines cannot contain more than 16384 characters, including the terminating new-line.

If no makefile exists in the current directory, typing

```
make filename
```

results in **make** attempting to build **filename** from **filename.c**

If **make** is invoked in a shell script with a quoted argument that evaluates to NULL (such as **$@**), **make** fails.

DEPENDENCIES

NFS Warning:

When comparing modification times of files located on different NFS servers, **make** behaves unpredictably if the clocks on the servers are not synchronized.

FILES

[Mm]akefile
s.[Mm]akefile
SCCS/s.[Mm]akefile

m

SEE ALSO

cc(1), cd(1), lex(1), mkmf(1), sh(1), yacc(1), environ(5), lang(5), regexp(5).

A Nutshell Handbook, Managing Projects With Make by Steve Talbot, Second Edition, O'Reilly & Associates, Inc., 1986.

STANDARDS CONFORMANCE

make: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

NAME

makekey - generate encryption key

SYNOPSIS

/usr/lbin/makekey

DESCRIPTION

makekey improves the usefulness of encryption schemes depending on a key by increasing the amount of time required to search the key space. It reads 10 bytes from its standard input and writes 13 bytes on its standard output. The output depends on the input in a way intended to be difficult to compute (i.e., to require a substantial fraction of a second).

The first eight input bytes (the *input key*) can be arbitrary ASCII characters. The last two (the *salt*) are best chosen from the set of digits, **.**, **/**, and uppercase and lowercase letters. The salt characters are repeated as the first two characters of the output. The remaining 11 output characters are chosen from the same set as the salt and constitute the *output key*.

The transformation performed is essentially the following: the salt is used to select one of 4,096 cryptographic machines all based on the National Bureau of Standards DES algorithm, but broken in 4,096 different ways. Using the *input key* as key, a constant string is fed into the machine and recirculated a number of times. The 64 bits that come out are distributed into the 66 *output key* bits in the result.

makekey is intended for programs that perform encryption (e.g., *ed*(1) and *crypt*(1)). Usually, its input and output will be pipes.

SEE ALSO

crypt(1), ed(1), passwd(4).

NAME
man - find manual information by keywords; print out a manual entry

SYNOPSIS
man [**-M** path] **-k** *keyword*...

man [**-M** path] **-f** *file*...

man [**-**] [**-M** path] [**-T** macro-package] [*section* [*subsection*]] *entry_name*...

DESCRIPTION
man accesses information from the HP-UX manual pages. It can be used to:

- List all manual entries whose one-line description contains any of a specified set of keywords.
- Display or print one-line descriptions of entries specified by name.
- Search on-line manual directories by entry name and display or print the specified entry or entries.
- Search a specified on-line manual section (directory) and display or print the specified entry or entries in that section.

Searching for Entry Names by Keyword (first form)
The first form above searches the one-line descriptions of individual entries for specified keywords. Arguments are as follows:

-k *keyword* **-k** followed by one or more keywords causes **man** to print the one-line description of each manual entry whose one-line description contains text matching one or more of the specified keywords (similar to the behavior of *grep*(1)). Keywords are separated by blanks (space or tab).

Before this option can be used, file **/usr/share/lib/whatis** must exist. **/usr/share/lib/whatis** can be created by running *catman*(1M).

Obtaining One-Line Description of an Entry (second form)
The second form above finds and displays or prints the one-line descriptions of specified individual entries. Arguments are as follows:

-f *file* **-f** followed by one or more file names causes **man** to print the one-line description of each manual entry found whose name matches *file*. When specifying two or more files, *file* arguments are separated by blanks (space or tab). If entry names matching *file* exist in two or more sections, the one-line description of each matched file name is output.

Before this option can be used, file **/usr/share/lib/whatis** must exist. **/usr/share/lib/whatis** can be created by running *catman*(1M).

Viewing Individual Manual Entries (third form)
The third form shown above is used for viewing one or more individual manual entries. **man** in this form recognizes the following arguments:

- (optional) When the **-** argument is present, **man** sends the formatted manual entry directly to standard output without processing it through the output filter specified by the **PAGER** environment variable.

-M *path* Change the search path for manual pages. *path* is a colon-separated list of directories that contain manual page directory subtrees. When used with the **-k** or **-f** options, the **-M** option must appear first.

-T *macro-package*
man uses macro-package rather than the standard -man macros defined in **/usr/share/lib/tmac/tmac.an** for formatting manual pages.

When specifying the **-T** option to **man** , the full path must be given. For example:

```
man -T /usr/share/lib/tmac/tmac.s ls
```

section[*subsection*]
(optional) Search in the specified section for the given *entry_name*. *section* specifies a single section number or one of the words **local**, **new**, **old**, or **public** to search

m

for one or more of the entries indicated. *section* corresponds to the section number where the entry appears in the *HP-UX Reference*. It can be followed by an optional uppercase/lowercase subsection identifier such as **3C** which would indicate a library routine in Section 3. **3**, **3c**, and **3C** are interpreted as equivalent, since all Section 3 manual entries are stored in the same or in related directories (such as **/usr/share/man/man3.Z** and **/usr/share/man/man3**. However, if an entry is in Section 1M, *section* must be specified as **1m** or **1M**.

entry_name Search for a specific entry name where *entry_name* is the name of the manual entry without its section-number suffix. Except for names exceeding 11 characters, *entry_name* is identical to the name of the manual entry as listed at the top of each page, or is the same as one of the keywords in the left-hand part of the one-line description in the corresponding manual entry.

If *entry_name* is longer than 11 characters, **man** first searches for the full-length *entry_name*. If not found, *entry_name* is truncated to 11 characters to ensure that there is room for the *section* suffix in 14-character source file names. Files in the **/usr/share/man/**∗ directories are normally installed with the filename truncated to 11 characters where necessary so that the name plus a three-character section suffix does not exceed the maximum filename length on short filename systems.

If *section* is not specified (see next argument description), **man** searches all sections of the manual in order: **man1**, **man2**, **man1M**, **man3**, **man4**, **man5**, **man6**, **man7**, **man8**, **man9**, **manlocal**, **mannew**, **manold**, then **manpublic**; and printing the first matching entry it encounters.

If the standard output is a teletype, and if the **-** flag is not given, **man** pipes its output through **more** (see *more*(1)), with the **-s** option, to eliminate multiple blank lines and stop after each screenful. This default behavior can be changed by setting the **PAGER** variable in the user's environment. The value of **PAGER** must be a string that names an output filter (such as *pg*(1)), along with the desired options.

m

File Search Conventions

man searches in several directories, as appropriate, for the specified manual entry. The search continues until either the entry is found or all candidate directories are searched. The first three directories searched, in order, are: **/usr/share/man**, **/usr/contrib/man**, and **/usr/local/man**.

The **MANPATH** environment variable can be used to specify directories to be searched, and, if set, overrides the default paths given above. The **MANPATH** variable follows the same form as the **PATH** variable (see *environ*(5)).

Within each of these directories, **man** searches in the **cat**∗**.Z** subdirectories, the **man**∗**.Z** subdirectories, the **cat**∗ subdirectories, and the **man**∗ subdirectories. **man**∗**.Z** and **man**∗ directories contain *nroff*(1)-compatible source text for the entries. **cat**∗**.Z** and **cat**∗ directories contain the formatted versions of the entries. **man**∗**.Z** and **cat.Z** directories contain entries in compressed form. Files in these directories are uncompressed by **uncompress** (see *compress*(1)) before being processed for printing or display.

If the **LANG** environment variable is set to any valid language name defined by *lang*(5), and the **MANPATH** variable is not set, or is set to the default directories, **man** searches in three additional directories for the manual entry before searching in **/usr/share/man**. First, **man** searches in **/usr/share/man/$LANG**, then in **/usr/contrib/man/$LANG**, then in **/usr/local/man/$LANG**. Thus, native-language manual entries are displayed if they are present and installed properly in the system.

If the **MANPATH** environment variable is set to anything other than the default, the above directories with **$LANG** as part of the path are not automatically searched. All directories must be explicitly given in **MANPATH.** The **%L**, **%l**, **%t**, and **%c** specifiers can be used as path components to cause locale-specific directories to be searched. See *environ*(5) for a complete description of **MANPATH**.

man uses the most recent version that it finds in the subdirectories searched. If the most recent version is in:

man∗**.Z**	The entry is uncompressed, formatted, and displayed. If the **cat**∗**.Z** directory exists, the formatted entry is compressed and installed in **cat**∗**.Z**. If the **cat**∗ directory exists, the formatted entry is installed in **cat**∗.
cat∗**.Z**	The entry is uncompressed and displayed.

man*	The entry is formatted, and displayed. If the **cat*.Z** directory exists, it is compressed, and installed in **cat*.Z**. If the **cat*** directory exists, the formatted entry is installed in **cat***.
cat*	The entry is displayed.

If only the **cat*** or **cat*.Z** subdirectory is present and/or *nroff*(1) is not installed, only entries that are already formatted can be displayed.

If you choose to have the formatted entries on your system, run *catman*(1M) with the default, which creates the **cat*.Z** directories (after removing any **cat*** directories that exist on your system) and also creates the file **/usr/share/lib/whatis** used by the **man -k** option. If you choose to have the **cat*** directories, it would be space-saving to remove any **cat*.Z** directories that may exist on your system. Beware that **man** updates both directories (**cat*** and **cat*.Z**) if they both exist.

Special Manual Entries

Some situations may require creation of manual entries for local use or distribution by third-party software suppliers. The manual formatting macros have been structured to redefine page footers so that manual entries not originating from Hewlett-Packard Company do not show the **HP** name in the footer. For more information about this change and a description of the manual formatting macros used with **nroff** or **troff**, see *man*(5).

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages are displayed. **LANG** is also used to determine the search path (as described above).

If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG** for messages, but not for the search path.

If any internationalization variable contains an invalid setting, **man** behaves as if all internationalization variables are set to "C". See *environ*(5).

MANPATH, if set, gives a list of directories to be searched for the given entry, replacing the default paths.

PAGER, if set, defines an output filter to be used instead of *more*(1) to paginate output.

International Code Set Support

Single- and multi-byte character code sets are supported.

EXAMPLES

List the manual entries that contain the word **grep** in their respective one-line description (NAME) lines:

```
man -k grep
```

The output is:

```
grep, egrep, fgrep (1) - search a file for a pattern
zgrep(1)               - search possibly compressed files for a
                         regular expression
```

Print the one-line description of the *grep*(1) manual entry:

```
man -f grep
```

Print the entire *grep*(1) manual entry:

```
man grep
```

Set a search path that includes a path directly below the current directory. The manual entry, **mypage** is assumed to exist in the directory **./man1** (or **./man1.Z**, **cat1**, or **cat1.Z**).

```
MANPATH=.:/usr/share/man:/usr/contrib/man:/usr/local/man
export MANPATH
man mypage
```

Display the manual entry for *id*(1), with the output piped through **pg -c**:

```
PAGER="pg -c"
export PAGER
man id
```

List all printed manuals available for the current system (see *manuals*(5):

```
man manuals
```

WARNINGS

Manual entries are structured such that they can be printed on a phototypesetter, conventional line printer, and screen display devices. However, due to line printer and display device limitations, some information may be lost in certain situations.

FILES

/usr/share/lib/whatis	keyword database
/usr/share/man/cat*[.Z]/*	formatted manual entries [compressed]
/usr/share/man/man*[.Z]/*	raw (*nroff*(1) source) manual entries [compressed]
/usr/contrib/man/cat*[.Z]/*	
/usr/contrib/man/man*[.Z]/*	
/usr/local/man/cat*[.Z]/*	
/usr/local/man/man*[.Z]/*	
/usr/share/man/$LANG/cat*[.Z]/*	formatted native-language manual entries [compressed]
/usr/share/man/$LANG/man*[.Z]/*	raw (*nroff*(1) source) native-language manual entries [compressed]
/usr/contrib/man/$LANG/cat*[.Z]/*	
/usr/contrib/man/$LANG/man*[.Z]/*	
/usr/local/man/$LANG/cat*[.Z]/*	
/usr/local/man/$LANG/man*[.Z]/*	

SEE ALSO

col(1), compress(1), grep(1), more(1), catman(1M), fixman(1M), environ(5), man(5), manuals(5).

STANDARDS CONFORMANCE

man: XPG4

NAME

mediainit - initialize disk or cartridge tape media, partition DDS tape

SYNOPSIS

mediainit [**-vr**] [**-f** *fmt_optn*] [**-i** *interleave*] [**-p** *size*] *pathname*

DESCRIPTION

mediainit initializes mass storage media by formatting the media, writing and reading test patterns to verify media integrity, then sparing any defective blocks found. This process prepares the disk or tape for error-free operation. Initialization destroys all existing user data in the area being initialized.

mediainit can also used for partitioning DDS tape media. See the **-p** option below for further details.

Options

The following command options are recognized. They can be specified in any order, but all must precede the *pathname*. Options without parameters can be listed individually or grouped together. Options with parameters must be listed individually, but white space between the option and its parameter is discretionary.

-v Normally, **mediainit** provides only fatal error messages which are directed to standard error. The **-v** (verbose) option sends device-specific information related to low-level operation of **mediainit** to standard output (stdout). This option is most useful to trained service personnel because it usually requires detailed knowledge of device operation before the information can be interpreted correctly.

-r (re-certify) This option forces a complete tape certification whether or not the tape has been certified previously. All record of any previously spared blocks is discarded, so any bad blocks will have to be rediscovered. This option should be used only if:

- It is suspected that numerous blocks on the tape have been spared which should not have been, or
- It is necessary to destroy (overwrite) all previous data on the tape.

-f *fmt_optn* The format option is a device-specific number in the range **0** through **239**. It is intended solely for use with certain SS/80 devices that support multiple media formats (independent from interleave factor). For example, certain microfloppy drives support 256-, 512-, and 1024-byte sectors. **mediainit** passes any supplied format option directly through to the device. The device then either accepts the format option if it is supported, or rejects it if it is not supported. Refer to device operating manuals for additional information. The default format option is **0**.

-i *interleave* The interleave factor, *interleave*, refers to the relationship between sequential logical records and sequential physical records. It defines the number of physical records on the media that lie between the beginning points of two consecutively numbered logical records. The choice of interleave factor can have a substantial impact on disk performance. For CS/80 and SS/80 drives, consult the appropriate device operating manual for details. For Amigo drives, see WARNINGS.

-p *size* Partition DDS cartridge media into two logical separate volumes: partition 0 and partition 1:

- *size* specifies the minimum size of partition 1 (in Mbytes). The maximum allowed value is 1200.
- Partition 0 is the remainder of the tape (partition 0 physically follows partition 1 on the tape).

The actual size of partition 1 is somewhat larger than the requested size to allow for tape media errors during writing. Thus, a *size* of 400 formats the DDS tape into two partitions where partition 1 holds at least 400 Megabytes of data, and the remainder of the tape is used for partition 0 (for a 1300 Mbyte DDS cartridge, this means that partition 0 has a size somewhat less than 900 Mbytes).

Note that it is unnecessary to format a DDS tape before use unless the tape is being partitioned. Unformatted DDS media does not require initialization when used as a single partition tape. Accessing partition 1 on a single-partition tape produces an error. To change a two-partition tape to single-partition, use **mediainit** with **0** specified as the *size*.

pathname — *pathname* is the path name to the character (raw) device special file associated with the device unit or volume that is to be initialized. **mediainit** aborts if you lack either read or write permission to the device special file, or if the device is currently open for any other process. This prevents accidental initialization of the root device or any mounted volume. **mediainit** locks the unit or volume being initialized so that no other processes can access it.

Except for SCSI devices, *pathname* must be a device special file whose minor number of the device being initialized has the diagnostic bit set. For device special files with the diagnostic bit set, the section number is meaningless. The entire device is accessed.

When a given CS/80 or SS/80 device contains multiple units, or a given unit contains multiple volumes as defined by the drive controller, any available unit or volume associated with that controller can be initialized, independent of other units and volumes that share the same controller. Thus, you can initialize one unit or volume to any format or interleave factor without affecting formats or data on companion units or volumes. However, be aware that the entire unit or volume (as defined by the drive controller) is initialized without considering the possibility that it may be subdivided into smaller structures by the the operating software. When such structures exist, unexpected loss of data is possible.

mediainit dominates controller resources and limits access by competing processes to other units or volumes sharing the same controller. If other simultaneous processes need access to the same controller, some access degradation can be expected until initialization is complete; especially if you are initializing a tape cartridge in a drive that shares the root disk controller.

In general, **mediainit** attempts to carefully check any **-f** (format option) or **-i** (interleave options) supplied, and aborts if an option is out of range or inappropriate for the media being initialized. Specifying an interleave factor or format option value of **0** has the same effect as not specifying the option at all.

m

For disks that support interleave factors, the acceptable range is usually **1** (no interleave) through $n-1$, where n is the number of sectors per track. With SS/80 hard disks, the optimum interleave factor is usually determined by the speed (normal or high) of the HP-IB interface card used and whether DMA is present in the system. The optimum interleave factor for SS/80 flexible disk drives is usually a constant (often **2**), and is independent of the type of HP-IB interface used. The optimum interleave factor for CS/80 disks is usually **1** and is also usually not related to the type of HP-IB interface being used. In any case, refer to the appropriate device operating manual for recommended values.

If a disk being initialized requires an interleave factor but none is specified, **mediainit** provides an appropriate, though not necessarily optimum default. For CS/80 and SS/80 disks, **mediainit** uses whatever the device reports as its current interleave factor. SS/80 floppy drives report their minimum (usually best) interleave factor, if the currently installed media is unformatted.

When a given device supports format options, the allowable range of interleave factors may be related to the specified format option. In such instances, **mediainit** cannot check the interleave factor if one is specified.

Notes

Most types of mass storage media must be initialized before they can be used. HP hard disks, flexible disks, and cartridge tapes require some form of initialization, but 9-track tapes do not. Initialization usually involves formatting the media, writing and reading test patterns, then sparing any defective blocks. Depending upon the media and device type, none, some, or all of the initialization process may have been performed at the factory. **mediainit** completes whatever steps are appropriate to prepare the media for error-free operation.

Most HP hard disks are formatted and exhaustively tested at the factory by use of a process more thorough but also more time-consuming than appropriate for **mediainit**. However, **mediainit** is still valuable for ensuring the integrity of the media after factory shipment, formatting with the correct interleave factor, and sparing any blocks which may have become defective since original factory testing was performed.

HP flexible disks are not usually formatted prior to shipment, so they must undergo the entire initialization process before they can be used.

All HP CS/80 cartridge tapes are certified and formatted prior to shipment from the factory. When a tape is certified, it is thoroughly tested and defective blocks are spared. **mediainit** usually certifies a tape only if it has not been certified previously. If the tape has been previously certified and spared, **mediainit** usually reorganizes the tape's spare block table, retaining any previous spares, and optimizing their assignment for maximum performance under sequential access. Reorganizing the spare block table takes only a few seconds, whereas complete certification takes about a half-hour for 150-foot tapes, and over an hour for 600-foot tapes.

HP CS/80 cartridge tape drives have a feature called "auto-sparing". If under normal usage the drive has trouble reading a block, the drive logs the fact and automatically spares out that block the next time data is written to it. Thus, as a tape is used, any marginal blocks that were not spared during certification are spared automatically if they cause problems. This sparing is automatic within the device, and is totally independent of **mediainit**.

Reorganization of a tape's spare block table technically renders any existing data undefined, but the data is not usually destroyed by overwriting. To ensure that old tape data is destroyed, which is useful for security, complete tape re-certification can be forced with the **-r** option.

Some applications may require that a file system be placed on the media before use. **mediainit** does not create a file system; it only prepares media for writing and reading. If such a file system is required, other utilities such as **newfs**, **lifinit**, or **mkfs** must be invoked after running **mediainit** (see *newfs*(1M), *lifinit*(1), and *mkfs*(1M)).

RETURN VALUE

mediainit returns one of the following values:

0	Successful completion.
1	A device-related error occurred.
2	A syntax-related error was encountered.

ERRORS

Appropriate error messages are printed on standard error during execution of **mediainit**.

EXAMPLES

Format an HP 9122 SS/80 3-1/2-inch flexible disk with an interleave factor of 2, 1024-byte sectors, and double-sided HP format:

```
mediainit -i 2 -f 3 /dev/rdsk/9122
```

WARNINGS

For a device that contains multiple units on a single controller, each unit can be initialized independently from any other unit. It should be noted, however, that **mediainit** requires that there be no other processes accessing the device before initialization begins, regardless of which unit is being initialized. If there are accesses currently in progress, **mediainit** aborts.

Aborting **mediainit** is likely to leave the medium in a corrupt state, even if it was previously initialized. To recover, the initialization must be restarted.

During the initialization process, **open()** rejects all other accesses to the device being initialized, producing the error EACCES (see *open*(2)).

DEPENDENCIES

Series 800

Partitioning of DDS tape media (**-p** option) is not supported.

AUTHOR

mediainit was developed by HP.

SEE ALSO

lifinit(1), mkfs(1M), newfs(1M).

NAME

merge - three-way file merge

SYNOPSIS

merge [**-p**] *file1 file2 file3*

DESCRIPTION

merge combines two files that are revisions of a single original file. The original file is *file2*, and the revised files are *file1* and *file3*. **merge** identifies all changes that lead from *file2* to *file3* and from *file2* to *file1*, then deposits the merged text into *file1*. If the **-p** option is used, the result goes to standard output instead of *file1*.

An overlap occurs if both *file1* and *file3* have changes in the same place. **merge** prints how many overlaps occurred, and includes both alternatives in the result. The alternatives are delimited as follows:

```
<<<<<<< file1
lines in file1
=======
lines in file3
>>>>>>> file3
```

If there are overlaps, edit the result in *file1* and delete one of the alternatives.

This command is particularly useful for revision control, especially if *file1* and *file3* are the ends of two branches that have *file2* as a common ancestor.

EXAMPLES

A typical use for **merge** is as follows:

1. To merge an RCS branch into the trunk, first check out the three different versions from RCS (see *co*(1)) and rename them for their revision numbers: 5.2, 5.11, and 5.2.3.3. File 5.2.3.3 is the end of an RCS branch that split off the trunk at file 5.2.
2. For this example, assume file 5.11 is the latest version on the trunk, and is also a revision of the "original" file, 5.2. Merge the branch into the trunk with the command:

   ```
   merge 5.11 5.2 5.2.3.3
   ```
3. File 5.11 now contains all changes made on the branch and the trunk, and has markings in the file to show all overlapping changes.
4. Edit file 5.11 to correct the overlaps, then use the **ci** command to check the file back in (see *ci*(1)).

WARNINGS

merge uses the *ed*(1) system editor. Therefore, the file size limits of *ed*(1) apply to **merge**.

AUTHOR

merge was developed by Walter F. Tichy.

SEE ALSO

diff3(1), diff(1), rcsmerge(1), co(1).

NAME

mesg - permit or deny messages to terminal

SYNOPSIS

mesg [[**-**] **g**] [[**-**] **y**] [[**-**] **n**]
mesg

DESCRIPTION

The command form **mesg** [**-**] **n** forbids messages via **write** by revoking write permission to users without appropriate privilege on the user's terminal (see *write*(1)). The command form **mesg** [**-**] **g** reinstates permission so that only legitimate commands (such as *write*(1)) can be used by other users to send messages. **mesg** [**-**] **y** allows applications such as **write** or **talk** to send messages to the user's terminal (that is, without restrictions). **mesg** without any other argument reports the current state without changing it.

RETURN VALUE

mesg returns the following values:

0	Messages are receivable.
1	Messages are not receivable.
2	An error occurred.

EXTERNAL INFLUENCES

Environment Variables

LC_MESSAGES determines the language in which messages are displayed.

If **LC_MESSAGES** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**.

If any internationalization variable contains an invalid setting, **mesg** behaves as if all internationalization variables are set to "C". See *environ*(5).

FILES

/dev/tty*

SEE ALSO

write(1).

STANDARDS CONFORMANCE

mesg: SVID2, SVID3, XPG2, XPG3, XPG4

NAME

mkdir - make a directory

SYNOPSIS

mkdir [**-p**] [**-m** *mode*] *dirname* ...

DESCRIPTION

mkdir creates specified directories in mode 0777 (possibly altered by **umask** unless specified otherwise by a **-m** *mode* option (see *umask*(1)). Standard entries, **.** (for the directory itself) and **..** (for its parent) are created automatically. If *dirname* already exists, **mkdir** exits with a diagnostic message, and the directory is not changed.

Options

mkdir recognizes the following command-line options:

-m *mode* After creating the directory as specified, the file permissions are set to *mode*, which is a symbolic mode string as defined for **chmod** (see *chmod*(1)). The *umask*(1) has precedence over **-m**.

-p Intermediate directories are created as necessary. Otherwise, the full path prefix of *dirname* must already exist. **mkdir** requires write permission in the parent directory.

For each directory name in the pathname prefix of the *dirname* argument that is not the name of an existing directory, the specified directory is created using the current **umask** setting, except that the equivalent of **chmod u+wx** is done on each component to ensure that **mkdir** can create lower directories regardless of the setting of **umask**. Each directory name in the pathname prefix of the *dirname* argument that matches an existing directory is ignored without error. If an intermediate path component exists, but has permissions set to prevent writing or searching, **mkdir** fails with an error message.

If the **-m** option is used, the directory specified by *dirname* (excluding directories in the pathname prefix) is created with the permissions specified by *mode*.

Only **LINK_MAX** subdirectories can be created (see *limits*(5)).

m

EXTERNAL INFLUENCES

Environment Variables

LANG provides a default value for the internationalization variables that are unset or null. If **LANG** is unset or null, the default value of "C" (see *lang*(5)) is used. If any of the internationalization variables contains an invalid setting, **mkdir** will behave as if all internationalization variables are set to "C". See *environ*(5).

LC_ALL If set to a non-empty string value, overrides the values of all the other internationalization variables.

LC_CTYPE determines the interpretation of text as single and/or multi-byte characters, the classification of characters as printable, and the characters matched by character class expressions in regular expressions.

LC_MESSAGES determines the locale that should be used to affect the format and contents of diagnostic messages written to standard error and informative messages written to standard output.

NLSPATH determines the location of message catalogues for the processing of **LC_MESSAGES.**

International Code Set Support

Single- and multi-byte character code sets are supported.

DIAGNOSTICS

mkdir returns exit code 0 if all directories were successfully made. Otherwise, it prints a diagnostic and returns non-zero.

EXAMPLES

Create directory **gem** beneath existing directory **raw** in the current directory:

```
    mkdir raw/gem
```

Create directory path **`raw/gem/diamond`** underneath the current directory and set permissions on directory **`diamond`** to read-only for all users (**`a=r`**):

```
    mkdir -p -m "a=r" raw/gem/diamond
```

which is equivalent to (see *chmod*(1)):

```
    mkdir -p -m 444 raw/gem/diamond
```

If directories **`raw`** or **`raw`** and **`gem`** already exist, only the missing directories in the specified path are created.

SEE ALSO

rm(1), sh(1), umask(1).

STANDARDS CONFORMANCE

`mkdir`: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

NAME

mkfifo - make FIFO (named pipe) special files

SYNOPSIS

mkfifo [**-p**] [**-m** *mode*] *filename* ...

DESCRIPTION

mkfifo creates the FIFO special files named by its operand list. The operands are taken sequentially in the order specified and, if the user has write permission in the appropriate directory, the FIFO is created with permissions 0666 modified by the user's file mode creation mask (see *umask*(2)).

The specific actions performed are equivalent to calling

mkfifo(*filename***, 0666)**

for each filename in the operand list (see *mkfifo*(2)).

Options

mkfifo recognizes the following command-line options:

-m *mode* After creating the FIFO special file, set the permission bits of the new file to the specified *mode* value. The *mode* option argument is a symbolic mode string as defined in *chmod*(1).

(XPG4 Only.) In the symbolic mode strings, the operators **+** and **-** will be interpreted relative to an initial mode of a=rw.

-p Create any missing intermediate path name components.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the locale to use for the locale categories when both **LC_ALL** and the corresponding environment variable (beginning with **LC_**) do not specify a locale. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used.

LC_ALL determines the locale to use to override any values for locale categories specified by the settings of **LANG** or any environment variables beginning with **LC_**.

LC_CTYPE determines the locale for the interpretation of sequences of bytes of text data as characters (e.g., single- versus multibyte characters in arguments).

If any internationalization variable contains an invalid setting, **mkfifo** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single-byte character code sets are supported.

RETURN VALUE

mkfifo returns zero if invoked with at least one operand and if all FIFO special files were created successfully. Otherwise, it prints a diagnostic message and returns non-zero.

EXAMPLES

The following command creates a FIFO special file named **peacepipe** in the current directory:

```
mkfifo peacepipe
```

SEE ALSO

chmod(1), umask(1), mknod(1M), mkfifo(3C).

STANDARDS CONFORMANCE

mkfifo: XPG3, XPG4, POSIX.2

NAME

mkmf - make a makefile

SYNOPSIS

mkmf [**-acdeil**] [**-f** *makefile*] [**-F** *template*] [**-M** *language*] [*macroname*=*value* ...]

DESCRIPTION

The **mkmf** command creates a makefile that informs the **make** command how to construct and maintain programs and libraries (see *make*(1)). After gathering up all source code file names in the current working directory and inserting them into the makefile, **mkmf** scans source code files for included files and generates dependency information that is appended to the makefile. Source code files are identified by their file name suffixes. **mkmf** recognizes the following suffixes:

.c	C
.C	C++
.f	FORTRAN
.h	Include files
.i	Pascal include files
.l	Lex or Lisp
.o	Object files
.p	Pascal
.r	Ratfor
.s	Assembler
.y	Yacc

The **mkmf** command checks for an existing makefile before creating one. If no **-f** option is present, **mkmf** tries the makefiles **makefile** and **Makefile**, respectively.

After the makefile has been created, arbitrary changes can be made using a text editor. **mkmf** can also be used to re-edit the macro definitions in the makefile, regardless of changes that may have been made since it was created.

By default, **mkmf** creates a program makefile. To create a makefile that handles libraries, the **-l** option must be used.

Make Requests

Given a makefile created by **mkmf**, **make** recognizes the following requests:

all	Compile and load a program or library.
clean	Remove all object and core files.
clobber	Remove all files that can be regenerated.
depend	Update included file dependencies in a makefile.
echo	List the names of the source code files on standard output.
extract	Extract all object files from the library and place them in the same directory as the source code files. The library is not altered.
index	Print an index of functions on standard output.
install	Compile and load the program or library and move it to its destination directory.
print	Print source code files on standard output.
tags	Create a tags file for the **ex** editor (see *ex*(1) and *ctags*(1)), for C, Pascal, and Fortran source code files.
update	Recompile only if there are source code files that are newer than the program or library, link and install the program or library.

Several requests can be given simultaneously. For example, to (1) compile and link a program, (2) move the program to its destination directory, and (3) remove any unnecessary object files, use:

```
make install clean
```

Macro Definitions

mkmf understands the following macro definitions:

CFLAGS	C compiler flags. After searching for included files in the directory currently being processed, **mkmf** searches in directories named in **-I** compiler options and then in the **/usr/include** directory.
COMPILESYSTYPE	Location of **/usr/include**. If the **COMPILESYSTYPE** macro or environment variable is defined, **mkmf** searches for included files in **/$COMPILESYSTYPE/usr/include** instead of **/usr/include**.
CXXFLAGS	C++ compiler flags. After searching for included files in the directory currently being processed, **mkmf** searches in directories named in **-I** compiler options and then in the **/usr/include/CC** directory, followed by the **/usr/include** directory.
DEST	Directory where the program or library is to be installed.
EXTHDRS	List of included files external to the current directory. **mkmf** automatically updates this macro definition in the makefile if dependency information is being generated.
FFLAGS	Fortran compiler flags. After searching for included files in the directory currently being processed, **mkmf** searches in directories named in **-I** compiler options, then in the **/usr/include** directory.
HDRS	List of included files in the current directory. **mkmf** automatically updates this macro definition in the makefile.
INSTALL	Installation program name.
LD	Link editor name.
LDFLAGS	Link editor flags.
LIBRARY	Library name. This macro also implies the **-l** option.
LIBS	List of libraries needed by the link editor to resolve external references.
MAKEFILE	Makefile name.
OBJS	List of object files. **mkmf** automatically updates this macro definition in the makefile.
PROGRAM	Program name.
SRCS	List of source code files. **mkmf** automatically updates this macro definition in the makefile.
SUFFIX	List of additional file name suffixes for **mkmf** to know about.
SYSHDRS	List of included files found in the **/usr/include** directory hierarchy. **mkmf** automatically updates this macro definition in the makefile if dependency information is being generated. If **SYSHDRS** is omitted from the makefile, **mkmf** does not generate **/usr/include** dependencies.

Both these and any other macro definitions already within the makefile can be replaced by definitions on the command line in the form *macroname***=***value*. For example, to change the C compiler flags and the program name, type the following line:

```
mkmf "CFLAGS=-I../include -O" PROGRAM=mkmf
```

Note that macro definitions such as **CFLAGS** with blanks in them must be enclosed in double quote (") marks.

Environment

The environment is read by **mkmf**. All variables are assumed to be macro definitions with the exception of **HDRS**, **EXTHDRS**, **SRCS**, and **OBJS**. Environment variables are processed after command line macro definitions and the macro definitions in a *makefile*. The **-e** option forces the environment to override the macro definitions in a *makefile*.

File Name Suffixes

mkmf can recognize additional file name suffixes, or ignore ones that it already recognizes, by specifying suffix descriptions in the **SUFFIX** macro definition. Each suffix description takes the form **.***suffix***:***tI* where *t* is a character indicating the contents of the file (**s** = source file, **o** = object file, **h** = header file, **x** = executable file) and *I* is an optional character indicating the include syntax for header files (**C** = C syntax, **C++** = C syntax plus the addition of **/usr/include/CC** as a standard search directory, **F** = Fortran and

Ratfor syntax, **P** = Pascal syntax). The following list shows the default configuration for **mkmf:**

.c:sC	C
.C:sC++	C++
.f:sF	Fortran
.h:h	Include files
.i:h	Pascal include files
.l:sC	Lex or Lisp
.o:o	Object files
.p:sP	Pascal
.r:sF	Ratfor
.s:s	Assembler
.y:sC	Yacc

For example, to change the object file suffix to **.obj**, undefine the Pascal include file suffix, and prevent Fortran files from being scanned for included files, the **SUFFIX** macro definition could be:

```
SUFFIX = .obj:o .i: .f:s
```

Include Statement Syntax

The syntax of include statements for C, C++, Fortran, and Pascal source code are of the form:

C/C++:

#include "*filename***"**
#include *filename*

where **#** must be the first character in the line.

Fortran:

$include ′ *filename* **′$**
$INCLUDE ′ *filename* **′$**

where **$** must be the first character in the line. Alternatively, the **$** can be omitted if the include statement starts in column 7. In either case the trailing **$** can be omitted.

Pascal:

$include ′ *filename* **′$**
$INCLUDE ′ *filename* **′$**

where **$** must be the first character in the line and the trailing **$** is optional.

User-defined Templates

If **mkmf** cannot find a makefile within the current directory, it normally uses one of the standard makefile templates, **C.p** or **C.l**, in **/usr/ccs/lib/mf** unless the user has alternative **C.p** or **C.l** template files in a directory **$PROJECT/lib/mf** where **$PROJECT** is the absolute path name of the directory assigned to the **PROJECT** environment variable.

Options

mkmf recognizes the following options:

-a Include source files beginning with a **.** in the makefile.

-c Suppress "**creating** *makefile* **from** ..." message.

-d Turn off scanning of source code for **include** files. Old dependency information is left untouched in the makefile.

-e Environment variables override macro definitions within *makefile*s.

-f *makefile* Specify an alternative *makefile* file name. The default file name is **Makefile**.

-i Prompt the user for the name of the program or library and the directory where it is to be installed. If a carriage-return is typed in response to each of these queries, **mkmf** assumes that the default program name is **a.out** or the default library name is **lib.a**, and the destination directory is the current directory.

-l Force the makefile to be a library makefile.

-F *template* Specify an alternative makefile template path name. The path name can be relative or absolute.

-M *language* Specify an alternative *language*-specific makefile template. The default language is C and the corresponding program and library makefile templates are **C.p** and **C.l**, respectively. **mkmf** looks for these templates in **/usr/ccs/lib/mf** or **$PROJECT/lib/mf**.

DIAGNOSTICS

Exit status 0 is normal. Exit status 1 indicates an error.

WARNINGS

The name of the makefile is included as a macro definition within the makefile and must be changed if the makefile is renamed.

Since executable files are dependent on libraries, standard library abbreviations must be expanded to full path names within the **LIBS** macro definition in the makefile.

Generated dependency information appears after a line in the makefile beginning with ###. This line must not be removed, nor must any other information be inserted in the makefile below this line.

The name of a program or library must not conflict with any predefined target names in a makefile. It is especially important to avoid the the name **update** to prevent **make** from recursively executing itself an infinite number of times.

AUTHOR

mkmf was developed by the University of California, Berkeley.

FILES

/usr/ccs/lib/mf/C.p	Standard program makefile template
/usr/ccs/lib/mf/C.l	Standard library makefile template
$PROJECT/lib/mf/C.p	User-defined program makefile template
$PROJECT/lib/mf/C.l	User-defined library makefile template

SEE ALSO

ar(1), ctags(1), ld(1), make(1).

"Automatic Generation of Make Dependencies", *Software-Practice and Experience*, Walden, K., vol. 14, no. 6, pp. 575-585, June 1984.

NAME

mkmsgs - create message files for use by gettxt()

SYNOPSIS

mkmsgs [**-o**] [**-i** *locale*] *textfile msgfile*

DESCRIPTION

The **mkmsgs** command takes as input a file of localized text strings and generates a message file that can be accessed by the *gettxt*(3C) routine. *textfile* is the name of the file that contains the text strings. *msgfile* is the name of the output message file. **mkmsgs** appends the suffix **.cat** to the message file name. The combined length of the file name should be less than 14 bytes for short file name file system. The *msgfile* file should not contain a colon since it will confuse the formatting routines.

The *textfile* file contains the localized text strings. The text strings are separated by a newline character. The text strings are processed sequentially and copied to the *msgfile* message file. An empty line in the input results in a corresponding empty message written to the *msgfile* message file.

Options

The **mkmsgs** command supports the following options:

-o	Overwrite the *msgfile* message file if it exists.
-i *locale*	The *msgfile* message file is installed in the system-wide localization directory corresponding to the specified *locale*. Only a user with the appropriate privileges can create or overwrite the message file in that directory. The directory will be created if it does not exist.

EXTERNAL INFLUENCES

Environment Variables

LC_CTYPE determines the interpretation of messages as single- and/or multibyte characters.

Messages are issued in **LANG** if it is set to a valid language and **LANG** messages are available. Otherwise "C" locale messages are issued.

If **LC_CTYPE** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**. If any internationalization variable contains an invalid setting, **mkmsgs** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multibyte character code sets are supported.

EXAMPLES

The following example shows the format of the input text strings:

```
global %s not found\n
\n\n<press return to continue>\n\n
\t%s, %d, %d,typ = %d, disp = '%s'\n
```

WARNINGS

mkmsgs is provided for SVID3 compatibility only. The user is encouraged to use the NLS mechanism developed by HP and the X/Open Company, Ltd.

SEE ALSO

gencat(1), gettxt(3C), setlocale(3C).

STANDARDS COMPLIANCE

mkmsgs: SVID3

NAME

mkstr - extract error messages from C source into a file

SYNOPSIS

mkstr [**-**] *messagefile prefix file ...*

DESCRIPTION

mkstr examines a C program and creates a file containing error message strings used by the program. Programs with many error diagnostics can be made much smaller by referring to places in the file, and reduce system overhead in running the program.

mkstr processes each of the specified *file*s, placing a revised version of each in a file whose name consists of the specified *prefix* concatenated in front of the original name. A typical usage of *mkstr* would be

```
mkstr mystrings xx *.c
```

This command would cause all the error messages from the C source files in the current directory to be placed in the file *mystrings* and revised copies of the source for these files to be placed in files whose names are prefixed with *xx*.

When processing the error messages in the source for transfer to the message file, **mkstr** searches for the string **error(** in the input file. Each time it is encountered, the C string starting after the leading quote is placed in the message file, followed by a null character and a new-line character. The null character terminates the message so that it can be easily used when retrieved, and the new-line character makes it possible to conveniently list the error message file (using **cat**, **more**, etc. — see *cat*(1) and *more*(1)) to review its contents.

The modified copy of the input file is identical to the original, except that each occurrence of any string that was moved to the error message file is replaced by an offset pointer usable by **lseek** to retrieve the message.

If the command line includes the optional **-**, extracted error messages are placed at the end of the specified message file (append) instead of overwriting it. This enables you to process individual files that are part of larger programs that have been previously processed by **mkstr** without reprocessing all the files.

All functions used by the original program whose names end in "error" that also can take a constant string as their first argument should be rewritten so that they search for the string in the error message file.

For example, a program based on the previous example usage would resemble the following:

```
#include <stdio.h>
#include <sys/types.h>
#include <fcntl.h>

char errfile[] = "mystrings" ;

error(offset, a2, a3, a4)
int offset, a1, a2, a3;
{
    char msg[256];
    static int fd = -1;

    if (fd < 0) {
        fd = open(errfile, O_RDONLY);
        if (fd < 0) {
     perror(errfile);
     exit(1);
        }
    }

    if (lseek(fd, (off_t) offset, 0) || read(fd, msg, 256) <= 0) {
        printf("? Can't find error message in %s:\n", errfile);
        perror(errfile);
        exit(1);
    }

    printf(msg, a1, a2, a3);
}
```

m

EXTERNAL INFLUENCES

Environment Variables

LC_CTYPE determines the interpretation of comments and string literals as single- and/or multi-byte characters.

If **LC_CTYPE** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**. If any internationalization variable contains an invalid setting, **mkstr** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported within file names, comments, and string literals.

SEE ALSO

lseek(2), perror(3C), xstr(1).

BUGS

Strings in calls to functions whose names end in **error**, notably **perror()**, may be replaced with offsets by **mkstr**.

Calls to error functions whose first argument is not a string constant are left unmodified without warning.

NAME

mktemp - make a name for a temporary file

SYNOPSIS

mktemp [**-c**] [**-d** *directory_name*] [**-p** *prefix*]

DESCRIPTION

mktemp makes a name that is suitable for use as the pathname of a temporary file, and writes that name to the standard output. The name is chosen such that it does not duplicate the name of an existing file. If the **-c** option is specified, a zero-length file is created with the generated name.

The name generated by **mktemp** is the concatenation of a directory name, a slash (**/**), the value of the **LOGNAME** environment variable truncated to **{NAME_MAX}** − 6 characters, and the process ID of the invoking process.

The directory name is chosen as follows:

1. If the **-d** option is specified, *directory_name* is used.
2. Otherwise, if the **TMPDIR** environment variable is set and a string that would yield a unique name can be obtained by using the value of that variable as a directory name, this value is used.
3. Otherwise, if a string that would yield a unique name can be obtained using **/tmp** as the directory, **/tmp** is used.
4. Otherwise, **.** (current directory) is used.

If the **-p** option is specified, *prefix* is used instead of the value of the **LOGNAME** environment variable for name generation.

RETURN VALUE

mktemp returns zero on successful completion and non-zero if syntax, file access, or file creation errors were encountered or a unique pathname could not be generated.

m

SEE ALSO

mktemp(3C), umask(1).

NAME

mm, osdd - print documents formatted with the mm macros

SYNOPSIS

mm [*options*] [*files*]

osdd [*options*] [*files*]

DESCRIPTION

mm can be used to format and print documents using **nroff** and the **mm** text-formatting macro package (see *nroff*(1)). It has options to specify preprocessing by **tbl** and/or **neqn**, (see *tbl*(1) and *neqn*(1)), and postprocessing by various terminal-oriented output filters. The proper pipelines and the required arguments and flags for **nroff** and **mm** are generated, depending on the options selected.

osdd is equivalent to the command **mm -mosd**.

Options

mm recognizes the following *options* and command-line arguments. Any other arguments or options (such as **-rC3**) are passed to **nroff** or to **mm**, as appropriate. Such options can occur in any order, but they must appear before the *files* arguments. If no arguments are given, **mm** prints a list of its options.

-T*term* Specifies the type of output terminal; for a list of recognized values for *term*, type **help term2**. If this option is *not* used, **mm** uses the value of the shell variable **$TERM** from the environment (see *profile*(4) and *environ*(5)) as the value of *term* if **$TERM** is set; otherwise, **mm** uses **450** as the value of *term*. If several terminal types are specified, the last one is used.

-12 Indicates that the document is to be produced in 12-pitch. Can be used when **$TERM** is set to one of **300**, **300s**, **450**, and **1620**. (The pitch switch on the DASI 300 and 300s terminals must be manually set to **12** if this option is used.)

-c Causes **mm** to invoke *col*(1); note that *col*(1) is invoked automatically by **mm** unless *term* is one of **300**, **300s**, **450**, **37**, **4000a**, **382**, **4014**, **tek**, **1620**, and **X**.

-e Causes **mm** to invoke **neqn**.

-t Causes **mm** to invoke **tbl**.

-E Invokes the **-e** option of **nroff**.

DIAGNOSTICS

mm sends the message **mm: no input file** if none of the arguments is a readable file and **mm** is not used as a filter.

EXAMPLES

Assuming that the shell variable **$TERM** is set in the environment to **450**, the two command lines below are equivalent:

```
mm -t -rC3 -12 ghh*
tbl ghh* | nroff -cm -T450-12 -h -rC3
```

mm reads the standard input when **-** is specified instead of any file names (mentioning other files along with **-** leads to disaster). This option allows **mm** to be used as a filter, as in this example:

```
cat dws | mm -
```

Hints

- **mm** invokes **nroff** with the **-h** option. With this option, **nroff** assumes that the terminal has tabs set every 8 character positions.
- Use the **-o***list* option of **nroff** to specify ranges of pages to be output. Note, however, that **mm**, if invoked with one or more of the **-e**, **-t**, and **-** options, *together* with the **-o***list* option of **nroff** may cause a harmless "broken pipe" diagnostic if the last page of the document is not specified in *list*.
- If you use the **-s** option of **nroff** (to stop between pages of output), use line-feed (rather than return or new-line) to restart the output. The **-s** option of **nroff** does not work with the **-c** option of **mm**, or if **mm** automatically invokes **col** (see **-c** option above and *col*(1)).

- If you specify an incorrect output terminal type, **mm** produces (often subtle) unpredictable results. However, if you are redirecting output into a file, use the **-T37** option, then use the appropriate terminal filter when actually printing the formatted file.

SEE ALSO

col(1), env(1), nroff(1), tbl(1), profile(4), mm(5), term(5).

mm section in *Text Formatting: User's Guide*.

NAME

model - print detailed hardware model information

SYNOPSIS

model

DESCRIPTION

model prints the machine hardware model. Its output is similar to that of **uname -m** with possible additional information. There are several systems for which **uname -m** returns the same value. The *model*(1) command can be used to distinguish between the different systems.

EXAMPLES

Executing the command **model** produces output resembling the following:

```
9000/715/50
```

This example indicates an HP 9000 Model 715 with a 50 MHz clock.

SEE ALSO

uname(1).

NAME

more, page - file perusal filter for crt viewing

SYNOPSIS

more [**-n**] [**-cdefisuvz**] [**-n** *number*] [**-p** *command*] [**-t** *tagstring*] [**-x** *tabs*] [**-W** *option*] [**+***linenumber*] [**+/***pattern*] [*name* ...]

page [**-n**] [**-cdefisuvz**] [**-n** *number*] [**-p** *command*] [**-t** *tagstring*] [**-x** *tabs*] [**-W** *option*] [**+***linenumber*] [**+/***pattern*] [*name* ...]

REMARKS:

pg is preferred in some standards and has some added functionality, but does not support character highlighting (see *pg*(1)).

DESCRIPTION

more is a filter for examining continuous text, one screenful at a time, on a soft-copy terminal. It is quite similar to **pg**, and is retained primarily for backward compatibility. **more** normally pauses after each screenful, printing the filename at the bottom of the screen. To display one more line, press **<Return>**. To display another screenful press **<Space>**. Other possibilities are described later.

more and **page** differ only slightly. **more** scrolls the screen upward as it prints the next page. **page** clears the screen and prints a new screenful of text when it prints a new page. Both provide one line of overlap between screenfuls.

name can be a filename or **-**, specifying standard input. **more** processes file arguments in the order given.

more supports the Basic Regular Expression syntax (see *regexp*(5)).

more recognizes the following command line options:

-n *number* Set the number of lines in the display window to *number*, a positive decimal integer. The default is one line less than the the number of lines displayed by the terminal; on a screen that displays 24 lines, the default is 23. The **-n** flag overrides any values obtained from the environment.

-*n* Same as **-n** *number* except that the number of lines is set to *n*.

-c Draw each page by beginning at the top of the screen, and erase each line just before drawing on it. This avoids scrolling the screen, making it easier to read while **more** is writing. This option is ignored if the terminal has no clear-to-end-of-line capability.

-d Prompt user with the message **Press space to continue, q to quit, h for help** at the end of each screenful. This is useful if **more** is being used as a filter in some setting, such as a training class, where many users might be unsophisticated.

-e Exit immediately after writing the last line of the last file in the argument list

-f Count logical lines, rather than screen lines. That is, long lines are not folded. This option is recommended if *nroff* output is being piped through *ul*, since the latter can generate escape sequences. These escape sequences contain characters that would ordinarily occupy screen positions, but which do not print when sent to the terminal as part of an escape sequence. Thus **more** might assume lines are longer than they really are, and fold lines erroneously.

-i Perform pattern matching in searches without regard to case.

-s Squeeze multiple blank lines from the output, producing only one blank line. Especially helpful when viewing *nroff* output, this option maximizes the useful information present on the screen.

-u Normally, **more** handles underlining and bold such as produced by *nroff* in a manner appropriate to the particular terminal: if the terminal supports underlining or has a highlighting (usually inverse-video) mode, **more** outputs appropriate escape sequences to enable underlining, else highlighting mode, for underlined information in the source file. If the terminal supports highlighting, **more** uses that mode information that should be printed in boldface type. The **-u** option suppresses this processing, as do the "ul" and "os" terminfo flags.

-v Do not display nonprinting characters graphically; by default, all non-ASCII and control characters (except **<Tab>**, **<Backspace>**, and **<Return>**) are displayed visibly in the form **^X** for **<Ctrl-x>**, or **M-x** for non-ASCII character **x**.

-z Same as not specifying **-v**, with the exception of displaying **<Backspace>** as **^H**, **<Return>** as **^M**, and **<Tab>** as **^I**.

-p *command* Execute the **more** command initially in the *command* argument for each file examined. If the command is a positioning command, such as a line number or a regular expression search, sets the current position to represent the final results of the command, without writing any intermediate lines of the file. If the positioning command is unsuccessful, the first line in the file is the current position.

-t *tagstring* Write the screenful of the file containing the tag named by the *tagstring* argument. The specified tag appears in the current position. If both **-p** and **-t** options are specified, **more** processes **-t** first; that is, the file containing the *tagstring* is selected by **-t** and then the command is executed.

-x *tabs* Set the tabstops every *tabs* position. The default value for the *tabs* argument is 8.

-W *option* Provides optional extensions to the **more** command. Currently, the following two options are supported:

notite Prevents **more** from sending the terminal initialization string before displaying the file. This argument also prevents **more** from sending the terminal de-initialization string before exiting.

tite Causes **more** to send the initialization and de-initialization strings. This is the default.

+*linenumber* Start listing such that the current position is set to *linenumber*.

+/*pattern* Start listing such that the current position is set to two lines above the line matching the regular expression *pattern*.

Note: Unlike editors, this construct should NOT end with a **/**. If it does, the trailing slash is taken as character in the search pattern.

The number of lines available per screen is determined by the **-n** option, if present or by examining values in the environment. The actual number of lines written is one less than this number, as the last line of the screen is used to write a user prompt and user input.

The number of columns available per line is determined by examining values in the environment. **more** writes lines containing more characters than would fit into this number of columns by breaking the line into one more logical lines where each of these lines but the last contains the number of characters needed to fill the columns. The logical lines are written independently of each other; that is, commands affecting a single line affect them separately.

While determining the number of lines and the number of columns, if the methods described above do not yield any number then **more** uses terminfo descriptor files (see *term*(4)). If this also fails then the number of lines is set to 24 and the number of columns to 80.

When standard output is a terminal and **-u** is not specified, **more** treats backspace characters and carriage-return characters specially.

- A character, followed first by a backspace character, then by an underscore (_), causes that character to be written as underlined text, if the terminal supports that. An underscore, followed first by a backspace character, then any character, also causes that character to be written as underlined text, if the terminal supports that.
- A backspace character that appears between two identical printable characters causes the first of those two characters to be written as emboldened text, if the terminal type supports that, and the second to be discarded. Immediately subsequent occurrences of backspaces/character pairs for that same character is also discarded.
- Other backspace character sequences is written directly to the terminal, which generally causes the character preceding the backspace character to be suppressed in the display.
- A carriage-return character at the end of a line is ignored, rather than being written as a control character.

If the standard output is not a terminal device, **more** always exits when it reaches end-of-file on the last file in its argument list. Otherwise, for all files but the last, **more** prompts, with an indication that it has reached the end of file, along with the name of the next file. For the last file specified, or for the standard input if no file is specified, **more** prompts, indicating end-fo-file, and accept additional commands. If the next command specifies forward scrolling, **more** will exit. If the **-e** option is specified, **more** will exit immediately after writing the last line of the last file.

more uses the environment variable **MORE** to preset any flags desired. The **MORE** variable thus sets a string containing flags and arguments, preceded with hyphens and blank-character-separated as on the command line. Any command-line flags or arguments are processed after those in the **MORE** variable, as if the command line were as follows:

> **more $MORE** *flags arguments*

For example, to view files using the **-c** mode of operation, the shell command sequence

```
MORE='-c' ; export MORE
```

or the *csh* command

```
setenv MORE -c
```

causes all invocations of **more**, including invocations by programs such as *man* and *msgs*, to use this mode. The command sequence that sets up the **MORE** environment variable is usually placed in the *.profile* or *.cshrc* file.

In the following descriptions, the *current position* refers to two things:

- the position of the current line on the screen
- the line number (in the file) of the current line on the screen

The line on the screen corresponding to the current position is the third line on the screen. If this is not possible (there are fewer than three lines to display or this is the first page of the file, or it is the last page of the file), then the current position is either the first or last line on the screen.

Other sequences that can be typed when **more** pauses, and their effects, are as follows (*i* is an optional integer argument, defaulting to 1):

*i***<Return>** *i***j** *i***<Ctrl-e>** *i***<Space>**	Scroll forward *i* lines. The default *i* for **<Space>** is one screenful; for **j** and **<Return>** it is one line. The entire *i* lines are written, even if *i* is more than the screen size. At end-of-file, **<Return>** causes **more** to continue with the next file in the list, or exits if the current file is the last file in the list.
*i***d** *i***<Ctrl-d>**	Scroll forward *i* lines, with a default of one half of the screen size. If *i* is specified, it becomes the new default for subsequent **d** and **u** commands.
*i***u** *i***<Ctrl-u>**	Scrolls backward *i* lines, with a default of one half of the screen size. If *i* is specified, it becomes the new default for subsequent **d** and **u** commands.
*i***k** *i***<Ctrl-y>**	Scrolls backward *i* lines, with a default of one line. The entire *i* lines are written, even if *i* is more than the screen size.
*i***z**	Display *i* more lines and sets the new window (screenful) size to *i* .
*i***g**	Go to line *i* in the file, with a default of 1 (beginning of file). Scroll or rewrite the screen so that the line is at the current position. If *i* is not specified, then **more** displays the first screenful in the file.
*i***G**	Go to line *i* in the file, with a default of the end of the file. If *i* is not specified, scrolls or rewrites screen so that the last line in the file is at the bottom of the screen. If *i* is specified, scrolls or rewrites the screen so that the line is at the current position.
*i***s**	Skip forward *i* lines, with a default of 1, and write the next screenful beginning at that point. If *i* would cause the current position to be such that less than one screenful would be written, the last screenful in the file is written.

*i***f**
*i***<Ctrl-f>** Move forward *i* lines, with a default of one screenful. At end-of-file, **more** will continue with the next file in the list, or exit if the current file is the last file in the list.

*i***b**
*i***<Ctrl-b>** Move backward *i* lines, with a default of one screenful. If *i* is more than the screen size, only the final screenful will be written.

q
Q
:q
:Q
ZZ Exit from **more**.

=
:f
<Ctrl-g> Write the name of the file currently being examined, the number relative to the total number of files there are to examine, the current line number, the current byte number, and the total bytes to write and what percentage of the file precedes the current position. All of these items reference the first byte of the line after the last line written.

v Invoke an editor to edit the current file being examined. The name of the editor is taken from the environment variable **EDITOR**, or default to **vi**. If **EDITOR** represents either **vi** or **ex**, the editor is invoked with options such that the current editor line is the physical line corresponding to the current position in **more** at the time of the invocation.

When the editor exits, **more** resumes on the current file by rewriting the screen with the current line as the current position.

h Display a description of all the **more** commands.

i **/**[**!**]*expression*
Search forward in the file for the *i*-th line containing the regular expression *expression*. The default value for *i* is 1. The search starts at the line following the current position. If the search is successful, the screen is modified so that the searched-for line is in the current position. The null regular expression (**/<Return>**) repeats the search using the previous regular expression. If the character **!** is included, the lines for searching are those that do not contain *expression*.

If there are less than *i* occurrences of *expression*, and the input is a file rather than a pipe, then the position in the file remains unchanged.

The user's erase and kill characters can be used to edit the regular expression. Erasing back past the first column cancels the search command.

i **?**[**!**]*expression*
Same as **/**, but searches backward in the file for the *i* th line containing the regular expression *expression*.

Note: Unlike editors, the **?**. construct should NOT end with a **/**. If it does, the trailing slash is taken as a character in the search pattern.

*i***n** Repeat the previous search for the *i*-th line (default 1) containing the last *expression* (or not containing the last *expression*, if the previous search was **/!** or **?!**).

*i***N** Repeat the search for the opposite direction of the previous search for the *i*-th line (default 1) containing the last *expression*

'' (2 apostrophes) Return to the position from which the last large movement command was executed ("large movement" is defined as any movement of more than a screenful of lines). If no such movements have been made, return to the beginning of the file.

!*command* Invoke a shell with *command*. The characters **%** and **!** in *command* are replaced with the current file name and the previous shell command, respectively. If there is no current file name, **%** is not expanded. The sequences **\%** and **\!** are replaced by **%** and **!** respectively.

:e [*file*]
E [*file*] Examine a new file. If the *file* argument is not specified, the "current" file (see the **:n** and **:p** commands) from the list of files in the command line is re-examined. The filename is subjected to the process of shell word expansions. If *file* is a **#** (number sign) character, the previously examined file is re-examined.

i **:n** Examine the next file. If *i* is specified, examines the *i*-th next file specified in the command line.

i **:p** Examine the previous file. If a number *i* is specified, examines the *i*-th previous file specified in the command line.

:t *tagstring* Go to the supplied *tagstring* and scroll or rewrite the screen with that line in the current position.

m *letter* Mark the current position with the specified letter, where *letter* represents the name of one of the lower-case letters of the portable character set.

' *letter* Return to the position that was previously marked with the specified *letter*, making that line the current position.

r
<Ctrl-l> Refresh the screen.

R Refresh the screen, discarding any buffered input.

. Dot. Repeat the previous command.

^\ Halt a partial display of text. **more** stops sending output, and displays the usual prompt. Unfortunately, some output is lost as a result.

The commands take effect immediately; i.e., it is not necessary to press **<Return>**. Up to the time when the command character itself is given, the line-kill character can be used to cancel the numerical argument being formed.

If the standard output is not a teletype, **more** is equivalent to *cat*(1).

more supports the **SIGWINCH** signal, and redraws the screen in response to window size changes.

EXTERNAL INFLUENCES

Environment Variables

COLUMNS Overrides the system-selected horizontal screen size.

EDITOR Used by the **v** command to select an editor.

LANG Provides a default value for the internationalization variables that are unset or null. If **LANG** is unset or null, the default value of "C" (see *lang*(5)) is used. If any of the internationalization variables contains an invalid setting, **more** will behave as if all internationalization variables are set to "C". See *environ*(5).

LC_ALL If set to a non-empty string value, overrides the values of all the other internationalization variables.

LC_CTYPE Determines the interpretation of text as single and/or multi-byte characters, the classification of characters as printable, and the characters matched by character class expressions in regular expressions.

LC_MESSAGES
Determines the locale that should be used to affect the format and contents of diagnostic messages written to standard error and informative messages written to standard output.

NLSPATH Determines the location of message catalogues for the processing of **LC_MESSAGES.**

LINES Overrides the system-selected vertical screen size, used as the number of lines in a screenful. The **-n** option takes precedence over the **LINES** variable for determining the number of lines in a screenful.

MORE Determines a string containing options, preceded with hyphens and blank-character-separated as on the command line. Any command-line options are processed after those in the **MORE** variable. The **MORE** variable takes precedence over the **TERM** and **LINES** variables for determining the number of lines in a screenful.

TERM Determines the name of the terminal type.

International Code Set Support

Single- and multi-byte character code sets are supported.

APPLICATION USAGE

When the standard output is not a terminal, none of the filter-modification options is effective. This is based on historical practice. For example, a typical implementation of **man** pipes its output through **more -s** to squeeze excess white space for terminal users. When **man** is piped to **lp**, however, it is undesirable for this squeezing to happen.

EXAMPLES

To view a simple file, use:

more *filename*

To preview *nroff* output, use a command resembling:

```
nroff -mm +2 doc.n | more -s
```

If the file contains tables, use:

tbl *file* **| nroff -mm | col | more -s**

To display file **stuff** in a fifteen line-window and convert multiple adjacent blank lines into a single blank line:

more -s -n 15 *stuff*

To examine each file with its last screenful:

more -p G *file1 file2*

To examine each file starting with line 100 in the current position (third line, so line 98 is the first line written):

more -p 100g *file1 file2*

To examine the file that contains the tagstring *tag* with line 30 in the current position:

```
more -t tag -p 30g
```

WARNINGS

Standard error, file descriptor 2, is normally used for input during interactive use and should not be redirected (see Input/Output section in the manpage of the shell in use).

FILES

/usr/share/lib/terminfo/?/* compiled terminal capability data base

AUTHOR

more was developed by Mark Nudleman, University of California, Berkeley, OSF, and HP.

SEE ALSO

csh(1), man(1), pg(1), sh(1), term(4), terminfo(4), environ(5), lang(5), regexp(5).

STANDARDS CONFORMANCE

more: XPG4

NAME

mt - magnetic tape manipulating program

SYNOPSIS

mt [**-f** *tapename*] *command* [*count*]

Obsolescent

mt [**-t** *tapename*] *command* [*count*]

DESCRIPTION

mt is used to give commands to the tape drive. If *tapename* is not specified, the environment variable **TAPE** is used; if **TAPE** is not defined, the default drive is used.

mt winds the tape in the requested direction (forward or backward), stopping after the specified *count* EOF marks or records are passed. If *count* is not specified, one is assumed. Each EOF mark counts as one record. When winding backwards, the tape always stops at the BOT marker, regardless of the number remaining in *count*.

mt accepts the following *command*s:

eof	Write *count* EOF marks.
fsf	Forward space *count* files.
fsr	Forward space *count* records.
bsf	Backward space *count* files.
bsr	Backward space *count* records.
rew	Rewind tape.
offl	Rewind tape and go offline.
eod	Seek to end of data (DDS and QIC drives only).
smk	Write *count* setmarks (DDS drives only).
fss	Forward space *count* setmarks (DDS drives only).
bss	Backward space *count* setmarks (DDS drives only).

Spacing operations (back or forward space file or record) leave the tape positioned past the object being spaced to in the direction of motion. That is, backspacing a file leaves the the tape positioned before the file mark, forward spacing a file leaves the tape positioned after the file mark. This is consistent with all classical usage on tapes.

WARNINGS

Only raw, no-rewind Berkeley-type devices should be specified. This type of device will not reposition the tape upon close. An example of such a device is **/dev/rmt/0mnb**. Please refer to *mt*(7) for more details.

It is possible to wind the tape beyond the EOT marker and off the end of the reel.

EXAMPLES

Rewind the tape associated with the device file **/dev/rmt/0mnb**:

```
mt -f /dev/rmt/0mnb rew
```

FILES

/dev/rmt/*	Raw magnetic tape interface
/dev/rmt/0mnb	Default tape interface

AUTHOR

mt was developed by the University of California, Berkeley.

SEE ALSO

dd(1), mt(7).

NAME

mv - move or rename files and directories

SYNOPSIS

mv [**-f**|**-i**] [**-e** *extarg*] *file1 new-file*

mv [**-f**|**-i**] [**-e** *extarg*] *file1* [*file2* ...] *dest-directory*

mv [**-f**|**-i**] [**-e** *extarg*] *directory1* [*directory2* ...] *dest-directory*

DESCRIPTION

The **mv** command moves:

- One file (*file1*) to a new or existing file (*new-file*).
- One or more files (*file1*, [*file2*, ...]) to an existing directory (*dest-directory*).
- One or more directory subtrees (*directory1*, [*directory2*, ...]) to a new or existing directory (*dest-directory*).

Moving *file1* to *new-file* is used to rename a file within a directory or to relocate a file within a file system or across different file systems. When the destination is a directory, one or more files are moved into that directory. If two or more files are moved, the destination must be a directory. When moving a single file to a new file, if *new-file* exists, its contents are destroyed.

If the access permissions of the destination *dest-directory* or existing destination file *new-file* forbid writing, **mv** asks permission to overwrite the file. This is done by printing the mode (see *chmod*(2) and Access Control Lists below), followed by the first letters of the words *yes* and *no* in the language of the current locale, prompting for a response, and reading one line from the standard input. If the response is affirmative and the action is permissible, the operation occurs; if not, the command proceeds to the next source file, if any.

If *file1* is a file and *new-file* is a link to another file with other links, the other links remain and *new-file* becomes a new file. If *file1* is a file with links or a link to a file, the existing file or link remains intact, but the name is changed to *new-file* which may or may not be in the directory where *file1* resided, depending on directory path names used in the **mv** command. The last access and modification times of the file or files being moved remain unchanged.

Options

mv recognizes the following options:

-f Perform **mv** commands without prompting for permission. This option is assumed when the standard input is not a terminal.

-i Causes **mv** to write a prompt to standard output before moving a file that would overwrite an existing file. If the response from the standard input is affirmative, the file is moved if permissions allow the move.

-e *extarg* Specifies the handling of any extent attributes of the files(s) to be moved. *extarg* can be one of the following values:

warn	Issue a warning message if extent attributes cannot be preserved, but move the file anyway.
ignore	Do not preserve extent attributes.
force	Do not move the file if the extent attributes cannot be preserved. If multiple source files are specified with a single target directory, **mv** will move the files that either do not have extent attributes or that have extent attributes that can be preserved. **mv** will not move the files if it cannot preserve their extent attributes.

Extent attributes cannot be preserved if the files are being moved to a file system that does not support extent attributes or if that file system has a different block size than the original. If **-e** is not specified, the default value for *extarg* is **warn**.

Access Control Lists (ACLs)

If optional ACL entries are associated with *new-file*, **mv** displays a plus sign (**+**) after the access mode when asking permission to overwrite the file.

m

If *new-file* is a new file, it inherits the access control list of *file1*, altered to reflect any difference in ownership between the two files (see *acl*(5)).

EXTERNAL INFLUENCES

Environment Variables

LC_CTYPE determines the interpretation of text as single byte and/or multibyte characters.

LANG and **LC_CTYPE** determine the local language equivalent of y (for yes/no queries).

LANG determines the language in which messages are displayed.

If **LC_CTYPE** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of **C** (see *lang*(5)) is used instead of **LANG**. If any internationalization variable contains an invalid setting, **mv** behaves as if all internationalization variables are set to **C**. See *environ*(5).

International Code Set Support

Single character and multibyte character code sets are supported.

EXAMPLES

Rename a file in the current directory:

```
mv old-filename new-filename
```

Rename a directory in the current directory:

```
mv old-dirname new-dirname
```

Rename a file in the current directory whose name starts with a nonprinting control character or a character that is special to the shell, such as **-** and ***** (extra care may be required depending on the situation):

```
mv ./bad-filename new-filename
mv ./?bad-filename new-filename
mv ./*bad-filename new-filename
```

Move directory **sourcedir** and its contents to a new location (**targetdir**) in the file system (upon completion, a subdirectory named **sourcedir** resides in directory **targetdir**):

```
mv sourcedir targetdir
```

Move all files and directories (including links) in the current directory to a new location underneath **targetdir**:

```
mv * targetdir
```

Move all files and directories (including links) in **sourcedir** to a new location underneath **targetdir** (**sourcedir** and **targetdir** are in separate directory paths):

```
mv sourcedir/* targetdir
```

WARNINGS

If *file1* and *new-file* exist on different file systems, **mv** copies the file and deletes the original. In this case the mover becomes the owner and any linking relationship with other files is lost. **mv** cannot carry hard links across file systems. If *file1* is a directory, **mv** copies the entire directory structure onto the destination file system and deletes the original.

mv *cannot* be used to perform the following operations:

- Rename either the current working directory or its parent directory using the **.** or **..** notation.
- Rename a directory to a new name identical to the name of a file contained in the same parent directory.

DEPENDENCIES

NFS

Access control lists of networked files are summarized (as returned in *st_mode* by *stat*(2)), but not copied to the new file. When using **mv** on such files, a **+** is not printed after the mode value when asking for permission to overwrite a file.

AUTHOR

`mv` was developed by AT&T, the University of California, Berkeley and HP.

SEE ALSO

cp(1), cpio(1), ln(1), rm(1), link(1M), lstat(2), readlink(2), stat(2), symlink(2), symlink(4), acl(5).

STANDARDS CONFORMANCE

`mv`: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

NAME

neqn - format mathematical text for nroff

SYNOPSIS

neqn [**-d** *xy*] [**-s** *n*] [**-f** *n*] [**-p** *n*] [*file* ...]

Note

The output of **neqn** is very device-dependent. See the "WARNINGS" section.

DESCRIPTION

neqn is a preprocessor for **nroff** (see *nroff*(1)) for typesetting mathematical text on typewriter-like terminals. Its invocation is almost always of the following two forms or equivalent:

```
neqn files | nroff | col
tbl files | neqn | nroff | col
```

If no files are specified (or if **-** is specified instead of *file*), **neqn** reads from standard input. A line beginning with **.EQ** marks the start of an equation. The end of an equation is marked by a line beginning with **.EN**. Neither of these lines is altered, which means that they can be defined in macro packages to get centering, numbering, etc.

It is also possible to designate two characters as *delimiters*; subsequent text between delimiters is then treated as **neqn** input. Delimiters can be set to characters *x* and *y* with the command-line argument **-d***xy* or (more commonly) with the sequence

```
.EQ
delim xy
.EN
```

The left and right delimiters can be the same character; the dollar sign (**$**) is often used as such a delimiter. Delimiters are turned off by **delim off** (see the "WARNINGS" section). All text that is neither between delimiters nor between **.EQ** and **.EN** is passed through untouched.

n

Tokens within **neqn** equations are separated by spaces, tabs, newlines, braces, double quotes, tildes, and circumflexes. Braces (**{}**) are used for grouping; generally speaking, anywhere a single character such as *x* can appear, a complicated construction enclosed in braces can be used instead. Tilde (**~**) represents a full space in the output; circumflex, (**^**) half as much.

Subscripts and Superscripts

Subscripts and superscripts are produced using **sub** and **sup** as follows:

Source Text	Result
`x sub j`	x_j
`a sub k sup 2`	a_k^2
`e sup {x sup 2 + y sup 2}`	$e^{x^2+y^2}$

Fractions

Fractions are produced by using **over**:

Source Text	Result
`a over b`	$\frac{a}{b}$

Square Roots

sqrt produces square roots:

Source Text	Result
`1 over sqrt {ax sup 2+bx+c}`	$\frac{1}{\sqrt{ax^2+bx+c}}$

Upper and Lower Limits

The keywords **from** and **to** specify lower and upper limits:

Source Text	**Result**
`lim from {n -> inf } sum from 0 to n x sub i`	$\lim_{n \to \infty} \sum_{0}^{n} x_i$

Brackets and Braces

Left and right brackets, braces, and such, of proper height are made with **left** and **right**:

Source Text	**Result**

`left [ {x sup 2 + y sup 2}`
`    over alpha right ] ~=~ 1`
	$\left[\frac{x^2+y^2}{\alpha} \right] ~=~ 1$

Legal characters after **left** and **right** are braces, brackets, bars, **c** and **f** for ceiling and floor, and **""** for nothing at all (useful for a right-side-only bracket). A **left** *char* need not have a matching **right** *char*.

Vertical Piles

Vertical piles of *elements* are made with **pile**, **lpile**, **cpile**, and **rpile**:

Source Text	**Result**
`pile {a above bb above ccc}`	$\begin{matrix} a \\ bb \\ ccc \end{matrix}$

Piles can have arbitrary numbers of elements; **lpile** left justifies, **pile** and **cpile** center (but with different vertical spacing), and **rpile** right justifies.

Matrices and Determinants

Matrices are made with **matrix**:

Source Text	**Result**

`left | { matrix {`
`    lcol { x sub i above y sub 2 }`
`    ccol { 1 above 234 } } } right |`
	$\left\| \begin{array}{lc} x_i & 1 \\ y_2 & 234 \end{array} \right\|$

In addition, there is **rcol** for a right-justified column.

Diacritical Marks

Diacritical marks are made with **dot**, **dotdot**, **hat**, **tilde**, **bar**, **vec**, **dyad**, and **under**:

Source Text	**Result**
`x dot = f(t) bar`	$\dot{x} = \overline{f(t)}$
`y dotdot bar ~=~ n under`	$\bar{\ddot{y}} ~=~ \underline{n}$
`x vec ~=~ y dyad`	$\vec{x} ~=~ \overleftrightarrow{y}$

Point Sizes and Fonts

Point sizes and fonts can be changed with **size** *n* or **size +**\|**-***n*, **roman**, **italic**, **bold**, and **font** *n*. Point sizes and fonts can be changed globally in a document by **gsize** *n* and **gfont** *n*, or by the command-line arguments **-s***n* and **-f***n*.

Normally, subscripts and superscripts are reduced by 3 points from the previous size; this can be changed by the command-line argument **-p***n*.

Vertical Alignment

Successive display arguments can be lined up. Place **mark** before the desired lineup point in the first equation; place **lineup** at the place that is to line up vertically in subsequent equations.

Shorthand Forms

Shorthand forms can be defined or existing keywords redefined with **define**:

define *thing % replacement %*

defines a new token called *thing* that is replaced by *replacement* whenever it appears thereafter. The % can be any character that does not occur in *replacement*.

Other Keywords

Keywords such as **sum** ($\sum$), **int** ($\int$), **inf** (∞), and shorthands such as **>=** ($\geq$), **!=** ($\neq$), and **->** ($\rightarrow$) are recognized. Greek letters are spelled out in uppercase or lowercase as desired, as in **alpha** (α) or **GAMMA** (Γ). Mathematical words such as **sin**, **cos**, and **log** are made Roman automatically. **nroff** four-character escapes such as **\(dd** (‡) and **\(bu** (•) can be used anywhere.

Verbatim Text

Strings enclosed in double quotes (**"***string***"**) are passed through untouched; this permits keywords to be entered as text, and can be used to communicate with **nroff** when other methods fail. Details are given in the manuals cited below.

EXTERNAL INFLUENCES

Environment Variables

LC_CTYPE determines the interpretation of text as single- or multibyte characters.

LANG determines the language in which messages are displayed.

If **LC_CTYPE** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**. If any internationalization variable contains an invalid setting, **neqn** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multibyte character code sets are supported.

WARNINGS

To embolden digits, parentheses, etc., it is necessary to quote them, as in **bold "12.3"**. Also see the "WARNINGS" section in *nroff*(1).

Good practice dictates that if a delimiter is specified in a file, the **delim off** directive should be included at the end of the file to prevent undesirable behavior when processing multiple files where a subsequent file may contain the delimiter character as part of regular text.

To properly display equations on terminal screens and other devices that do not support reverse line feeds, **nroff** output should be piped through **col** (see *col*(1)).

The display on devices that do not support partial linefeeds is often difficult to understand; Greek characters and other symbols are often not well supported and can mismatched printing of bold words on the same line (see a printed version of the "Other Keywords" subsection above). Consider using "computer-program" coding instead.

SEE ALSO

col(1), mm(1), nroff(1), tbl(1), mm(5).

Typesetting Mathematics – User's Guide, by B.W. Kernighan and L.L. Cherry.

New Graphic Symbols for EQN and NEQN, by C. Scrocca.

NAME

netstat - show network status

SYNOPSIS

netstat [**-an**] [**-f** *address-family*] [*system* [*core*]]

netstat [**-Mnrsv**] [**-f** *address-family*] [**-p** *protocol*] [*system* [*core*]]

netstat [**-gin**] [**-I** *interface*] [*interval*] [*system* [*core*]]

DESCRIPTION

netstat displays statistics for network interfaces and protocols, as well as the contents of various network-related data structures. The output format varies according to the options selected. Some options are ignored when used in combination with other options.

Generally, the **netstat** command takes one of the three forms shown above:

- The first form of the command displays a list of active sockets for each protocol.
- The second form displays the contents of one of the other network data structures according to the option selected.
- The third form displays configuration information for each network interface. It also displays network traffic data on configured network interfaces, optionally updated at each *interval*, measured in seconds.

Options are interpreted as follows:

-a Show the state of all sockets, including passive sockets used by server processes. When **netstat** is used without any options only active sockets are shown. This option does not show the state of X.25 programmatic access sockets. The option is ignored if the **-g**, **-i**, **-I**, **-M**, **-p**, **-r**, **-s** or *interval* option is specified.

-f *address-family*
Show statistics or address control block for only the specified *address-family*. The following address families are recognized: **inet** for **AF_INET**, and **unix** for **AF_UNIX**. This option applies to the **-a** and **-s** options.

-g Show multicast information for network interfaces. Only the address family **AF_INET** is recognized by this option. This option may be combined with the **-i** option to display both kinds of information. The option is ignored if the **-p** option is specified.

-i Show the state of network interfaces. Only the interfaces that have been configured with an IP address or the **plumb** option using the **ifconfig** command are shown. The output includes both the primary and logical interfaces. (See *ifconfig*(1M)). The counts for Ipkts and Opkts fields are for IP packets only. This option is ignored if the **-p** option is specified.

-I *interface* Show information about the specified interface only. This option applies to the **-g** and **-i** options.

-M Show the multicast routing tables. When **-s** is used with the **-M** option, **netstat** displays multicast routing statistics instead. This option is ignored if the **-p** option is specified.

-n Show network addresses as numbers. Normally, **netstat** interprets addresses and attempts to display them symbolically. This option applies to the **-a**, **-i**, **-r** and **-v** options.

-p *protocol* Show statistics for the specified protocol. The following protocols are recognized: **tcp**, **udp**, **ip**, **icmp**, and **igmp**.

-r Show the routing tables. When **-v** is used with the **-r** option, **netstat** also displays the network masks in the route entries. When **-s** is used with the **-r** option, **netstat** displays routing statistics instead. This option is ignored if the **-g**, **-i**, **-I**, **-p** or *interval* option is specified.

-s Show statistics for all protocols. When this option is used with the **-M** option, **netstat** displays multicast routing statistics instead. This option is ignored if the **-g**, **-i**, **-I**, **-p** or *interval* option is specified.

n

-v Show additional routing information. When **-v** is used with the **-r** option, **netstat** also displays the network masks in the route entries. This option only applies to the **-r** option.

The arguments *system* and *core* allow substitutes for the defaults, **/stand/vmunix** and **/dev/kmem**.

If no options is specified, **netstat** displays the status of only active sockets. The display of active and passive sockets status shows the local and remote addresses, send and receive queue sizes (in bytes), protocol, and the internal state of the protocol. Address formats are of the form *host*.*port*, or *network*.*port* if the host portion of a socket address is zero. When known, the host and network addresses are displayed symbolically by using **gethostbyname()** and **getnetbyname()**, respectively (see *gethostent*(3N) and *getnetent*(3N)). If a symbolic name for an address is unknown, the address is displayed numerically according to the address family. For more information regarding the Internet "dot format", refer to *inet*(3N). Unspecified or "wildcard" addresses and ports appear as an asterisk (*****).

The interface display provides a table of cumulative statistics regarding packets transferred, both inbound and outbound. The network addresses of the interface and the maximum transmission unit (MTU) are also displayed. When the *interval* argument is specified, **netstat** displays a running count of statistics related to network interfaces. This display consists of a column for the first interface found during autoconfiguration and a column summarizing information for all interfaces. To display a running count of statistics for a specific interface, use the **-I** option. The first line of each screen of information contains a summary since the system was last rebooted. Subsequent lines of output show values accumulated over the preceding interval.

The routing table display indicates the available routes and their status. Each route consists of a destination host or network, a netmask and a gateway to use in forwarding packets. The **Flags** field shows whether the route is up (**U**), whether the route is to a gateway (**G**), or whether the route is a host or network route (with or without **H**).

The **Netmask** field shows the mask to be applied to the destination IP address of an IP packet to be forwarded. The result will be compared with the destination address in the route entry. If they are the same, then the route is one of the candidates for routing this IP packet. If there are several candidate routes, then the route with the longest **Netmask** field (contiguous 1's starting from the leftmost bit position) will be chosen. (see *routing*(7).)

n

The **Gateway** field shows the address of the immediate gateway for reaching the destination. It can be the address of the outgoing interface if the destination is on a directly connected network.

The **Use** field shows a count of the number of packets sent using this route.

The **Interface** field identifies which network interface is used for the route.

The **Pmtu** field displays the path maximum transmission unit (PMTU). If the route is created with a static PMTU value (see *route*(1M)), the corresponding PMTU value permanently overrides the interface MTU. Otherwise, the **Pmtu** value is the same as the MTU of the network interface used for the route.

DEPENDENCIES

X.25:

-a option does not list X.25 programmatic access information.

AUTHOR

netstat was developed by HP and the University of California, Berkeley.

SEE ALSO

hosts(4), networks(4), gethostent(3N), getnetent(3N), protocols(4), route(1M), ifconfig(1M), lanscan(1M), lanadmin(1M), services(4).

NAME

newalias - install new elm aliases for user or system

SYNOPSIS

newalias [**-g**]

Remarks

newalias replaces the former functionality of the **elmalias** command.

DESCRIPTION

The **newalias** command creates new alias database files from an alias text file for use by **elm** and other programs. For user aliases, this functionality can also be performed from the Alias Menu of the **elm** program (see *elm*(1)).

Options

newalias recognizes the following option:

-g Global. The program updates the system alias files instead of a user's alias files.

Operation

Without the **-g** option, **newalias** updates a user's alias files, based on an input file named

```
$HOME/.elm/aliases.text
```

Upon finding the file, it creates the output files named

```
$HOME/.elm/aliases
$HOME/.elm/aliases.dir
$HOME/.elm/aliases.pag
```

With the **-g** option, **newalias** updates the system alias files, based on an input file named

```
/var/mail/.elm/aliases.text
```

Upon finding the file, it creates the output files named

```
/var/mail/.elm/aliases
/var/mail/.elm/aliases.dir
/var/mail/.elm/aliases.pag
```

In either case, you need read access to the **aliases.text** file and write access to the other files and the **.elm** directory.

Text File Entries

Each entry in either **aliases.text** file is expected to be in the following format:

alias-list **=** [*lastname* [**;** *firstname*]] [**,** *comment*] **=** *address-list*

Field Names

The field names are defined as follows:

address-list	A blank- or comma-separated list of one or more mail addresses, personal alias names, and/or group alias names. In practice, each item is tested first as an alias name. If it is not an alias name, it is assumed to be a mail address. A mail address can be in Internet form (**user@host.domain**), in UUCP form (**host.domain!user**), or in **sendmail** alias form (see *sendmail*(1M)). It can also be the name of a local mail user, which is appended with the local host name in Internet form.
alias-list	A blank- or comma-separated list of alias names. Each name identifies the same alias entry. An alias name can be made up of letters (**A**–**Z**, **a**–**z**), digits (**0**–**9**), underscores (_), dashes (**-**), and periods (**.**). Alias names are not case-sensitive, so **dave** and **Dave** are equivalent.
comment	A string containing any information you wish about the entry, such as location and phone numbers. It is displayed in the Alias Menu of the **elm** program, but **elm** does not transmit it in a mail message. This field can contain any characters except an unquoted equal sign (**=**). See the Quoting Characters subsection.

n

firstname	The first name of the person (or group). It is combined with *lastname* to form the *fullname*. This field can contain any characters except an unquoted equal sign (**=**) or an unquoted comma (**,**). See the Quoting Characters subsection. The only first name under the Personal or Group Aliases subheading below is: **John** in **Smith; John**.
lastname	The last name of the person (or group). It is combined with *firstname* to form the *fullname*. This field can contain any characters except an unquoted equal sign (**=**), an unquoted semicolon (**;**), or an unquoted comma (**,**). See the Quoting Characters subsection. The last names under the Personal or Group Aliases subheading below are: **Dave Taylor**, **Smith**, **Unix Gurus**, and **Unix people**.
fullname	The combination of *firstname lastname*. It is usually sent in a mail header in parentheses after the address. It is also displayed in the Alias Menu of the **elm** program and by the **elmalias** command (see *elm*(1) and *elmalias*(1)).

Delimiters

The delimiters have the following precedence:

- The first and second equal signs (**=**) mark the end of the *alias-list* and the beginning of the *address-list*, respectively. Both equal signs are required.
- The first comma (**,**) after the first equal sign and before the second equal sign marks the beginning of the *comment* field.
- The first semicolon (**;**) after the first equal sign and before the next comma or second equal sign marks the beginning of the *firstname* field.

Personal or Group Aliases

A personal or individual alias has only one address in *address-list*, as in:

```
dave, taylor = Dave Taylor = taylor@company.com

j_smith = Smith; John, 408-555-1212 =johns@pocahontas.gov
```

A group alias has two or more addresses in *address-list*, as in:

```
gurus = Unix Gurus = alan, john, dave, mike,
    richard, larry, t_richardson

unix = Unix people = gurus, taylor, jonboy
```

Other Rules

Entries can be continued over several lines; the continuation lines must start with a blank (a space or tab).

A comment is any line starting with a number sign (**#**). It is ignored.

Blank lines and comments can be interspersed within entries.

Quoting Characters

You can include normally excluded characters in *firstname*, *lastname*, *comment*, and mail addresses in *address-list* by escaping each character with a backslash (\) or by enclosing the string in quotation marks (**"**). To include a quotation mark or a backslash, escape it with a backslash, whether inside or outside quotation marks.

FILES

$HOME/.elm/aliases	User alias database data table
$HOME/.elm/aliases.dir	User alias database directory table
$HOME/.elm/aliases.pag	User alias database hash table
$HOME/.elm/aliases.text	User alias source text
/var/mail/.elm/aliases	System alias database data table
/var/mail/.elm/aliases.dir	System alias database directory table
/var/mail/.elm/aliases.pag	System alias database hash table
/var/mail/.elm/aliases.text	System alias source text

AUTHOR

newalias was developed by HP.

SEE ALSO

elm(1), elmalias(1), mail(1), mailx(1).

NAME

newform - change or reformat a text file

SYNOPSIS

newform [**-i** *tabspec*] [**-o** *tabspec*] [**-l** *n*] [**-b** *n*] [**-e** *n*] [**-c** *char*] [**-p** *n*] [**-a** *n*] [**-f**] [**-s**] [*files*]

DESCRIPTION

newform reads lines from the named *files*, or the standard input if no input file is named, and reproduces the lines on the standard output. Lines are reformatted in accordance with command line options in effect.

Except for **-s**, command line options can appear in any order, can be repeated, and can be intermingled with the optional *files*. Command line options are processed in the order specified. This means that option sequences such as **-e 15 -l 60** yield results different from **-l 60 -e 15**. Options are applied to all *files* on the command line.

Options

newform recognizes the following options:

-i*tabspec* Input tab specification: expands tabs to spaces, according to the tab specifications given. *Tabspec* recognizes all tab specification forms described in *tabs*(1). In addition, *tabspec* can be **--**, in which **newform** assumes that the tab specification is to be found in the first line read from the standard input (see *fspec*(4)). If no *tabspec* is given, *tabspec* defaults to **-8**. A *tabspec* of **-0** expects no tabs; if any are found, they are treated as **-1**.

-o*tabspec* Output tab specification: replaces spaces with tabs, according to the tab specifications given. The tab specifications are the same as for **-i***tabspec*. If no *tabspec* is given, *tabspec* defaults to **-8**. A *tabspec* of **-0** means that no spaces will be converted to tabs on output.

-l*n* Set the effective line length to *n* characters. If *n* is not entered, **-l** defaults to 72. The default line length without the **-l** option is 80 characters. Note that tabs and backspaces are treated as single characters (use **-i** to expand tabs to spaces).

n

-b*n* Truncate *n* characters from the beginning of the line when the line length is greater than the effective line length (see **-l***n*). Default is to truncate the number of characters necessary to obtain the effective line length. The default value is used when **-b** with no *n* is used. This option can be used to delete the sequence numbers from a COBOL program as follows:

newform -l1 -b7 *file-name*

The **-l1** must be used to set the effective line length shorter than any existing line in the file so that the **-b** option is activated.

-e*n* Same as **-b***n* except that characters are truncated from the end of the line.

-c*k* Change the prefix/append character to *k*. Default character for *k* is a space.

-p*n* Prefix *n* characters (see **-c***k*) to the beginning of a line when the line length is less than the effective line length. Default is to prefix the number of characters necessary to obtain the effective line length.

-a*n* Same as **-p***n* except characters are appended to the end of a line.

-f Write the tab specification format line on the standard output before any other lines are output. The tab specification format line which is printed will correspond to the format specified in the *last* **-o** option. If no **-o** option is specified, the line which is printed contains the default specification of **-8**.

-s Shears off leading characters on each line up to the first tab and places up to 8 of the sheared characters at the end of the line. If more than 8 characters (not counting the first tab) are sheared, the eighth character is replaced by a ***** and any characters to the right of it are discarded. The first tab is always discarded.

An error message and program exit occur if this option is used on a file without a tab on each line. The characters sheared off are saved internally until all other options specified are applied to that line. The characters are then added at the end of the processed line.

For example, to convert a file with leading digits, one or more tabs, and text on each line, to a file beginning with the text, all tabs after the first expanded to spaces, padded with spaces out to column 72 (or truncated to column 72), and the leading digits placed starting at column 73, the command would be:

`newform -s -i -l -a -e` *file-name*

RETURN VALUE

`newform` returns one of the following values upon completion:

0 No errors encountered.

1 An error occurred.

DIAGNOSTICS

All diagnostics are fatal.

`usage:` ...
: **`newform`** was called with a bad option.

`not -s format`
: There was no tab on one line.

`can't open file`
: Self-explanatory.

`internal line too long`
: A line exceeds 512 characters after being expanded in the internal work buffer.

`tabspec in error`
: A tab specification is incorrectly formatted, or specified tab stops are not ascending.

`tabspec indirection illegal`
: A *tabspec* read from a file (or standard input) must not contain a *tabspec* referencing another file (or standard input).

WARNINGS

`newform` normally only keeps track of physical characters; however, for the **`-i`** and **`-o`** options, **`newform`** keeps track of backspaces in order to line up tabs in the appropriate logical columns.

`newform` does not prompt the user if a *tabspec* is to be read from the standard input (by use of **`-i--`** or **`-o--`**).

If the **`-f`** option is used, and the last **`-o`** option specified was **`-o--`**, and was preceded by either a **`-o--`** or a **`-i--`**, the tab specification format line will be incorrect.

SEE ALSO

fspec(4), csplit(1), tabs(1).

n

NAME

newgrp - switch to a new group

SYNOPSIS

newgrp [**-**] [*group*]

DESCRIPTION

The **newgrp** command changes your group ID without changing your user ID and replaces your current shell with a new one.

If you specify *group*, the change is successful if *group* exists and either your user ID is a member of the new *group*, or *group* has a password and you can supply it from the terminal.

If you omit *group*, **newgroup** changes to the group specified in your entry in the password file, **/etc/passwd**.

Whether the group is changed successfully or not, or the new group is the same as the old one or not, **newgrp** proceeds to replace your current shell with the one specified in the shell field of your password file entry. If that field is empty, **newgrp** uses the POSIX shell, **/usr/bin/sh** (see *sh-posix*(1)).

If you specify **-** (hyphen) as the first argument, the new shell starts up as if you had just logged in. If you omit **-**, the new shell starts up as if you had invoked it as a subshell.

You remain logged in and the current directory is unchanged, but calculations of access permissions to files are performed with respect to the new real and effective group IDs.

Exported variables retain their values and are passed to the new shell. All unexported variables are deleted, but the new shell may reset them to default values.

Since the current process is replaced when the new shell is started, exiting from the new shell has the same effect as exiting from the shell in which **newgrp** was executed.

EXTERNAL INFLUENCES

International Code Set Support

Characters from the 7-bit USASCII code set are supported in group names (see *ascii*(5)).

DIAGNOSTICS

The **newgrp** command issues the following error messages:

Sorry	Your user ID does not qualify as a group member.
Unknown group	The group name does not exist in **/etc/group**.
Permission denied	If a password is required, it must come from a terminal.
You have no shell	Standard input is not a terminal file, causing the new shell to fail.

EXAMPLES

To change from your current group to group **users** without executing the login routines:

```
newgrp users
```

To change from your current group to group **users** and execute the login routines:

```
newgrp - users
```

WARNINGS

There is no convenient way to enter a password into **/etc/group**.

The use of group passwords is not recommended because, by their very nature, they encourage poor security practices. Group passwords may be eliminated in future HP-UX releases.

FILES

/etc/group	System group file
/etc/passwd	System password file

SEE ALSO

csh(1), ksh(1), login(1), sh-bourne(1), sh-posix(1), group(4), passwd(4), environ(5).

STANDARDS CONFORMANCE

newgrp: SVID2, SVID3, XPG2, XPG3, XPG4

NAME

newmail - notify users of new mail in mailboxes

SYNOPSIS

newmail [**-i** *interval*] [**-w**] [*file-spec*]...

DESCRIPTION

The **newmail** utility monitors your incoming mailbox or specified mail folders.

The basic operation is that the program checks the folders each *interval* seconds (default 60) and lists any new mail that has arrived in any of the mailboxes, indicating the sender's name, and the subject of the message.

Without any options, **newmail** runs in the background at a default interval of 60 seconds to monitor the user's incoming mailbox. So that they are suitable for display on an already active screen, messages are prefixed with a pair of pointer characters as follows:

> **>> Mail from** *sender-name* **-** *subject-of-message*
> **>> Priority** *sender-name* **-** *subject-of-message*

If there is no subject, the message **(No Subject Specified)** is displayed. If there is more than one folder, output lines are prefixed by the *folder-name* or the prefix string specified by *file-spec*.

newmail runs until you log out or explicitly kill it. It can internally reset itself if the mailbox shrinks in size and then grows again.

Options

newmail recognizes the following options:

-i *interval* Set the time interval between mailbox checks to the value specified, in seconds. The default is 60.

interval must be less than 2^{32} seconds. If it is set to less than 10 seconds, **newmail** warns that such short intervals are not recommended.

-w Run the program within the current window in the foreground with a more succinct output format.

The output formats become:

Mail from *sender-name* **-** *subject-of-message*
Priority *sender-name* **-** *subject-of-message*

Operands

newmail recognizes the following operand:

file-spec Specifies the name of a folder and an optional prefix string, in the form:

foldername[**=***prefix-string*]

Metacharacters such as **+**, **=**, and **%** indicate the folder directory. The default is the value of the environment variable **MAILDIR** or **$HOME/Mail**.

EXAMPLES

Check incoming mailbox every 60 seconds:

```
newmail
```

Check incoming mailboxes of **joe** and **root** every 15 seconds for new messages.

```
newmail -i 15 joe root
```

Monitor the incoming mailbox of user **mary** and the folder in your mail directory called **postmaster**. Prefix all new messages in the incoming mailbox of **mary** with the string **Mary**, and the new messages in the folder **postmaster** with **POBOX**. Also, monitor folder **/tmp/mbox**:

```
newmail "mary=Mary" +postmaster=POBOX /tmp/mbox
```

AUTHOR

newmail was developed by HP.

n

NAME

news - print news items

SYNOPSIS

news [**-a**] [**-n**] [**-s**] [*items*]

DESCRIPTION

news is used to keep the user informed of current events. By convention, these events are described by files in the directory **/var/news**.

When invoked without arguments, **news** prints the contents of all current files in **/var/news**, most recent first, with each preceded by an appropriate header. **news** stores the "currency" time as the modification date of a file named **.news_time** in the user's home directory (the identity of this directory is determined by the environment variable **$HOME**); only files more recent than this currency time are considered "current."

Options

news recognizes the following options:

-a Print all items, regardless of currency. The stored time is not changed.

-n Report the names of the current items without printing their contents, and without changing the stored time.

-s Report how many current items exist without printing their names or contents, and without changing the stored time. It is useful to include such an invocation of *news* in one's **.profile** file, or in the system's **/etc/profile**.

All other arguments are assumed to be specific news items that are to be printed.

If an interrupt is typed during the printing of a news item, printing stops and the next item is started. Another interrupt within one second of the first causes the program to terminate.

EXTERNAL INFLUENCES

International Code Set Support

Single- and multi-byte character code sets are supported.

FILES

/var/news/*
$HOME/.news_time
/etc/profile

SEE ALSO

mail(1), profile(4), environ(5).

STANDARDS CONFORMANCE

news: SVID2, SVID3, XPG2

NAME

nice - run a command at nondefault priority

SYNOPSIS

nice [**-***priority_change*] *command* [*command_args*]

nice [**-n** *priority_change*] *command* [*command_args*]

DESCRIPTION

The **nice** command executes *command* at a nondefault CPU scheduling priority. (The name is derived from being "nice" to other system users by running large programs at lower priority.)

Arguments

The command-line arguments are as follows:

-n *priority_change*

priority_change	The difference between the system nice value (relative priority) of the current (or parent) process and the actual system nice value at which *command* is to run.
	An unsigned value increases the system nice value for *command*, causing it to run at lower priority.
	A negative value requires superuser privileges, and assigns a lower system nice value (higher priority) to *command*. If the current process is not privileged, the value is silently treated as if it were 0.
	If the value of *priority_change* would result in a system nice value outside the range 0 through 39, the corresponding limit value of 0 or 39 is used instead.
	Note that a positive *priority_change* (lower priority) has a single **-** option character before the numeric value; a negative (higher priority) *priority_change* has two: the option character followed by the minus sign (**--**). If **-***priority_change* is not specified, it defaults to **10**.
command	A program, HP-UX command, user shell script, etc. to be executed at the nondefault priority. *command* can be run as a foreground or background process.
	If *command* is run as a background process, any nice *priority_change* made by the shell (**ksh** executes all background processes via **nice -4**) is in addition to that specified in the **nice** command line.
command_args	Any arguments recognized by *command*.

Process Priorities

All processes have an associated system nice value which is used to compute the instantaneous-priority of the process when it is scheduled to run. Normally, all processes inherit the system nice value of their parent process when they are spawned. The shell (**sh**, **csh**, **ksh**, etc.) can create a child process with a different priority from the current shell process by spawning the child process via the **nice** command. If the *priority_change* value is unsigned (positive), the child process is nicer (lower in priority) relative to the parent. If the *priority_change* value is negative, the child process runs at a higher priority with a greater share of available system resources. To spawn a higher priority child process, the parent process must be owned by a user who has the appropriate privileges.

At boot-up, the system starts the **init** process at a system nice value of 20 (system default). On most systems, all processes (down to the login shells) inherit this priority. Starting from their individual login shell processes, users can alter the system nice value of descendent processes to as much as 39, or, with appropriate privileges, as little as 0. A system nice value of 0 establishes an extremely high priority, whereas a value of 39 indicates a very low priority.

Ordinary users can only increase the system nice value of any child process relative to the current process; i.e., *priority_change* must be a positive (unsigned) value, resulting in a lower priority. To start a child process at a lower system nice value (higher priority) than the current process, the user must have the appropriate privileges, regardless of the relative nice-priority value desired.

For example, using the command

```
nice ksh
```

from a login shell whose current nice value is 20 spawns a subshell with a system nice value of 30. Attempting to use

```
nice --2 ksh
```

from the new shell to spawn another subshell whose system nice value would be 28, is rejected (unless the user has appropriate privileges), even though the resulting system nice value would be less than the priority of the original login shell process.

The system nice value for current processes is listed under the **NI** column produced by the **ps -l** command (see *ps*(1)).

Background Processes

Foreground processes are run at same system nice value as the parent shell. Background processes spawned by **ksh** run at the equivalent of a **nice -4** by default. If a background process is started via **nice** from **ksh**, any *priority_change* specified in the **nice** command is added to default **nice -4**. Thus the command

nice 12 *command* **&**

runs at a system nice value of 36 if executed from **ksh**.

EXTERNAL INFLUENCES

Environment Variables

LC_MESSAGES determines the language in which messages are displayed.

If **LC_MESSAGES** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**.

If any internationalization variable contains an invalid setting, **nice** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported.

RETURN VALUE

nice returns the value returned by *command*.

EXAMPLES

The following examples assume the current process is running with a system nice value of 20 and **nice** is executed from the Korn shell (see *ksh*(1)).

Run a program named **prog** in the current directory at the default *priority_change* of 10 (system nice value of 30):

nice ./prog *prog_args*

Run the same program in the background using a system nice value of 36 (*priority_change*=12 plus 4 for the Korn shell):

nice -12 ./prog *prog_args* **&**

As a user with appropriate privileges, run **prog** as a foreground process with a system nice value of 6:

nice --14 ./prog *prog_args*

WARNINGS

The C shell, **csh**, has a built-in **nice** command with different syntax. See *csh*(1) for details.

SEE ALSO

csh(1), ksh(1), nohup(1), renice(1), nice(2).

STANDARDS CONFORMANCE

nice: SVID2, SVID3, XPG4

NAME

nis+, NIS+, nis - a new version of the network information name service

DESCRIPTION

NIS+ is a new version of the network information name service. This version differs in several significant ways from version 2, which is referred to as NIS or YP in earlier releases. Specific areas of enhancement include the ability to scale to larger networks, security, and the administration of the service.

The man pages for NIS+ are broken up into three basic categories. Those in section 1 are the user commands that are most often executed from a shell script or directly from the command line. Section 1M man pages describe utility commands that can be used by the network administrator to administer the service itself. The NIS+ programming API is described by man pages in section 3N.

All commands and functions that use NIS version 2 are prefixed by the letters **yp** as in *ypmatch*(1), *ypcat*(1), *yp_match*(3N), and *yp_first*(3N). Commands and functions that use the new replacement software NIS+ are prefixed by the letters **nis** as in *nismatch*(1), *nischown*(1), *nis_list*(3N), and *nis_add_entry*(3N). A complete list of NIS+ commands is in the *LIST OF COMMANDS* section.

This man page introduces the NIS+ terminology. It also describes the NIS+ namespace, authentication, and authorization policies.

NIS+ NAMESPACE

The naming model of NIS+ is based upon a tree structure. Each node in the tree corresponds to an NIS+ object. There are six types of NIS+ objects: *directory*, *table*, *group*, *link*, *entry*, and *private*.

NIS+ Directory Object

Each NIS+ namespace will have at least one NIS+ directory object. An NIS+ directory is like a UNIX file system directory which contains other NIS+ objects including NIS+ directories. The NIS+ directory that forms the root of the NIS+ namespace is called the root directory. There are two special NIS+ directories: **org_dir** and **groups_dir**. The **org_dir** directory consists of all the system-wide administration tables, such as **passwd**, **hosts**, and **mail_aliases**. The **groups_dir** directory consists of NIS+ group objects which are used for access control. The collection of **org_dir**, **groups_dir** and their parent directory is referred to as an NIS+ domain. NIS+ directories can be arranged in a tree-like structure so that the NIS+ namespace can match the organizational or administrative hierarchy.

NIS+ Table Object

NIS+ tables (not files), contained within NIS+ directories, store the actual information about some particular type. For example, the **hosts** system table stores information about the IP address of the hosts in that domain. NIS+ tables are multicolumn and the tables can be searched through any of the searchable columns. Each table object defines the schema for its table. The NIS+ tables consist of NIS+ entry objects. For each entry in the NIS+ table, there is an NIS+ entry object. NIS+ entry objects conform to the schema defined by the NIS+ table object.

NIS+ Group Object

NIS+ group objects are used for access control at group granularity. NIS+ group objects, contained within the **groups_dir** directory of a domain, contain a list of all the NIS+ principals within a certain NIS+ group. An NIS+ principal is a user or a machine making NIS+ requests.

NIS+ Link Object

NIS+ link objects are like UNIX symbolic file-system links—they are typically used for shortcuts in the NIS+ namespace.

Refer to *nis_objects*(3N) for more information about the NIS+ objects.

NIS+ NAMES

The NIS+ service defines two forms of names, **simple** names and **indexed** names. Simple names are used by the service to identify NIS+ objects contained within the NIS+ namespace. Indexed names are used to identify NIS+ entries contained within NIS+ tables. Furthermore, entries within NIS+ tables are returned to the caller as NIS+ objects of type *entry*. NIS+ objects are implemented as a union structure which is described in the file **<rpcsvc/nis_object.h>**. The differences between the various types and the meanings of the components of these objects are described in *nis_objects*(3N).

Simple Names

Simple names consist of a series of labels that are separated by the ‘.’(dot) character. Each label is composed of printable characters from the ISO Latin 1 set. Each label can be of any nonzero length, provided that the fully qualified name is fewer than **`NIS_MAXNAMELEN`** octets including the separating dots. (See **`<rpcsvc/nis.h>`** for the actual value of **`NIS_MAXNAMELEN`** in the current release.) Labels that contain special characters (see *Grammar*) must be quoted.

The NIS+ namespace is organized as a singly rooted tree. Simple names identify nodes within this tree. These names are constructed such that the leftmost label in a name identifies the leaf node and all of the labels to the right of the leaf identify that object’s parent node. The parent node is referred to as the leaf’s *directory*. This is a naming directory and should not be confused with a file system directory.

For example, the name *example.simple.name.* is a simple name with three labels, where *example* is the leaf node in this name, the directory of this leaf is *simple.name.* which by itself is a simple name. The leaf of which is *simple* and its directory is simply *name*.

The function *nis_leaf_of*(3N) returns the first label of a simple name. The function *nis_domain_of*(3N) returns the name of the directory that contains the leaf. Iterative use of these two functions can break a simple name into each of its label components.

The name ‘.’ (dot) is reserved to name the *global root* of the namespace. For systems that are connected to the Internet, this global root will be served by a Domain Name Service. When an NIS+ server is serving a root directory whose name is not ‘.’(dot), this directory is referred to as a *local root*.

NIS+ names are said to be *fully qualified* when the name includes all of the labels identifying all of the directories, up to the global root. Names without the trailing dot are called *partially* qualified.

Indexed Names

Indexed names are compound names that are composed of a search criterion and a simple name. The search criterion component is used to select entries from a table; the simple name component is used to identify the NIS+ table that is to be searched. The search criterion is a series of column names and their desired values enclosed in bracket ‘**[ ]**’ characters. These criteria take the following form:

[*column_name***=***value***,** *column_name***=***value***, ...]**

A search criterion is combined with a simple name to form an indexed name by concatenating the two parts, separated by a ‘,’(comma) character as follows.

[*search-criterion* **],***table.directory.*

When multiple column name/value pairs are present in the search criterion, only those entries in the table that have the appropriate value in all columns specified are returned. When no column name/value pairs are specified in the search criterion, **[ ]**, *all* entries in the table are returned.

Grammar

The following text represents a context-free grammar that defines the set of legal NIS+ names. The terminals in this grammar are the characters ‘.’ (dot), ‘[’ (open bracket), ‘]’ (close bracket), ‘,’ (comma), ‘=’ (equals) and whitespace. Angle brackets (‘<’ and ‘>’), which delineate non-terminals, are not part of the grammar. The character ‘|’ (vertical bar) is used to separate alternate productions and should be read as “this production *OR* this production”.

name	::=	. | <*simple name*> | <*indexed name*>
simple name	::=	<*string*>. | <*string*>.<*simple name*>
indexed name	::=	<*search criterion*>,<*simple name*>
search criterion	::=	[<*attribute list*>]
attribute list	::=	<*attribute*> | <*attribute*>,<*attribute list*>
attribute	::=	<*string*> = <*string*>
string	::=	ISO Latin 1 character set except the character ’/’ (slash). The initial character may not be a terminal character or the characters ’@’ (at), ’+’ (plus), or (‘-’) hyphen.

Terminals that appear in strings must be quoted with ‘"’ (double quote). The ‘"’ character may be quoted by quoting it with itself ‘""’.

Name Expansion

The NIS+ service only accepts fully qualified names. However, since such names may be unwieldy, the NIS+ commands in section 1 employ a set of standard expansion rules that will attempt to fully qualify a partially qualified name. This expansion is actually done by the NIS+ library function *nis_getnames*(3N) which generates a list of names using the default NIS+ directory search path or the **NIS_PATH** environment variable. The default NIS+ directory search path includes all the names in its path. **nis_getnames()** is invoked by the functions *nis_lookup*(3N) and *nis_list*(3N) when the **EXPAND_NAME** flag is used.

The **NIS_PATH** environment variable contains an ordered list of simple names. The names are separated by the ':' (colon) character. If any name in the list contains colons, the colon should be quoted as described in the *Grammar* section. When the list is exhausted, the resolution function returns the error **NIS_NOTFOUND**. This may mask the fact that the name existed but a server for it was unreachable. If the name presented to the list or lookup interface is fully qualified, the **EXPAND_NAME** flag is ignored.

In the list of names from the **NIS_PATH** environment variable, the '$' (dollar sign) character is treated specially. Simple names that end with the label '$' have this character replaced by the default directory (see *nis_local_directory*(3N)). Using "$" as a name in this list results in this name being replaced by the list of directories between the default directory and the global root that contain at least two labels.

Below is an example of this expansion. Given the default directory of *some.long.domain.name.*, and the **NIS_PATH** variable set to **fred.bar.:org_dir.$:$**. This path is initially broken up into the list:

1 **fred.bar.**

2 **org_dir.$**

3 **$**

The dollar sign in the second component is replaced by the default directory. The dollar sign in the third component is replaced with the names of the directories between the default directory and the global root that have at least two labels in them. The effective path value becomes:

1 **fred.bar.**

2a **org_dir.some.long.domain.name.**

3a **some.long.domain.name.**

3b **long.domain.name.**

3c **domain.name.**

Each of these simple names is appended to the partially qualified name that was passed to the *nis_lookup*(3N) or *nis_list*(3N) interface. Each is tried in turn until **NIS_SUCCESS** is returned or the list is exhausted.

If the **NIS_PATH** variable is not set, the path "$" is used.

The library function *nis_getnames*(3N) can be called from user programs to generate the list of names that would be attempted. The program *nisdefaults*(1) with the **-s** option can also be used to show the fully expanded path.

Concatenation Path

Normally all the entries for a certain type of information are stored within the table itself. However, there are times when it is desirable for the table to point to other tables where entries can be found. For example, you may want to store all the IP addresses in the host table for their own domain, and yet want to be able to resolve hosts in some other domain without explicitly specifying the new domain name. NIS+ provides a mechanism for concatenating different but related tables with an "NIS+ Concatenation Path". With a concatenation path, you can create a sort of flat namespace from a hierarchical structure. You can also create a table with no entries and just point the hosts or any other table to its parent domain. Note that with such a setup, you are moving the administrative burden of managing the tables to the parent domain. The concatenation path will slow down the request response time because more tables and more servers are searched. It will also decrease the availability if all the servers are incapacitated for a particular directory in the table path.

The NIS+ Concatenation Path is also referred to as the "table path". This path is set up at table creation time through *nistbladm*(1). You can specify more than one table to be concatenated and they will be searched in the given order. Note that the NIS+ client libraries, by default, will not follow the concatenation path set in site-specific tables. Refer to *nis_list*(3N) for more details.

Namespaces

The NIS+ service defines two additional *disjoint* namespaces for its own use. These namespaces are the NIS+ *Principal* namespace, and the NIS+ *Group* namespace. The names associated with the group and principal namespaces are syntactically identical to simple names. However, the information they represent *cannot* be obtained by directly presenting these names to the NIS+ interfaces. Instead, special interfaces are defined to map these names into NIS+ names so that they may then be resolved.

Principal Names

NIS+ principal names are used to uniquely identify users and machines that are making NIS+ requests. These names have the form:

principal.domain

Here *domain* is the fully qualified name of an NIS+ directory where the named principal's credentials can be found. See *Directories and Domains* for more information on domains. Note that in this name, *principal*, is not a leaf in the NIS+ namespace.

Credentials are used to map the identity of a host or user from one context such as a process UID into the NIS+ context. They are stored as records in an NIS+ table named *cred*, which always appears in the *org_dir* subdirectory of the directory named in the principal name.

This mapping can be expressed as a replacement function:

principal.domain -> **[cname**=*principal.domain* **],cred.org_dir**.*domain*

This latter name is an NIS+ name that can be presented to the *nis_list*(3N) interface for resolution. NIS+ principal names are administered using the *nisaddcred*(1M) command.

The *cred* table contains five columns named *cname*, *auth_name*, *auth_type*, *public_data*, and *private_data*. There is one record in this table for each identity mapping for an NIS+ principal. The current service supports two such mappings:

LOCAL This mapping is used to map from the UID of a given process to the NIS+ principal name associated with that UID. If no mapping exists, the name *nobody* is returned. When the effective UID of the process is 0 (for example, the super-user), the NIS+ name associated with the host is returned. Note that UIDs are sensitive to the context of the machine on which the process is executing.

DES This mapping is used to map to and from a Secure RPC "netname" into an NIS+ principal name. See *secure_rpc*(3N) for more information on netnames. Note that since netnames contain the notion of a domain, they span NIS+ directories.

The NIS+ client library function *nis_local_principal*(3N) uses the *cred.org_dir* table to map the UNIX notion of an identity, a process' UID, into an NIS+ principal name. Shell programs can use the program *nisdefaults*(1) with the **-p** switch to return this information.

Mapping from UIDs to an NIS+ principal name is accomplished by constructing a query of the form:

[auth_type=LOCAL, auth_name=*uid***],cred.org_dir.***default-domain***.**

This query will return a record containing the NIS+ principal name associated with this UID in the machine's default domain.

The NIS+ service uses the DES mapping to map the names associated with Secure RPC requests into NIS+ principal names. RPC requests that use Secure RPC include the *netname* of the client making the request in the RPC header. This netname has the form:

unix.*UID@domain*

The service constructs a query using this name of the form:

[auth_type=DES, auth_name=*netname***],cred.org_dir.***domain***.**

where the domain part is extracted from the netname rather than using the default domain. This query is used to look up the mapping of this netname into an NIS+ principal name in the domain where it was created.

This mechanism of mapping UID and netnames into an NIS+ principal name guarantees that a client of the NIS+ service has only one principal name. This principal name is used as the basis for authorization which is described below. All objects in the NIS+ namespace and all entries in NIS+ tables must have an owner specified for them. This owner field always contains an NIS+ principal name.

Group Names

Like NIS+ principal names, NIS+ group names take the form:

group_name`.domain`

All objects in the NIS+ namespace and all entries in NIS+ tables may optionally have a *group owner* specified for them. This group owner field, when filled in, always contains the fully qualified NIS+ group name.

The NIS+ client library defines several interfaces (*nis_groups*(3N)) for dealing with NIS+ groups. These interfaces internally map NIS+ group names into an NIS+ simple name which identifies the NIS+ group object associated with that group name. This mapping can be shown as follows:

group.domain **->** *group*`.groups_dir`*.domain*

This mapping eliminates collisions between NIS+ group names and NIS+ directory names. For example, without this mapping, a directory with the name *engineering.foo.com.*, would make it impossible to have a group named *engineering.foo.com.*. This is due to the restriction that within the NIS+ namespace, a name unambiguously identifies a single object. With this mapping, the NIS+ *group* name *engineering.foo.com.* maps to the NIS+ *object* name *engineering.groups_dir.foo.com.*

The contents of a group object is a list of NIS+ principal names and the names of other NIS+ groups. See *nis_groups*(3N) for a more complete description of their use.

NIS+ SECURITY

NIS+ defines a security model to control access to information managed by the service. The service defines access rights that are selectively granted to individual clients or groups of clients. Principal names and group names are used to define clients and groups of clients that may be granted or denied access to NIS+ information. These principals and groups are associated with NIS+ domains as defined below.

The security model also uses the notion of a class of principals called *nobody*, which contains all clients, whether or not they have authenticated themselves to the service. The class *world* includes any client who has been authenticated.

Directories and Domains

Some directories within the NIS+ namespace are referred to as NIS+ *Domains*. Domains are those NIS+ directories that contain the subdirectories *groups_dir* and *org_dir*. Further, the subdirectory *org_dir* should contain the table named *cred*. NIS+ Group names and NIS+ Principal names *always* include the NIS+ domain name after their first label.

Authentication

The NIS+ name service uses Secure RPC for the integrity of the NIS+ service. This requires that users of the service and their machines must have a Secure RPC key pair associated with them. This key is initially generated with either the *nisaddcred*(1M) or *nisclient*(1M) commands and modified with the *chkey*(1) or *nispasswd*(1) commands.

The use of Secure RPC allows private information to be stored in the name service that will not be available to untrusted machines or users on the network.

In addition to the Secure RPC key, users need a mapping of their UID into an NIS+ principal name. This mapping is created by the system administrator using the *nisclient*(1M) or *nisaddcred*(1M) command.

Users that will be using machines in several NIS+ domains must insure that they have a *local* credential entry in each of those domains. This credential should be created with the NIS+ principal name of the user in their "home" domain. For the purposes of NIS+ and Secure RPC, the home domain is defined to be the one where your Secure RPC key pair is located.

Authorization

The NIS+ service defines four access rights that can be granted or denied to clients of the service. These rights are *read*, *modify*, *create*, and *destroy*. These rights are specified in the object structure at creation time and may be modified later with the *nischmod*(1) command. In general, the rights granted for an object apply only to that object. However, for purposes of authorization, rights granted to clients reading *directory* and *table* objects are granted to those clients for all of the objects "contained" by the parent object. This notion of containment is abstract. The objects do not actually contain other objects within them. Note that *group* objects do contain the list of principals within their definition.

Access rights are interpreted as follows:

read This right grants read access to an object. For directory and table objects, having read access on the parent object conveys read access to all of the objects that are direct children of a directory, or entries within a table.

modify This right grants modification access to an existing object. Read access is not required for modification. However, in many applications, one will need to read an object before modifying it. Such modify operations will fail unless read access is also granted.

create This right gives a client permission to create new objects where one had not previously existed. It is only used in conjunction with directory and table objects. Having create access for a table allows a client to add additional entries to the table. Having create access for a directory allows a client to add new objects to an NIS+ directory.

destroy This right gives a client permission to destroy or remove an existing object or entry. When a client attempts to destroy an entry or object by removing it, the service first checks to see if the table or directory containing that object grants the client destroy access. If it does, the operation proceeds. If the containing object does not grant this right then the object itself is checked to see if it grants this right to the client. If the object grants the right, then the operation proceeds; otherwise the request is rejected.

Each of these rights may be granted to any one of four different categories.

owner A right may be granted to the *owner* of an object. The owner is the NIS+ principal identified in the owner field. The owner can be changed with the *nischown*(1) command. Note that if the owner does not have modification access rights to the object, the owner cannot change any access rights to the object, unless the owner has modification access rights to its parent object.

group owner
A right may be granted to the *group owner* of an object. This grants the right to any principal that is identified as a member of the group associated with the object. The group owner may be changed with the *nischgrp*(1) command. The object owner need not be a member of this group.

world A right may be granted to everyone in the *world*. This grants the right to all clients who have authenticated themselves with the service.

nobody A right may be granted to the *nobody* principal. This has the effect of granting the right to any client that makes a request of the service, regardless of whether they are authenticated or not.

Note that for bootstrapping reasons, directory objects that are NIS+ domains, the *org_dir* subdirectory and the *cred* table within that subdirectory must have *read* access to the *nobody* principal. This makes navigation of the namespace possible when a client is in the process of locating its credentials. Granting this access does not allow the contents of other tables within *org_dir* to be read (such as the entries in the password table) unless the table itself gives "read" access rights to the *nobody* principal.

Directory Authorization

Additional capabilities are provided for granting access rights to clients for directories. These rights are contained within the *object access rights* (OAR) structure of the directory. This structure allows the NIS+ service to grant rights that are not granted by the directory object to be granted for objects contained by the directory of a specific type.

An example of this capability is a directory object which does not grant create access to all clients, but does grant create access in the OAR structure for *group* type objects to clients who are members of the NIS+ group associated with the directory. In this example the only objects that could be created as children of the directory would have to be of the type *group*.

Another example is a directory object that grants create access only to the owner of the directory, and then additionally grants create access through the OAR structure for objects of type *table*, *link*, *group*, and *private* to any member of the directory's group. This has the effect of giving nearly complete create access to the group with the exception of creating subdirectories. This restricts the creation of new NIS+ domains because creating a domain requires creating both a *groups_dir* and *org_dir* subdirectory.

Note that there is currently no command line interface to set or change the OAR of the directory object.

Table Authorization

As with directories, additional capabilities are provided for granting access to entries within tables. Rights granted to a client by the access rights field in a table object apply to the table object and all of the entry objects "contained" by that table. If an access right is not granted by the table object, it may be granted by an entry within the table. This holds for all rights except *create*.

For example, a table may not grant read access to a client performing a *nis_list*(3N) operation on the table. However, the access rights field of entries within that table may grant read access to the client. Note that access rights in an entry are granted to the owner and group owner of the *entry* and not the owner or group of the table. When the list operation is performed, all entries that the client has read access to are returned. Those entries that do not grant read access are not returned. If none of the entries that match the search criterion grant read access to the client making the request, no entries are returned and the result status contains the **NIS_NOTFOUND** error code.

Access rights that are granted by the rights field in an entry are granted for the entire entry. However, in the table object an additional set of access rights is maintained for each column in the table. These rights apply to the equivalent column in the entry. The rights are used to grant access when neither the table nor the entry itself grant access. The access rights in a column specification apply to the owner and group owner of the entry rather than the owner and group owner of the table object.

When a read operation is performed, if read access is not granted by the table and is not granted by the entry but *is* granted by the access rights in a column, that entry is returned with the correct values in all columns that are readable and the string ***NP*** (No Permission) in columns where read access is not granted.

As an example, consider a client that has performed a list operation on a table that does not grant read access to that client. Each entry object that satisfied the search criterion specified by the client is examined to see if it grants read access to the client. If it does, it is included in the returned result. If it does not, then each column is checked to see if it grants read access to the client. If any columns grant read access to the client, data in those columns is returned. Columns that do not grant read access have their contents replaced by the string ***NP***. If none of the columns grant read access, then the entry is not returned.

LIST OF COMMANDS

The following lists all commands and programming functions related to NIS+:

NIS+ User Commands

nisaddent(1)	add /etc files and NIS maps into their corresponding NIS+ tables
niscat(1)	display NIS+ tables and objects
nischgrp(1)	change the group owner of a NIS+ object
nischmod(1)	change access rights on a NIS+ object
nischown(1)	change the owner of a NIS+ object
nischttl(1)	change the time to live value of a NIS+ object
nisdefaults(1)	display NIS+ default values
niserror(1)	display NIS+ error messages
nisgrep(1)	utilities for searching NIS+ tables
nisgrpadm(1)	NIS+ group administration command
nisln(1)	symbolically link NIS+ objects
nisls(1)	list the contents of a NIS+ directory
nismatch(1)	utilities for searching NIS+ tables
nismkdir(1)	create NIS+ directories
nispasswd(1)	change NIS+ password information
nisrm(1)	remove NIS+ objects from the namespace
nisrmdir(1)	remove NIS+ directories
nisshowcache(1)	NIS+ utility to print out the contents of the shared cache file
nistbladm(1)	NIS+ table administration command
nistest(1)	return the state of the NIS+ namespace using a conditional expression

NIS+ Administrative Commands

nis_cachemgr(1M)	NIS+ utility to cache location information about NIS+ servers
nisaddcred(1M)	create NIS+ credentials
nisaddent(1M)	create NIS+ tables from corresponding /etc files or NIS maps
nisclient(1M)	initialize NIS+ credentials for NIS+ principals
nisd(1M)	NIS+ service daemon
nisd_resolv(1M)	NIS+ service daemon

nisinit(1M)	NIS+ client and server initialization utility
nislog(1M)	display the contents of the NIS+ transaction log
nisping(1M)	send ping to NIS+ servers
nispopulate(1M)	populate the NIS+ tables in a NIS+ domain
nisserver(1M)	set up NIS+ servers
nissetup(1M)	initialize a NIS+ domain
nisshowcache(1M)	NIS+ utility to print out the contents of the shared cache file
nisstat(1M)	report NIS+ server statistics
nisupdkeys(1M)	update the public keys in a NIS+ directory object
rpc.nisd(1M)	NIS+ service daemon
rpc.nisd_resolv(1M)	NIS+ service daemon

NIS+ Programming API

__nis_map_group(3N)	NIS+ group manipulation functions
db_add_entry(3N)	NIS+ Database access functions
db_checkpoint(3N)	NIS+ Database access functions
db_create_table(3N)	NIS+ Database access functions
db_destroy_table(3N)	NIS+ Database access functions
db_first_entry(3N)	NIS+ Database access functions
db_free_result(3N)	NIS+ Database access functions
db_initialize(3N)	NIS+ Database access functions
db_list_entries(3N)	NIS+ Database access functions
db_next_entry(3N)	NIS+ Database access functions
db_remove_entry(3N)	NIS+ Database access functions
db_reset_next_entry(3N)	NIS+ Database access functions
db_standby(3N)	NIS+ Database access functions
db_table_exists(3N)	NIS+ Database access functions
db_unload_table(3N)	NIS+ Database access functions
nis_add(3N)	NIS+ namespace functions
nis_add_entry(3N)	NIS+ table functions
nis_addmember(3N)	NIS+ group manipulation functions
nis_checkpoint(3N)	misc NIS+ log administration functions
nis_clone_object(3N)	NIS+ subroutines
nis_creategroup(3N)	NIS+ group manipulation functions
nis_db(3N)	NIS+ Database access functions
nis_destroy_object(3N)	NIS+ subroutines
nis_destroygroup(3N)	NIS+ group manipulation functions
nis_dir_cmp(3N)	NIS+ subroutines
nis_domain_of(3N)	NIS+ subroutines
nis_error(3N)	display NIS+ error messages
nis_first_entry(3N)	NIS+ table functions
nis_freenames(3N)	NIS+ subroutines
nis_freeresult(3N)	NIS+ namespace functions
nis_freeservlist(3N)	miscellaneous NIS+ functions
nis_freetags(3N)	miscellaneous NIS+ functions
nis_getnames(3N)	NIS+ subroutines
nis_getservlist(3N)	miscellaneous NIS+ functions
nis_groups(3N)	NIS+ group manipulation functions
nis_ismember(3N)	NIS+ group manipulation functions
nis_leaf_of(3N)	NIS+ subroutines
nis_lerror(3N)	display some NIS+ error messages
nis_list(3N)	NIS+ table functions
nis_local_directory(3N)	NIS+ local names
nis_local_group(3N)	NIS+ local names
nis_local_host(3N)	NIS+ local names
nis_local_names(3N)	NIS+ local names
nis_local_principal(3N)	NIS+ local names
nis_lookup(3N)	NIS+ namespace functions
nis_map_group(3N)	NIS+ group manipulation functions
nis_mkdir(3N)	miscellaneous NIS+ functions
nis_modify(3N)	NIS+ namespace functions

nis_modify_entry(3N)	NIS+ table functions
nis_name_of(3N)	NIS+ subroutines
nis_names(3N)	NIS+ namespace functions
nis_next_entry(3N)	NIS+ table functions
nis_objects(3N)	NIS+ object formats
nis_perror(3N)	display NIS+ error messages
nis_ping(3N)	misc NIS+ log administration functions
nis_print_group_entry(3N)	NIS+ group manipulation functions
nis_print_object(3N)	NIS+ subroutines
nis_remove(3N)	NIS+ namespace functions
nis_remove_entry(3N)	NIS+ table functions
nis_removemember(3N)	NIS+ group manipulation functions
nis_rmdir(3N)	miscellaneous NIS+ functions
nis_server(3N)	miscellaneous NIS+ functions
nis_servstate(3N)	miscellaneous NIS+ functions
nis_sperrno(3N)	display NIS+ error messages
nis_sperror(3N)	display NIS+ error messages
nis_sperror_r(3N)	display NIS+ error messages
nis_stats(3N)	miscellaneous NIS+ functions
nis_subr(3N)	NIS+ subroutines
nis_tables(3N)	NIS+ table functions
nis_verifygroup(3N)	NIS+ group manipulation functions

NIS+ Files and Directories

nisfiles(4)	NIS+ database files and directory structure

AUTHOR

NIS+ was developed by Sun Microsystems, Inc.

FILES

<rpcsvc/nis_object.h>	Protocol description of an NIS+ object.
<rpcsvc/nis.h>	Defines the NIS+ protocol using the RPC language. It should be included by all clients of the NIS+ service

n

SEE ALSO

nischown(1), nisdefaults(1), nismatch(1), nispasswd(1), newkey(1M), nisaddcred(1M), nisclient(1M), nispopulate(1M), nisserver(1M), nis_add_entry(3N), nis_domain_of(3N), nis_getnames(3N), nis_groups(3N), nis_leaf_of(3N), nis_list(3N), nis_local_directory(3N), nis_lookup(3N), nis_objects(3N).

NAME

niscat - display NIS+ tables and objects

SYNOPSIS

niscat [**-AhLMv**] *tablename*...

niscat [**-ALMP**] **-o** *name*...

DESCRIPTION

In the first synopsis, **niscat** displays the contents of the NIS+ tables named by *tablename*. In the second synopsis, it displays the internal representation of the NIS+ objects named by *name*.

Options

-A Display the data within the table and all of the data in tables in the initial table's concatenation path.

-h Display the header line prior to displaying the table. The header consists of the '#' (hash) character followed by the name of each column. The column names are separated by the table separator character.

-L Follow links. When this option is specified, if *tablename* or *name* names a LINK type object, the link is followed and the object or table named by the link is displayed.

-M Master server only. This option specifies that the request should be sent to the master server of the named data. This guarantees that the most up-to-date information is seen at the possible expense of increasing the load on the master server and increasing the possibility of the NIS+ server being unavailable or busy for updates.

-P Follow concatenation path. This option specifies that the request should follow the concatenation path of a table if the initial search is unsuccessful. This option is only useful when using an indexed name for *name* and the **-o** option.

-v Display binary data directly. This option displays columns containing binary data on the standard output. Without this option, binary data is displayed as the string ***BINARY***.

-o *name* Display the internal representation of the named NIS+ object(s). If *name* is an indexed name (see *nismatch*(1)), then each of the matching entry objects is displayed. This option is used to display access rights and other attributes of individual columns.

EXAMPLES

Display the contents of the hosts table:

```
niscat -h host.org_dir
# cname name    addr            comment
client1 client1 129.144.201.100 Joe Smith
crunchy crunchy 129.144.201.44  Jane Smith
crunchy softy   129.144.201.44
```

The string ***NP*** is returned in those fields where the user has insufficient access rights.

Display the **passwd.org_dir** on the standard output:

```
niscat passwd.org_dir
```

Display the contents of table **frodo** and the contents of all tables in its concatenation path:

```
niscat -A frodo
```

Display the entries in the table **group.org_dir** as NIS+ objects (note that the brackets are protected from the shell by single quotes):

```
niscat -o '[ ]group.org_dir'
```

Display the table object of the **passwd.org_dir** table:

```
niscat -o passwd.org_dir
```

The previous example displays the passwd table object and not the passwd table. The table object includes information such as the number of columns, column type, searchable or not searchable, separator, access rights, and other defaults.

Display the directory object for `org_dir`, which includes information such as the access rights and replica information:

```
niscat -o org_dir
```

EXTERNAL INFLUENCES

Environment Variables

NIS_PATH If this variable is set and the NIS+ table name is not fully qualified, each directory specified will be searched until the table is found (see *nisdefaults*(1)).

RETURN VALUE

niscat returns **0** on success and **1** on failure.

AUTHOR

niscat was developed by Sun Microsystems, Inc.

SEE ALSO

nis+(1), nismatch(1), nistbladm(1), nisdefaults(1), nis_objects(3N), nis_tables(3N).

NOTES

Columns without values in the table are displayed by two adjacent separator characters.

NAME

nischgrp - change the group owner of an NIS+ object

SYNOPSIS

nischgrp [**-AfLP**] *group name . . .*

DESCRIPTION

nischgrp changes the group owner of the NIS+ objects or entries specified by *name* to the specified NIS+ *group*. Entries are specified using indexed names (see *nismatch*(1)). If *group* is not a fully qualified NIS+ group name, it will be resolved using the directory search path (see *nisdefaults*(1)).

The only restriction on changing an object's group owner is that you must have modify permissions for the object.

This command will fail if the master NIS+ server is not running.

Options

-A Modify all entries in all tables in the concatenation path that match the search criterion specified in *name*. This option implies the **-P** switch.

-f Force the operation and fail silently if it does not succeed.

-L Follow links and change the group owner of the linked object or entries rather than the group owner of the link itself.

-P Follow the concatenation path within a named table. This option only makes sense when either *name* is an indexed name or the **-L** switch is also specified and the named object is a link pointing to entries.

EXAMPLES

Change the group owner of an object to a group in a different domain, and how to change it to a group in the local domain, respectively:

```
nischgrp  newgroup.remote.domain.  object
nischgrp  my-buds  object
```

Change the group owner for a password entry:

```
nischgrp  admins  '[uid=99],passwd.org_dir'
```

In the above example, **admins** is an NIS+ group in the same domain.

Change the group owner of the object or entries pointed to by a link, and the group owner of all entries in the **hobbies** table:

```
nischgrp  -L  my-buds  linkname
nischgrp  my-buds  '[],hobbies'
```

EXTERNAL INFLUENCES

Environment Variables

NIS_PATH If this variable is set and the NIS+ name is not fully qualified, each directory specified will be searched until the object is found (see *nisdefaults*(1)).

RETURN VALUE

nischgrp returns **0** on success and **1** on failure.

AUTHOR

nischgrp was developed by Sun Microsystems, Inc.

SEE ALSO

nis+(1), nischmod(1), nischown(1), nisdefaults(1), nisgrpadm(1), nis_objects(3N).

NOTES

The NIS+ server will check the validity of the group name prior to effecting the modification.

NAME

nischmod - change access rights on an NIS+ object

SYNOPSIS

nischmod [**-AfLP**] *mode name*...

DESCRIPTION

nischmod changes the access rights (mode) of the NIS+ objects or entries specified by *name* to *mode*. Entries are specified using indexed names (see *nismatch*(1)). Only principals with modify access to an object may change its mode.

mode has the following form:

rights [,*rights*] ...

rights has the form:

[*who*] *op permission* [*op permission*]...

who is a combination of:

n	Nobody's permissions.
o	Owner's permissions.
g	Group's permissions.
w	World's permissions.
a	All, or **owg**.

If *who* is omitted, the default is **a**.

op is one of:

+	To grant the *permission*.
-	To revoke the *permission*.
=	To set the permissions explicitly.

permission is any combination of:

r	Read.
m	Modify.
c	Create.
d	Destroy.

Options

-A Modify all entries in all tables in the concatenation path that match the search criteria specified in *name*. This option implies the **-P** switch.

-f Force the operation and fail silently if it does not succeed.

-L Follow links and change the permission of the linked object or entries rather than the permission of the link itself.

-P Follow the concatenation path within a named table. This option is only applicable when either *name* is an indexed name or the **-L** switch is also specified and the named object is a link pointing to an entry.

EXAMPLES

Give everyone read access to an object (that is, access for owner, group, and all):

nischmod a+r *object*

Deny create and modify privileges to **group** and unauthenticated clients (**nobody**):

nischmod gn-cm *object*

Set a complex set of permissions for an object:

nischmod o=rmcd,g=rm,w=rc,n=r *object*

Set the permissions of an entry in the password table so that the group owner can modify them:

nischmod g+m '[uid=55],passwd.org_dir'

Change the permissions of a linked object:

```
nischmod -L w+mr linkname
```

EXTERNAL INFLUENCES

Environment Variables

NIS_PATH If this variable is set and the NIS+ name is not fully qualified, each directory specified will be searched until the object is found (see *nisdefaults*(1)).

RETURN VALUE

nischmod returns **0** on success and **1** on failure.

AUTHOR

nischmod was developed by Sun Microsystems, Inc.

SEE ALSO

chmod(1), nis+(1), nischgrp(1), nischown(1), nisdefaults(1), nis_objects(3N).

NOTES

Unlike the system **chmod** command, this command does not accept an octal notation.

NAME

nischown - change the owner of an NIS+ object

SYNOPSIS

nischown [**-AfLP**] *owner name*...

DESCRIPTION

nischown changes the owner of the NIS+ objects or entries specified by *name* to *owner*. Entries are specified using indexed names (see *nismatch*(1)). If *owner* is not a fully qualified NIS+ principal name (see *nisaddcred*(1M)), the default domain (see *nisdefaults*(1)) will be appended to it.

The only restriction on changing an object's owner is that you must have modify permissions for the object. Note: If you are the current owner of an object and you change ownership, you may not be able to regain ownership unless you have modify access to the new object.

The command will fail if the master NIS+ server is not running.

Options

-A Modify all entries in all tables in the concatenation path that match the search criteria specified in *name*. It implies the **-P** option.

-f Force the operation and fail silently if it does not succeed.

-L Follow links and change the owner of the linked object or entries rather than the owner of the link itself.

-P Follow the concatenation path within a named table. This option is only meaningful when either *name* is an indexed name or the **-L** option is also specified and the named object is a link pointing to entries.

EXAMPLES

Change the owner of an object to a principal in a different domain, and to change it to a principal in the local domain, respectively:

```
nischown bob.remote.domain. object
nischown skippy object
```

Change the owner of an entry in the passwd table:

```
nischown bob.remote.domain. '[uid=99],passwd.org_dir'
```

Change the object or entries pointed to by a link:

```
nischown -L skippy linkname
```

EXTERNAL INFLUENCES

Environment Variables

NIS_PATH If this variable is set, and the NIS+ name is not fully qualified, each directory specified will be searched until the object is found (see *nisdefaults*(1)).

RETURN VALUE

nischown returns **0** on success and **1** on failure.

AUTHOR

nischown was developed by Sun Microsystems, Inc.

SEE ALSO

nis+(1), nischgrp(1), nischmod(1), nischttl(1), nisdefaults(1), nisaddcred(1M), nis_objects(3N).

NOTES

The NIS+ server will check the validity of the name before making the modification.

NAME

nischttl - change the time to live value of an NIS+ object

SYNOPSIS

nischttl [**-AfLP**] *time name*...

DESCRIPTION

nischttl changes the time to live value (**ttl**) of the NIS+ objects or entries specified by *name* to *time*. Entries are specified using indexed names (see *nismatch*(1)).

The time to live value is used by object caches to expire objects within their cache. When an object is read into the cache, this value is added to the current time in seconds yielding the time when the cached object would expire. The object may be returned from the cache as long as the current time is earlier than the calculated expiration time. When the expiration time has been reached, the object will be flushed from the cache.

The time to live *time* may be specified in seconds or in days, hours, minutes, seconds format. The latter format uses a suffix letter of **d**, **h**, **m**, or **s** to identify the units of time. See the examples below for usage.

The command will fail if the master NIS+ server is not running.

Options

-A Modify all tables in the concatenation path that match the search criterion specified in *name*. This option implies the **-P** switch.

-f Force the operation and fail silently if it does not succeed.

-L Follow links and change the time to live of the linked object or entries rather than the time to live of the link itself.

-P Follow the concatenation path within a named table. This option only makes sense when either *name* is an indexed name or the **-L** switch is also specified and the named object is a link pointing to entries.

n

EXAMPLES

Change the **ttl** of an object using the seconds format and the days, hours, minutes, seconds format (the **ttl** of the second object is set to 1 day and 12 hours):

```
nischttl 184000 object
nischttl 1d12h object
```

Change the **ttl** for a password entry:

```
nischttl 1h30m '[uid=99],passwd.org_dir'
```

Change the **ttl** of the object or entries pointed to by a link, and the **ttl** of all entries in the **hobbies** table:

```
nischttl -L 12h linkname
nischttl 3600 '[],hobbies'
```

EXTERNAL INFLUENCES

Environment Variables

NIS_PATH If this variable is set and the NIS+ name is not fully qualified, each directory specified will be searched until the object is found (see *nisdefaults*(1)).

RETURN VALUE

nischttl returns **0** on success and **1** on failure.

AUTHOR

nischttl was developed by Sun Microsystems, Inc.

SEE ALSO

nis+(1), nischgrp(1), nischmod(1), nischown(1), nisdefaults(1), nis_objects(3N).

NOTES

Setting a high **ttl** value allows objects to stay persistent in caches for a longer period of time and can improve performance. However, when an object changes, in the worst case, the number of seconds in this attribute must pass before that change is visible to all clients. Setting a **`ttl`** value of **`0`** means that the object should not be cached at all.

A high **`ttl`** value is a week, a low value is less than a minute. Password entries should have **`ttl`** values of about 12 hours (easily allows one password change per day), entries in the RPC table can have **`ttl`** values of several weeks (this information is effectively unchanging).

Only directory and group objects are cached in this implementation.

NAME
nisdefaults - display NIS+ default values

SYNOPSIS
nisdefaults [**-adghprstv**]

DESCRIPTION
nisdefaults prints the default values that are returned by calls to the NIS+ local name functions (see *nis_local_names*(3N)). With no options specified, all defaults will be printed in a verbose format. With options, only that option is displayed in a terse form suitable for shell scripts. See the example below.

Options

-a	Print all defaults in a terse format.
-d	Print the default domain name.
-g	Print the default group name.
-h	Print the default host name.
-p	Print the default principal name.
-r	Print the default access rights with which new objects will be created.
-s	Print the default directory search path.
-t	Print the default time to live value.
-v	Print the defaults in a verbose format. This prepends an identifying string to the output.

EXAMPLES
Print the NIS+ defaults for a root process on machine **example** in the **foo.bar.** domain:

```
example# nisdefaults
Principal Name : example.foo.bar.
Domain Name    : foo.bar.
Host Name      : example.foo.bar.
Group Name     :
Access Rights  : ----rmcdr---r---
Time to live   : 12:00:00
Search Path    : foo.bar.
```

Set a variable in a shell script to the default domain:

```
DOMAIN=‘nisdefaults -d‘
```

Print out the default time to live in a verbose format:

```
nisdefaults -tv
Time to live : 12:00:00
```

Print out the time to live in the terse format:

```
nisdefaults -t
43200
```

EXTERNAL INFLUENCES
Environment Variables
Several environment variables affect the defaults associated with a process.

NIS_DEFAULTS This variable contains a defaults string that will override the NIS+ standard defaults. The defaults string is a series of tokens separated by colons. These tokens represent the default values to be used for the generic object properties. All of the legal tokens are described below.

ttl=*time*
This token sets the default time to live for objects that are created. The value *time* is specified in the format as defined by the **nischttl** command. (See *nischttl*(1)). The default value is **12** hours.

n

owner=*ownername*
: This token specifies that the NIS+ principal *ownername* should own created objects. The default for this value is the principal who is executing the command.

group=*groupname*
: This token specifies that the group *groupname* should be the group owner for created objects. The default is **NULL**.

access=*rights*
: This token specifies the set of access rights that are to be granted for created objects. The value *rights* is specified in the format as defined by the **nischmod** command. (See *nischmod*(1)). The default value is **----rmcdr---r---**.

NIS_GROUP This variable contains the name of the local NIS+ group. If the name is not fully qualified, the default domain will be appended to it.

NIS_PATH This variable overrides the default NIS+ directory search path. It contains an ordered list of directories separated by ':' (colon) characters. The '$' (dollar sign) character is treated specially. Directory names that end in '$' have the default domain appended to them, and a '$' by itself is replaced by the list of directories between the default domain and the global root that are at least two levels deep. The default NIS+ directory search path is '$'.

Refer to the *Name Expansion* subsection in *nis+*(1) for more details.

AUTHOR

nisdefaults was developed by Sun Microsystems, Inc.

SEE ALSO

nis+(1), nis_local_names(3N).

NAME

niserror - display NIS+ error messages

SYNOPSIS

niserror *error-num*

DESCRIPTION

niserror prints the NIS+ error associated with status value *error-num* on the standard output. It is used by shell scripts to translate NIS+ error numbers that are returned into text messages.

EXAMPLES

Print the error associated with the error number **20**:

```
niserror 20
Not Found, no such name
```

AUTHOR

niserror was developed by Sun Microsystems, Inc.

SEE ALSO

nis+(1), nis_error(3N).

NAME

nisgrpadm - NIS+ group administration command

SYNOPSIS

nisgrpadm -a | **-r** | **-t**] [**-s**] *group principal*...

nisgrpadm -c | **-d** | **-l** [**-M**] [**-s**] *group*

DESCRIPTION

nisgrpadm is used to administer NIS+ groups. This command administers both groups and the groups' membership lists. **nisgrpadm** can create, destroy, or list NIS+ groups. **nisgrpadm** can be used to administer a group's membership list. It can add or delete principals to the group, or test principals for membership in the group.

The names of NIS+ groups are syntactically similar to names of NIS+ objects but they occupy a separate namespace. A group named "a.b.c.d." is represented by a NIS+ group object named "a.groups_dir.b.c.d."; the functions described here all expect the name of the group, not the name of the corresponding group object.

There are three types of group members:

- An **explicit** member is just a NIS+ principal-name, for example "wickedwitch.west.oz."
- An **implicit** ("domain") member, written "*.west.oz.", means that all principals in the given domain belong to this member. No other forms of wildcarding are allowed: "wickedwitch.*.oz." is invalid, as is "wickedwitch.west.*.". Note that principals in subdomains of the given domain are *not* included.
- A **recursive** ("group") member, written "@cowards.oz.", refers to another group; all principals that belong to that group are considered to belong here.

Any member may be made **negative** by prefixing it with a minus sign ('-'). A group may thus contain explicit, implicit, recursive, negative explicit, negative implicit, and negative recursive members.

A principal is considered to belong to a group if it belongs to at least one non-negative group member of the group and belongs to no negative group members.

n

Options

-a Add the list of NIS+ principals specified to *group*. The principal name should be fully qualified.

-c Create *group* in the NIS+ namespace. The NIS+ group name should be fully qualified.

-d Destroy (remove) *group* from the namespace.

-l List the membership list of the specified *group*. (See **-M**.)

-M Master server only. Send the lookup to the master server of the named data. This guarantees that the most up to date information is seen at the possible expense that the master server may be busy. Note that the **-M** flag is applicable only with the **-l** flag.

-r Remove the list of principals specified from *group*. The principal name should be fully qualified.

-s Work silently. Results are returned using the exit status of the command. This status can be translated into a text string using the *niserror*(1) command.

-t Display whether the principals specified are members in *group*.

EXAMPLES

Administering Groups

Create a group in the **foo.com.** domain:

```
nisgrpadm -c my_buds.foo.com.
```

Remove the group from the current domain:

```
nisgrpadm -d freds_group
```

Administering Members

Add two principals, **bob** and **betty** to the group **my_buds.foo.com**:

```
nisgrpadm -a my_buds.foo.com. bob.bar.com. betty.foo.com.
```

Remove **betty** from **freds_group**:

```
nisgrpadm -r freds_group betty.foo.com.
```

EXTERNAL INFLUENCES

Environment Variables

NIS_PATH If this variable is set and the NIS+ group name is not fully qualified, each directory specified will be searched until the group is found (see *nisdefaults*(1)).

DIAGNOSTICS

NIS_SUCCESS On success, this command returns an exit status of **0**.

NIS_PERMISSION
When you do not have the needed access right to change the group, the command returns this error.

NIS_NOTFOUND This is returned when the group does not exist.

NIS_TRYAGAIN This error is returned when the server for the group's domain is currently checkpointing or otherwise in a read-only state. The command should be retried at a later date.

NIS_MODERROR This error is returned when the group was modified by someone else during the execution of the command. Reissue the command and optionally recheck the group's membership list.

AUTHOR

nisgrpadm was developed by Sun Microsystems, Inc.

SEE ALSO

nis+(1), nischgrp(1), nisdefaults(1), niserror(1), nis_groups(3N).

NOTES

Principal names *must* be fully qualified, whereas groups can be abbreviated on all operations *except* create.

NAME

nisln - symbolically link NIS+ objects

SYNOPSIS

nisln [**-L**] [**-D** *defaults*] *name linkname*

DESCRIPTION

The **nisln** command links an NIS+ object named *name* to an NIS+ name *linkname*. If *name* is an indexed name (see *nismatch*(1)), the link points to entries within an NIS+ table. Clients wishing to look up information in the name service can use the **FOLLOW_LINKS** flag to force the client library to follow links to the name they point to. Further, all of the NIS+ administration commands accept the **-L** switch indicating they should follow links (see *nis_names*(3N) for a description of the **FOLLOW_LINKS** flag).

Options

-L When present, this option specifies that this command should follow links. If *name* is itself a link, then this command will follow it to the linked object that it points to. The new link will point to that linked object rather than to *name*.

-D *defaults* Specify a different set of defaults to be used for the creation of the link object. The *defaults* string is a series of tokens separated by colons. These tokens represent the default values to be used for the generic object properties. All of the legal tokens are described below.

ttl=*time* This token sets the default time to live for objects that are created by this command. The value *time* is specified in the format as defined by the *nischttl*(1) command. The default is **12** hours.

owner=*ownername* This token specifies that the NIS+ principal *ownername* should own the created object. The default for this value is the the principal who is executing the command.

group=*groupname* This token specifies that the group *groupname* should be the group owner for the object that is created. The default is **NULL**.

access=*rights* This token specifies the set of access rights that are to be granted for the given object. The value *rights* is specified in the format as defined by the *nischmod*(1) command. The default value is **----rmcdr---r---**.

EXAMPLES

Create a link in the domain **foo.com.** named **hosts** that points to the object **hosts.bar.com**:

```
nisln hosts.bar.com. hosts.foo.com.
```

Make a link *example.sun.com.* that points to an entry in the hosts table in *eng.sun.com.*:

```
nisln '[name=example],hosts.eng.sun.com.' example.sun.com:
```

EXTERNAL INFLUENCES

Environment Variables

NIS_PATH If this variable is set and the NIS+ name is not fully qualified, each directory specified will be searched until the object is found (see *nisdefaults*(1)).

RETURN VALUE

nisln returns **0** on success and **1** on failure.

AUTHOR

nisln was developed by Sun Microsystems, Inc.

SEE ALSO

nisdefaults(1), nismatch(1), nisrm(1), nistbladm(1), nis_names(3N), nis_tables(3N).

NOTES

When creating the link, **nisln** verifies that the linked object exists. Once created, the linked object may be deleted or replaced and the link will not be affected. At that time the link will become invalid and attempts to follow it will return **`NIS_LINKNAMEERROR`** to the client. When the path attribute in tables specifies a link rather than another table, the link will be followed if the flag **`FOLLOW_LINKS`** was present in the call to **`nis_list`** () (see *nis_tables*(3N)) and ignored if the flag is not present. If the flag is present and the link is no longer valid, a warning is sent to the system logger and the link is ignored.

NAME

nisls - list the contents of an NIS+ directory

SYNOPSIS

nisls [**-dglLmMR**] [*name* ...]

DESCRIPTION

For each *name* that is an NIS+ directory, **nisls** lists the contents of the directory. For each *name* that is an NIS+ object other than a directory, **nisls** simply echos the name. If no *name* is specified, the first directory in the search path (see *nisdefaults*(1)) is listed.

Options

-d Treat NIS+ directories like other NIS+ objects, rather than listing their contents.

-g Display group owner instead of owner when listing in long format.

-l List in long format. This option displays additional information about the objects such as their type, creation time, owner, and access rights.

The access rights are listed in the following order in long mode: nobody, owner, group owner, and world.

-L This option specifies that links are to be followed. If *name* actually points to a link, it is followed to the linked object.

-m Display modification time instead of creation time when listing in long format.

-M Master only. This specifies that information is to be returned from the master server of the named object. This guarantees that the most up-to-date information is seen at the possible expense that the master server may be busy.

-R List directories recursively. This option will reiterate the list for each subdirectory found in the process of listing each *name*.

n

EXTERNAL INFLUENCES

Environment Variables

NIS_PATH If this variable is set and the NIS+ name is not fully qualified, each directory specified will be searched until the object is found (see *nisdefaults*(1)).

RETURN VALUE

nisls returns **0** on success and **1** on failure.

AUTHOR

nisls was developed by Sun Microsystems, Inc.

SEE ALSO

nisdefaults(1), nisgrpadm(1), nismatch(1), nistbladm(1), nis_objects(3N).

NAME

nismatch, nisgrep - utilities for searching NIS+ tables

SYNOPSIS

nismatch [**-AchMoPv**] *key tablename*

nismatch [**-AchMoPv**] *colname=key... tablename*

nismatch [**-AchMoPv**] *indexedname*

nisgrep [**-AchMov**] *keypat tablename*

nisgrep [**-AchMov**] *colname=keypat... tablename*

DESCRIPTION

nismatch and **nisgrep** can be used to search NIS+ tables. The command **nisgrep** differs from the **nismatch** command in its ability to accept regular expressions *keypat* for the search criteria rather than simple text matches.

Because **nisgrep** uses a callback function, it is not constrained to searching only those columns that are specifically made searchable at the time of table creation. This makes it more flexible, but slower than **nismatch**.

In **nismatch**, the server does the searching; whereas in **nisgrep**, the server returns all the readable entries and then the client does the pattern-matching.

In both commands, the parameter *tablename* is the NIS+ name of the table to be searched. If only one key or key pattern is specified without the column name, then it is applied searching the first column. Specific named columns can be searched by using the *colname=key* syntax. When multiple columns are searched, only entries that match in all columns are returned. This is the equivalent of a logical join operation.

nismatch accepts an additional form of search criteria, *indexedname*, which is a NIS+ indexed name of the form:

[*colname***=***value***, ...],***tablename*

Options

-A All data. Return the data within the table and all of the data in tables in the initial table's concatenation path.

-c Print only a count of the number of entries that matched the search criteria.

-h Display a header line before the matching entries that contains the names of the table's columns

-M Master server only. Send the lookup to the master server of the named data. This guarantees that the most up to date information is seen at the possible expense that the master server may be busy.

-o Display the internal representation of the matching NIS+ object(s).

-P Follow concatenation path. Specify that the lookup should follow the concatenation path of a table if the initial search is unsuccessful.

-v Verbose. Do not suppress the output of binary data when displaying matching entries. Without this option, binary data is displayed as the string ***BINARY***.

RETURN VALUES

0 Successfully matches some entries.

1 Successfully searches the table and no matches are found.

2 An error condition occurs. An error message is also printed.

EXAMPLES

This example searches a table named **passwd** in the **org_dir** subdirectory of the **zotz.com.** domain. It returns the entry that has the username of **skippy**. In this example, all the work is done on the server.

```
nismatch name=skippy passwd.org_dir.zotz.com.
```

This example is similar to the one above except that it uses **nisgrep** to find all users in the table named **passwd** that are using either *ksh*(1) or *csh*(1).

```
nisgrep 'shell=[ck]sh' passwd.org_dir.zotz.com.
```

EXTERNAL INFLUENCES

Environment Variables

NIS_PATH If this variable is set and the NIS+ table name is not fully qualified, each directory specified will be searched until the table is found (see *nisdefaults*(1)).

DIAGNOSTICS

No memory
An attempt to allocate some memory for the search failed.

tablename **is not a table**
The object with the name *tablename* was not a table object.

Can't compile regular expression
The regular expression in *keypat* was malformed.

column not found: *colname*
The column named *colname* does not exist in the table named *tablename*.

AUTHOR

nismatch and **nisgrep** were developed by Sun Microsystems, Inc.

SEE ALSO

niscat(1), nisdefaults(1), nisls(1), nistbladm(1), nis_objects(3N).

NAME

nismkdir - create NIS+ directories

SYNOPSIS

nismkdir [**-D** *defaults*] [**-m** *hostname* | **-s** *hostname*] *dirname*

DESCRIPTION

The **nismkdir** command creates new NIS+ subdirectories within an existing domain. It can also be used to create replicated directories. Without options, this command will create a subdirectory with the same master and the replicas as its parent directory.

It is advisable to use *nisserver*(1M) to create an NIS+ domain which consists of the specified directory along with the **org_dir** and **groups_dir** subdirectories.

The two primary aspects that are controlled when making a directory are its access rights, and its degree of replication.

dirname is the fully qualified NIS+ name of the directory that has to be created.

Options

-D *defaults* Specify a different set of defaults to be used when creating new directories. The *defaults* string is a series of tokens separated by colons. These tokens represent the default values to be used for the generic object properties. All of the legal tokens are described below.

ttl=*time*
This token sets the default time to live for objects that are created by this command. The value *time* is specified in the format as defined by the *nischttl*(1) command. The default value is **12h** (12 hours).

owner=*ownername*
This token specifies that the NIS+ principal *ownername* should own the created object. The default for this value is the principal who is executing the command.

group=*groupname*
This token specifies that the group *groupname* should be the group owner for the object that is created. The default value is **NULL**.

access=*rights*
This token specifies the set of access rights that are to be granted for the given object. The value *rights* is specified in the format as defined by the *nischmod*(1) command. The default value is **----rmcdr---r---**.

-m *hostname* If the directory named by *dirname* does not exist, then a new directory that is *not* replicated is created with host *hostname* as its master server.

If the directory name by *dirname* does exist, then the host named by *hostname* is made its master server.

-s *hostname* Specify that the host *hostname* will be a replica for an existing directory named *dirname*.

RETURN VALUES

This command returns **0** if successful and **1** otherwise.

EXAMPLES

Create a new directory **bar** under the **foo.com.** domain that shares the same master and replicas as the **foo.com.** directory:

```
nismkdir bar.foo.com.
```

Create a new directory *bar.foo.com.* that is not replicated under the **foo.com.** domain:

```
nismkdir -m myhost.foo.com. bar.foo.com.
```

Add a replica server of the *bar.foo.com.* directory:

```
nismkdir -s replica.foo.com. bar.foo.com.
```

n

EXTERNAL INFLUENCES

Environment Variables

`NIS_DEFAULTS` This variable contains a defaults string that will override the NIS+ standard defaults. If the **`-D`** switch is used, those values will then override both the **`NIS_DEFAULTS`** variable and the standard defaults.

`NIS_PATH` If this variable is set and the NIS+ directory name is not fully qualified, each directory specified will be searched until the directory is found (see *nisdefaults*(1)).

AUTHOR

`nismkdir` was developed by Sun Microsystems, Inc.

SEE ALSO

nis+(1), nischmod(1), nisdefaults(1), nisls(1), nisrmdir(1), nisserver(1M).

NOTES

A host that serves an NIS+ directory *must be* an NIS+ client in a directory above the one it is serving. The exceptions to this rule are the root NIS+ servers which are both clients and servers of the same NIS+ directory.

When the host's default domain is different from the default domain on the client where the command is executed, the hostname supplied as an argument to the **`-s`** or **`-m`** options must be fully qualified.

NAME

nispasswd - change NIS+ password information

SYNOPSIS

nispasswd [**-ghs**] [**-D** *domainname*] [*username*]

nispasswd -a

nispasswd -D *domainname*] [**-d** [*username*]]

nispasswd [**-l**] [**-f**] [**-n** *min*] [**-x** *max*] [**-w** *warn*] [**-D** *domainname*] *username*

DESCRIPTION

nispasswd changes a password, gecos (finger) field (**-g**option), home directory (**-h**option), or login shell (**-s**option) associated with the *username* (invoker by default) in the NIS+ passwd table.

Additionally, the command can be used to view or modify aging information associated with the user specified if the invoker has the right NIS+ privileges.

nispasswd uses secure RPC to communicate with the NIS+ server, and therefore, never sends unencrypted passwords over the communication medium.

nispasswd does not read or modify the local password information stored in the **/etc/passwd** file.

When used to change a password, **nispasswd** prompts non-privileged users for their old password. It then prompts for the new password twice to forestall typing mistakes. When the old password is entered, **nispasswd** checks to see if it has aged sufficiently. If aging is insufficient, **nispasswd** terminates; see *getpwent*(3C).

The old password is used to decrypt the username's secret key. If the password does not decrypt the secret key, **nispasswd** prompts for the old secure-RPC password. It uses this password to decrypt the secret key. If this fails, it gives the user one more chance. The old password is also used to ensure that the new password differs from the old by at least three characters. Assuming aging is sufficient, a check is made to ensure that the new password meets construction requirements described below. When the new password is entered a second time, the two copies of the new password are compared. If the two copies are not identical, the cycle of prompting for the new password is repeated twice. The new password is used to re-encrypt the user's secret key. Hence, it also becomes their secure-RPC password.

Passwords must be constructed to meet the following requirements:

- Each password must have at least six characters. Only the first eight characters are significant.
- Each password must contain at least two alphabetic characters and at least one numeric or special character. In this case, "alphabetic" refers to all upper or lower case letters.
- Each password must differ from the user's login *username* and any reverse or circular shift of that login *username*. For comparison purposes, an upper case letter and its corresponding lower case letter are equivalent.
- New passwords must differ from the old by at least three characters. For comparison purposes, an upper case letter and its corresponding lower case letter are equivalent.

Network administrators, who own the NIS+ password table, may change any password attributes if they establish their credentials (see *keylogin*(1)) before invoking **nispasswd**. Hence, **nispasswd** does not prompt these privileged-users for the old password and they are not forced to comply with password aging and password construction requirements.

Any user may use the **-d** option to display password attributes for his or her own login name. The format of the display will be:

username status mm/dd/yy min max warn

or, if password aging information is not present,

username status

where

username	The login ID of the user.
status	The password status of *username*: "PS" stands for password exists or locked, "LK" stands for locked, and "NP" stands for no password.

n

mm/dd/yy	The date password was last changed for *username*. (Note that all password aging dates are determined using Greenwich Mean Time and, therefore, may differ by as much as a day in other time zones.)
min	The minimum number of days required between password changes for *username*.
max	The maximum number of days the password is valid for *username*.
warn	The number of days relative to *max* before the password expires that the *username* will be warned.

Options

-g Change the gecos (finger) information.

-h Change the home directory.

-s Change the login shell. By default, only the NIS+ administrator can change the login shell. User will be prompted for the new login shell.

-a Show the password attributes for all entries. This will show only the entries in the NIS+ passwd table in the local domain that the invoker is authorized to "read".

-d [*username*] Display password attributes for the caller or the user specified if the invoker has the right privileges.

-l Locks the password entry for *username*. Subsequently, *login*(1) would disallow logins with this NIS+ password entry.

-f Force the user to change password at the next login by expiring the password for *username*.

-n *min* Set minimum field for *username*. The *min* field contains the minimum number of days between password changes for *username*. If *min* is greater than *max*, the user may not change the password. Always use this option with the **-x** option, unless *max* is set to -1 (aging turned off). In that case, *min* need not be set.

-x *max* Set maximum field for *username*. The *max* field contains the number of days that the password is valid for *username*. The aging for *username* will be turned off immediately if *max* is set to -1. If it is set to 0, then the user is forced to change the password at the next login session and aging is turned off.

-w *warn* Set *warn* field for *username*. The *warn* field contains the number of days before the password expires that the user will be warned whenever he or she attempts to log in.

-D *domainname*
Consult the **passwd.org_dir** table in *domainname*. If this option is not specified, the default domainname returned by **nis_local_directory()** will be used. This domainname is the same as that returned by *domainname*(1).

RETURN VALUE

The **nispasswd** command exits with one of the following values:

0 SUCCESS.

1 Permission denied.

2 Invalid combination of options.

3 Unexpected failure. NIS+ passwd table unchanged.

4 NIS+ passwd table missing.

5 NIS+ is busy. Try again later.

6 Invalid argument to option.

7 Aging is disabled.

AUTHOR

nispasswd was developed by Sun Microsystems, Inc.

SEE ALSO

keylogin(1), login(1), nis+(1), nistbladm(1), passwd(1), domainname(1), getpwent(3C), nsswitch.conf(4), passwd(4).

NOTES

The login program, file access display programs (for example, '**ls -l**') and network programs that require user passwords (for example, *rlogin*(1), *ftp*(1), etc.) use the standard *getpwent*(3C) interface to get password information. These programs will get the NIS+ password information, which is modified by **nispasswd**, only if the **passwd:** entry in the **/etc/nsswitch.conf** file includes **nisplus**. See *nsswitch.conf*(4) for more details.

NAME

nisrm - remove NIS+ objects from the namespace

SYNOPSIS

nisrm [**-if**] *name* ...

DESCRIPTION

The **nisrm** command removes NIS+ objects named *name* from the NIS+ namespace.

This command will fail if the NIS+ master server is not running.

Options

-i Interactive mode. Like the system *rm*(1) command, the **nisrm** command will ask for confirmation prior to removing an object. If the name specified by *name* is a non-fully qualified name, this option is forced on. This prevents the removal of unexpected objects.

-f Force. The removal is attempted, and if it fails for permission reasons, an *nischmod*(1) is attempted and the removal retried. If the command fails, it fails silently.

EXAMPLES

Remove the objects *foo*, *bar*, and *baz* from the namespace:

```
nisrm foo bar baz
```

EXTERNAL INFLUENCES

Environment Variables

NIS_PATH If this variable is set and the NIS+ name is not fully qualified, each directory specified will be searched until the object is found (see *nisdefaults*(1)).

RETURN VALUE

nisrm returns **0** on success and **1** on failure.

AUTHOR

nisrm was developed by Sun Microsystems, Inc.

SEE ALSO

nis+(1), nischmod(1), nisdefaults(1), nisrmdir(1), nistbladm(1), rm(1).

NOTES

This command will not remove directories (see *nisrmdir*(1)) nor will it remove non-empty tables (see *nistbladm*(1)).

NAME

nisrmdir - remove NIS+ directories

SYNOPSIS

nisrmdir [**-if**] [**-s** *hostname*] *dirname*

DESCRIPTION

nisrmdir deletes existing NIS+ subdirectories. It can remove a directory outright, or simply remove replicas from serving a directory.

This command modifies the object that describes the directory *dirname*, and then notifies each replica to remove the directory named *dirname.* If the notification of any of the affected replicas fails, the directory object is returned to its original state unless the **-f** option is present.

This command will fail if the NIS+ master server is not running.

Options

-i Interactive mode. Like the system *rm*(1) command, the **nisrmdir** command will ask for confirmation prior to removing a directory. If the name specified by *dirname* is a non-fully qualified name, this option is forced on. This prevents the removal of unexpected directories.

-f Force the command to succeed even though it may not be able to contact the affected replicas. This option should be used when a replica is known to be down and will not be able to respond to the removal notification. When the replica is finally rebooted, it will read the updated directory object, note that it is no longer a replica for that directory, and stop responding to lookups on that directory. Cleanup of the files that held the now removed directory can be accomplished manually by removing the appropriate files in the **/var/nis** directory (see *nisfiles*(4) for more information).

-s *hostname*
Specify that the host *hostname* should be removed as a replica for the directory named *dirname*. If this option is not present, *all* replicas and the master server for a directory are removed and the directory is removed from the namespace.

n

RETURN VALUE

This command returns **0** if it is successful, and **1** otherwise.

EXAMPLES

Remove a directory **bar** under the **foo.com.** domain:

```
nisrmdir bar.foo.com.
```

Remove a replica that is serving directory **bar.foo.com.** :

```
nisrmdir -s replica.foo.com. bar.foo.com.
```

Force the removal of directory **bar.foo.com.** from the namespace:

```
nisrmdir -f bar.foo.com.
```

EXTERNAL INFLUENCES

Environment Variables

NIS_PATH If this variable is set and the NIS+ directory name is not fully qualified, each directory specified will be searched until the directory is found (see *nisdefaults*(1)).

AUTHOR

nisrmdir was developed by Sun Microsystems, Inc.

SEE ALSO

nis+(1), nisdefaults(1), nisrm(1), nisfiles(4).

NAME

nistbladm - NIS+ table administration command

SYNOPSIS

nistbladm -a | **-A** [**-D** *defaults*] *colname=value* ... *tablename*

nistbladm -a | **-A** [**-D** *defaults*] *indexedname*

nistbladm -c [**-D** *defaults*] [**-p** *path*] [**-s** *sep*] *type colname*=[*flags*][,*access*] ... *tablename*

nistbladm -d *tablename*

nistbladm -e | **-E** *colname=value* ... *indexedname*

nistbladm -m *colname=value* ... *indexedname*

nistbladm -r | **-R** [*colname=value* ...] *tablename*

nistbladm -r | **-R** *indexedname*

nistbladm -u [**-p** *path*] [**-s** *sep*] [**-t** *type*] [*colname*=*access* ...] *tablename*

DESCRIPTION

The **nistbladm** command is used to administer NIS+ tables. There are five primary operations that it performs: creating and deleting tables, adding entries to, modifying entries within, and removing entries from tables.

Though NIS+ does not place restrictions on the size of tables or entries, the size of data has an impact on the performance and the disk space requirements of the NIS+ server. NIS+ is not designed to store huge pieces of data, such as files; instead pointers to files should be stored in NIS+.

NIS+ design is optimized to support 10,000 objects with a total size of 10M bytes. If the requirements exceed the above, it is suggested that a domain hierarchy be created, or the data stored in the tables be pointers to the actual data, instead of the data itself.

n

When creating tables, a table type, *type*, and a list of column definitions must be provided.

type is a string that is stored in the table and later used by the service to verify that entries being added to it are of the correct type.

Syntax for column definitions is:

colname=[*flags*][,*access*]

flags is a combination of:

S	Searchable. Specifies that searches can be done on the column's values (see *nismatch*(1)).
I	Case-insensitive (only makes sense in combination with **S**). Specifies that searches should ignore case.
C	Crypt. Specifies that the column's values should be encrypted.
B	Binary data (does not make sense in combination with **S**). If not set, the column's values are expected to be null terminated ASCII strings.
X	XDR encoded data (only makes sense in combination with **B**).

access is specified in the format as defined by the *nischmod*(1) command.

When manipulating entries, this command takes two forms of entry name. The first uses a series of space separated *colname=value* pairs that specify column values in the entry. The second is an NIS+ indexed name, *indexedname*, of the form:

[*colname*=*value*, ... **],***tablename*

Options

-a | A Add entries to a NIS+ table. The difference between the lowercase 'a' and the uppercase 'A' is in the treatment of preexisting entries. The entry's contents are specified by the *column=value* pairs on the command line. Note: Values for *all* columns must be specified when adding entries to a table.

Normally, NIS+ reports an error if an attempt is made to add an entry to a table that would overwrite an entry that already exists. This prevents multiple parties from adding duplicate entries and having one of them get overwritten. If you wish to force the add, the uppercase 'A' specifies that the entry is to be added, even if it already exists. This is analogous to a modify operation on the entry.

-c Create a table named *tablename* in the namespace. The table that is created must have at least one column and at least one column must be searchable.

-d *tablename* Destroy the table named *tablename*. The table that is being destroyed must be empty. The table's contents can be deleted with the **-R** option below.

-e|E Edit the entry in the table that is specified by *indexdname*. *indexdname* must uniquely identify a single entry. It is possible to edit the value in a column that would change the indexed name of an entry.

The change (*colname*=*value*) may affect other entries in the table if the change results in an entry whose indexed name is different from *indexedname* and which matches that of another existing entry. In this case, the **-e** option will fail and an error will be reported. The **-E** option will force the replacement of the existing entry by the new entry (effectively removing two old entries and adding a new one).

-m Modify an entry in the table that is specified by *indexedname*. Note: Since it is possible to modify the value in a column that would change the indexed name for an entry, both the column value pair and the indexed name are required. It uses the indexed name to look up the entry, modify it, and write it back with the new value. The indexed name must uniquely identify a single entry.

-r|R Remove entries from a table. The entry is specified by either a series of *column*=*value* pairs on the command line, or an indexed name that is specified as *entryname*. The difference between the interpretation of the lowercase **r** versus the uppercase **R** is in the treatment of non-unique entry specifications. Normally the NIS+ server will disallow an attempt to remove an entry when the search criterion specified for that entry resolves to more than one entry in the table. However, it is sometimes desirable to remove more than one entry, as when you are attempting to remove all of the entries from a table. In this case, using the uppercase **R** will force the NIS+ server to remove all entries matching the passed search criterion. If that criterion is null and no column values specified, then all entries in the table will be removed.

n

-u Update attributes of a table. This allows the concatenation path (**-p**), separation character (specified with the (**-s**)), column access rights, and table type string (**-t**) of a table to be changed. Neither the number of columns, nor the columns that are searchable may be changed.

-D *defaults* When creating objects, this option specifies a different set of defaults to be used during this operation. The *defaults* string is a series of tokens separated by colons. These tokens represent the default values to be used for the generic object properties. All of the legal tokens are described below.

ttl=*time*
This token sets the default time to live for objects that are created by this command. The value *time* is specified in the format as defined by the *nischttl*(1) command. The default value is 12 hours.

owner=*ownername*
This token specifies that the NIS+ principal *ownername* should own the created object. Normally this value is the same as the principal who is executing the command.

group=*groupname*
This token specifies that the group *groupname* should be the group owner for the object that is created. The default value is NULL.

access=*rights*
This token specifies the set of access rights to be granted for the given object. The value *rights* is specified in the format as defined by the *nischmod*(1) command. The default value is **----rmcdr---r---**.

-p *path* When creating or updating a table, this option specifies the table's search path. When an **nis_list**() function is invoked, the user can specify the flag FOLLOW_PATH to tell the client library to continue searching tables in the table's path if the search criteria used does not yield any entries. The path consists of an ordered list of table names, separated by colons. The names in the path must be fully qualified.

-s *sep* When creating or updating a table, this option specifies the table's separator character. The separator character is used by *niscat*(1) when displaying tables on the standard output. Its purpose is to separate column data when the table is in ASCII form. The default value is a space.

-t *type* When updating a table, this option specifies the table's type string.

RETURN VALUE

This example returns **0** on success and **1** on failure.

EXAMPLES

Create a table named **hobbies** in the directory **foo.com.** of the type **hobby_tbl** with two searchable columns, **name** and **hobby**:

```
nistbladm -c hobby_tbl name=S,a+r,o+m hobby=S,a+r
hobbies.foo.com.
```

The column **name** has read access for all (that is, **owner**, **group**, and **world**) and modify access for only the owner. The column **hobby** is readable by all, but not modifiable by anyone.

In this example, if the access rights had not been specified, the tables access rights would have come from either the standard defaults or the **NIS_DEFAULTS** variable (see below).

Add entries to this table:

```
nistbladm -a name=bob hobby=skiing hobbies.foo.com.
nistbladm -a name=sue hobby=skiing hobbies.foo.com.
nistbladm -a name=ted hobby=swimming hobbies.foo.com.
```

Add the concatenation path:

```
nistbladm -u -p hobbies.bar.com.:hobbies.baz.com. hobbies
```

Delete the skiers from our list:

```
nistbladm -R hobby=skiing hobbies.foo.com.
```

n

Note: The use of the **-r** option would fail because there are two entries with the value of **skiing**.

To create a table with a column that is named with no flags set, you supply only the name and the equal sign (=) as follows:

```
nistbladm -c notes_tbl name=S,a+r,o+m note=  notes.foo.com.
```

This example created a table, named *notes.foo.com.,of*type *notes_tbl* with two columns **name** and **note**. The **note** column is not searchable.

When entering data for columns in the form of a *value* string, it is essential that terminal characters be protected by single or double quotes. These are the characters equals (=), comma (,), left bracket ([), right bracket (]), and space ( ). These characters are parsed by NIS+ within an indexed name. These characters are protected by enclosing the entire value in double quote (") characters as follows:

```
nistbladm -a fullname="Joe User" nickname=Joe nicknames
```

If there is any doubt about how the string will be parsed, it is better to enclose it in quotes.

WARNINGS

To modify one of the entries, say, for example, from **bob** to **robert**:

```
nistbladm -m name=robert [name=bob],hobbies
```

Note that **[name=bob],hobbies** is an indexed name, and that the characters '[' (open bracket) and ']' (close bracket) are interpreted by the shell. When typing entry names in the form of NIS+ indexed names, the name must be protected by using single quotes.

It is possible to specify a set of defaults such that you cannot read or modify the table object later.

EXTERNAL INFLUENCES

Environment Variables

NIS_DEFAULTS This variable contains a defaults string that will override the NIS+ standard defaults. If the **-D** switch is used, those values will then override both the **NIS_DEFAULTS**

variable and the standard defaults.

NIS_PATH If this variable is set and the NIS+ table name is not fully qualified, each directory specified will be searched until the table is found (see *nisdefaults*(1)).

AUTHOR

nistbladm was developed by Sun Microsystems, Inc.

SEE ALSO

nis+(1), niscat(1), nischmod(1), nischown(1), nisdefaults(1), nismatch(1), nissetup(1M).

NAME

nistest - return the state of the NIS+ namespace using a conditional expression

SYNOPSIS

nistest [**-ALMP**] [**-a** *rights* | **-t** *type*] *object*

nistest [**-ALMP**] [**-a** *rights*] *indexedname*

DESCRIPTION

nistest provides a way for shell scripts and other programs to test for the existence, type, and access rights of objects and entries. Entries are named using indexed names (see *nismatch*(1)).

Options

-A All data. This option specifies that the data within the table and all of the data in tables in the initial table's concatenation path be returned. This option is only valid when using indexed names or following links.

-L Follow links. If the object named by *object* or the tablename component of *indexedname* names a LINK type object, the link is followed when this switch is present.

-M Master server only. This option specifies that the lookup should be sent to the master server of the named data. This guarantees that the most up-to-date information is seen at the possible expense that the master server may be busy.

-P Follow concatenation path. This option specifies that the lookup should follow the concatenation path of a table if the initial search is unsuccessful. This option is only valid when using indexed names or following links.

-a *rights* This option is used to verify that the current process has the desired or required access rights on the named object or entries. The access rights are specified in the same way as the **nischmod** command.

-t *type* This option tests the type of *object*. The value of *type* can be one of the following:

G	Return true if the object is a group object.
D	Return true if the object is a directory object.
T	Return true if the object is a table object.
L	Return true if the object is a link object.
P	Return true if the object is a private object.

RETURN VALUE

0	Success.
1	Failure due to object not present, not of specified type and/or no such access.
2	Failure due to illegal usage.

EXAMPLES

When testing for access rights, **nistest** returns success (0) if the specified rights are granted to the current user. Thus, testing for access rights

```
nistest -a w=mr skippy.domain
```

Tests that all authenticated NIS+ clients have read and modify access to the object named *skippy.domain*.

Testing for access on a particular entry in a table can be accomplished using the indexed name syntax. The following example tests to see if an entry in the password table can be modified.

```
nistest -a o=m '[uid=99],passwd.org_dir'
```

EXTERNAL INFLUENCES

Environment Variables

NIS_PATH If this variable is set and the NIS+ name is not fully qualified, each directory specified will be searched until the object is found (see *nisdefaults*(1)).

AUTHOR

`nistest` was developed by Sun Microsystems, Inc.

SEE ALSO

nis+(1), nischmod(1), nisdefaults(1).

NAME

nl - line numbering filter

SYNOPSIS

nl [**-b***type*] [**-h***type*] [**-f***type*] [**-p**] [**-v***start#*] [**-i***incr*] [**-s***sep*] [**-w***width*] [**-n***format*] [**-l***num*] [**-d***delim*] [*file*]

DESCRIPTION

nl reads lines from the named *file* or the standard input if no *file* is named and reproduces the lines on the standard output. Lines are numbered on the left in accordance with the command options in effect.

nl views the text it reads in terms of logical pages. Line numbering is reset at the start of each logical page. A logical page consists of a header, a body, and a footer section. Empty sections are valid. Different line numbering options are independently available for header, body, and footer (e.g., no numbering of header and footer lines while numbering blank lines only in the body).

The start of logical page sections are signaled by input lines containing nothing but the following delimiter character(s):

Line contents	Start of
\:\:\:	header
\:\:	body
\:	footer

Unless told otherwise, **nl** assumes the text being read is in a single logical page body.

Command options can appear in any order and can be intermingled with an optional file name. Only one file can be named. **nl** recognizes the following options:

-b*type* Specifies which logical page body lines are to be numbered. Recognized *types* and their meanings are:

a	number all lines;
t	number lines with printable text only;
n	no line numbering;
p*string*	number only lines that contain the regular expression specified in *string*. Basic Regular Expression syntax is supported (see *regexp*(5)).

The default *type* for logical page body is **t** (text lines numbered).

-h*type* Same as **-b***type* except for header. Default *type* for logical page header is **n** (no lines numbered).

-f*type* Same as **-b***type* except for footer. Default for logical page footer is **n** (no lines numbered).

-p Do not restart numbering at logical page delimiters.

-v*start#* *start#* is the initial value used to number logical page lines. Default is **1**.

-i*incr* *incr* is the increment value used to number logical page lines. Default is **1**.

-s*sep* *sep* is the character or characters used in separating the line number and the corresponding text line. Default *sep* is a tab.

-w*width* *width* is the number of character columns to be used for the line number. Default *width* is **6**.

-n*format* *format* is the line numbering format. Recognized values are:

ln	left justified, leading zeroes suppressed;
rn	right justified, leading zeroes suppressed;
rz	right justified, leading zeroes kept.

Default *format* is **rn** (right justified).

-l*num* *num* is the number of consecutive blank lines to be treated and numbered as a single line. For example, **-l3** results in every third adjacent blank line being numbered if the appropriate **-ha**, **-ba**, and/or **-fa** option is set. Default is **1**.

-d*xx* The delimiter characters specifying the start of a logical page section can be changed from the default characters (**\:**) to two user-specified characters. If only one character is entered, the second character remains the default character (**:**). No space should

n

appear between the **-d** and the delimiter characters, however, this restriction is not there for **XPG4** compliant **nl**. To define a backslash as the delimiter, use two backslashes.

EXTERNAL INFLUENCES

Environment Variables

LC_COLLATE determines the collating sequence used in evaluating regular expressions.

LC_CTYPE determines the characters matched by character class expressions in regular expressions.

If **LC_COLLATE** or **LC_CTYPE** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**.

If any internationalization variable contains an invalid setting, **nl** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single-byte character code sets are supported.

EXAMPLES

Number **file1** starting at line number 10, using an increment of ten. The logical page delimiters are **!** and **+**:

```
nl -v10 -i10 -d!+ file1
```

SEE ALSO

pr(1), environ(5), lang(5), regexp(5).

STANDARDS CONFORMANCE

nl: SVID2, SVID3, XPG2, XPG3, XPG4

NAME

nljust - justify lines, left or right, for printing

SYNOPSIS

nljust [**-acilnt**] [**-d** *digits*] [**-e** *seq*] [**-j** *just*] [**-m** *mode*] [**-o** *order*] [**-r** *margin*] [**-w** *width*] [**-x** *ck*] [*file* ...]

DESCRIPTION

nljust formats for printing data written in languages with a right-to-left orientation. It is designed to be used with the **pr** and the **lp** commands (see *pr*(1) and *lp*(1)).

nljust reads the concatenation of input files (or standard input if none are given) and produces on standard output a right-to-left formatted version of its input. If **-** appears as an input file name, **nljust** reads standard input at that point. Use **--** to delimit the end of options.

nljust formats input files for all languages that are read from right to left. For languages that have a left-to-right orientation, the command merely copies input files to standard output.

Options

nljust recognizes the following options:

-a Justify data for all languages, including those having a left-to-right text orientation. By default only right-to-left language data is justified. For all other languages, input files are directly copied to standard output.

-c Select enhanced printer shapes for some Arabic characters. With this option, two-character combinations of laam and alif are replaced by a single character.

-i Triggers ISO 8859-6 interpretation of the data.

-d *digits* Processes digits for output as hindi, western, or both. *digits* can be **h**, **w**, or both.

-e *seq* Use *seq* as the escape sequence to select the primary character set. This escape sequence is used by languages that have too many characters to be accommodated by ASCII in a single 256-character set. In these cases, the *seq* escape sequence can be used to select the non-ASCII character set. The *escape* character itself (0x1b) is not given on the command line. Hewlett-Packard escape sequences are used by default.

-j *just* If *just* is **l**, left justify print lines. If *just* is **r**, right-justify print lines starting from the (designated or default) print width column. The default is right justification.

-l Replace leading spaces with alternative spaces. Some right-to-left character sets have a non-ASCII or alternative space. This option can be useful when filtering **pr -n** output (see *pr*(1)). With right justification, the **-l** option causes line numbers to be placed immediately to the right of the tab character. Without the **-l** option, right justification causes line numbers to be placed at the print-width column. By default, leading spaces are not replaced by alternative spaces.

-m *mode* Indicate *mode* of any file to be formatted. Mode refers to the text orientation of the file when it was created (see *hpnls*(5) for more details). If *mode* is **l**, assume Latin mode. If *mode* is **n**, assume non-Latin mode. By default, mode information is obtained from the **LANGOPTS** environment variable.

-n Do not terminate lines containing printable characters with a new-line. By default, print lines are terminated by new-lines.

-o *order* Indicate data *order* of any file to be formatted. The text orientation of a file can affect the way its data is arranged (see *hpnls*(5) for more details). If *order* is **k**, assume keyboard order. If *order* is **s**, assume screen order. By default, order information is obtained from the **LANGOPTS** environment variable.

-t Truncate print lines that do not fit the designated or default line length. Print lines are folded (that is, wrapped to next line) by default.

-x *ck* Expand input tabs to column positions *k*+1, *2**k*+1, 3**k*+1,* etc. Tab characters in the input are expanded to the appropriate number of spaces. If *k* is 0 or is omitted, default tab settings at every eighth position is assumed. If *cd* (any non-digit character) is given, it is treated as the input tab character. The default for *c* is the tab character. **nljust** always expands input tabs. This option provides a way to change the tab character and setting. If this option is specified, at least one of the parameters *c* or *k* must be given.

-r *margin* Designate a number as the print *margin*. The print margin is the column where truncation or folding takes place. The print margin determines how many characters appear on a single line and can never exceed the print width. The print margin is relative to the justification. If the print margin is 80, folding or truncation occurs at column 80 starting from the right during a right justification. Similarly, folding or truncation occurs at column 80 starting from the left during a left justification. By default, the print margin is set to column 80.

-w *width* Designates a number as the print *width*. The print width is the maximum number of columns in the print line. Print width determines the start of text during a right justification. The larger the print width, the further to the right the text will start. By default, an 80-column print width is used.

EXTERNAL INFLUENCES

Environment Variables

The **LANGOPTS** environment variable determines the mode and order of the file. The syntax of **LANGOPTS** is [*mode*][_*order*]. *mode* describes the mode of a file where **l** represents Latin mode and **n** represents non-Latin mode. Non-Latin mode is assumed for values other than **l** and **n**. *order* describes the data order of a file where **k** is keyboard and **s** is screen. Keyboard order is assumed for values other than **k** and **s**. Mode and order information in **LANGOPTS** can be overridden from the command line.

The **LC_ALL** environment variable determines the direction of a language (left-to-right or right-to-left) and whether context analysis of characters is necessary.

The **LC_NUMERIC** environment variable determines whether a language has alternative numbers.

The **LANG** environment variable determines the language in which messages are displayed.

International Code Set Support

Single-byte character code sets are supported.

EXAMPLES

Right justify **file1** on a 132-column printer with a print margin at column 80 (the default):

```
nljust -w 132 file1 | lp
```

Right justify **pr** output of **file2** with line numbers on a 132-column printer with a print margin at column 132:

```
pr -n file2 | nljust -w 132 -r 132 | lp
```

WARNINGS

If **pr** with line numbers (**-n** option) is piped to **nljust**, the separator character must be a tab (0x09).

It is the user's responsibility to ensure that the **LANGOPTS** environment variable accurately reflects the status of the file.

Mode and justification must be consistent. Only non-Latin-mode files can be right justified in a meaningful way. Similarly, only Latin-mode files can be safely left justified. If mode and justification do not match, the results are undefined.

If present, alternative numbers always have a left-to-right orientation.

The **nljust** command is HP proprietary, not portable to other vendors' systems, and will not be provided in future HP-UX releases.

AUTHOR

nljust was developed by HP.

SEE ALSO

forder(1), lp(1), pr(1), strord(3C), hpnls(5).

NAME

nm - print name list of common object file

SYNOPSIS

/usr/ccs/bin/nm [**-ACefghlnNqrsTuUvV**] [**-d**|**-o**|**-x**] [**-p**|**-P**] [**-t** *format*] *file* ...

DESCRIPTION

The **nm** command displays the symbol table of each object file, *file*.

file can be a relocatable object file or an executable object file, or it can be an archive of relocatable or executable object files.

There are three general output formats: the default (neither **-p** nor **-P** specified), **-p** specified, and **-P** specified. The output formats are described after the "Options" subsection.

Options

nm recognizes the following options:

-A	Prefix each output line with the name of the object file or archive, *file*. Equivalent to **-r**.
-C	Demangle C++ names before printing them (ELF only).
-d	Display the *value* and *size* of a symbol in decimal. This is the default for the default format or the **-p** format. Equivalent to **-t d**.
-e	Display only **external** and **static** symbols. This option is ignored (see **-f**).
-f	Display full output. This option is in force by default.
-g	Display only **external** (global) symbol information.
-h	Do not display the output header data.
-l	Distinguish between weak and global symbols by appending * to the key letter of weak symbols. Only takes effect with **-p** and/or **-P.**
-n	Sort symbols by *name*, in ascending collation order, before they are printed. This is the default. See "Environment Variables" in EXTERNAL INFLUENCES below.
-N	Display symbols in the order in which they appear in the symbol table.
-o	Display the *value* and *size* of a symbol in octal. Equivalent to **-t o**.
-p	Display information in a blank-separated output format. Each symbol *name* is preceded by its value (blanks if undefined) and one of the letters **A** (absolute), **B** (bss symbol), **C** (common symbol), **D** (data symbol), **R** (section region), **S** (tstorage symbol), **T** (text symbol) or **U** (undefined). **S** is used on SOM files only. If the symbol is local (nonexternal), the type letter is in lowercase. If the symbol is a secondary definition, the type letter is followed by the letter **S**. Note that **-p** is not compatible with **-P**.
-P	Display information in a portable output format, as specified below, to standard output. Note that **-P** is not compatible with **-p**.
-q	Silence some warning messages (SOM only).
-r	Prefix each output line with the name of the object file or archive, *file*. Equivalent to **-A**.
-s	Print the section index instead of the section name (ELF only).
-t *format*	Display each numeric value in the specified format. *format* can be one of: **d** Display the *value* and *size* of a symbol in decimal. This is the default for the default format or the **-p** format. Equivalent to **-d**. **o** Display the *value* and *size* of a symbol in octal. Equivalent to **-o**. **x** Display the *value* and *size* of a symbol in hexadecimal. This is the default for the **-P** format. Equivalent to **-x**.
-T	Truncate every name that would otherwise overflow its column and place an asterisk as the last character in the displayed name to mark it as truncated (SOM only). If **-A** or **-r** is also specified, the *file* prefix is truncated first.

By default, **nm** prints the entire name of the symbols listed. Since object files can have symbol names with an arbitrary number of characters, a name that is longer than the width of the column set aside for names will overflow its column, forcing every column after the name to be misaligned.

-u	Display undefined symbols only.
-U	Print the usage menu. **-v** Sort symbols by *value* before they are printed.
-V	Display the executing version of the **nm** command on standard error.
-x	Display the *value* and *size* of a symbol in hexadecimal. This is the default for the **-P** format. Equivalent to **-t x**.

Default Output Format - 32 bit

If the default (neither the **-p** nor the **-P** option) output format is specified, each symbol has the following columns, separated by vertical bars (**|**). The default for numbers is decimal (**-d** or **-t d**).

If decimal:

"%20s|%10d|%6s|%7s|%s", *name*, *value*, *scope*, *type*, *subspace*

If octal:

"%20s|%012o|%6s|%7s|%s", *name*, *value*, *scope*, *type*, *subspace*

If hexadecimal:

"%20s|0x%08x|%6s|%7s|%s", *name*, *value*, *scope*, *type*, *subspace*

Default Output Format - 64 bit

If the default (neither the **-p** nor the **-P** option) output format is specified, each symbol has the following columns, separated by vertical bars (**|**). The default for numbers is decimal (**-d** or **-t d**).

If decimal:

"[%u]%s|%22llu|%8u|%s|%s|%1d|%s|%s",
index, *value*, *size*, *type*, *bind*, *O*, *shndx*, *name*

If octal:

"[%u]%s|%022llo|%010o|%s|%s|%1o|%s|%s",
index, *value*, *size*, *type*, *bind*, *O*, *shndx*, *name*

If hexadecimal:

"[%u]%s|0x%016llx|0x%08x|%s|%s|%1x|%s|%s",
index, *value*, *size*, *type*, *bind*, *O*, *shndx*, *name*

The descriptions are explained below:

name	The name of the symbol.
value	Its value expressed as an offset or an address depending on its storage class.
scope	The scope of the symbol (**external**, **sdef**, **static**, or **undefined**). The **sdef** scope indicates an external symbol that is flagged as a secondary definition.
type	The type of the symbol (**absolute**, **arg_ext**, **code**, **data**, **entry**, **milli_ext**, **millicode**, **module**, **null**, **oct_dis**, **plabel**, **pri_prog**, **sec_prog**, **storage**, **stub**, **sym_ext**, **tstor**).
subspace	The subspace to which the symbol belongs.
bind	Specifies the symbol binding type (local, weak, global).
O	This field is used for files that have large section tables (>65K sections). For smaller files, the value of this field is 0.
Shndx	Identifies the index of the section that the symbol belongs to.

Identifies the index of the symbol in the symbol table.

Output Format for −p

If the **-p** option is specified, information is displayed using the following portable C-language formats. The default for numbers is decimal (**-d** or **-t d**).

If decimal: **"%010d %s %s",** *value***,** *type***,** *name*

If octal: **"%012o %s %s",** *value***,** *type***,** *name*

If hexadecimal: **"0x%08x %s %s",** *value***,** *type***,** *name*

If **-A** or **-r**, the line is preceded by: **"%20s:",** *file*

Output Format for −P

If the **-P** option is specified, information is displayed using the following portable C-language formats. The default for numbers is hexadecimal (**-x** or **-t x**). In the format string, **%s** represents string output; **%d** represents decimal output; **%o** represents octal output; **%x** represents hexadecimal output; **\n** represents newline; all other characters represent themselves.

- If decimal is specified:

  "%s %s %d %d\n", *library-object***,** *name***,** *type***,** *value***,** *size*

- If octal is specified:

  "%s%s %s %o %o\n", *library-object* **,** *name***,** *type***,** *value***,** *size*

- If hexadecimal is specified, or by default:

  "%s%s %s %x %x\n", *library-object***,** *name***,** *type***,** *value***,** *size*

where *library-object* is a string preformatted as follows:

- If **-A** and **-r** are not specified, *library-object* is an empty string.
- If **-A** or **-r** is specified, and the corresponding *file* operand does not name a library:

  "%s: ", *file*

- If **-A** or **-r** is specified and the corresponding *file* operand names a library, *object-file* names the object file in the library containing the symbol being described:

  "%s[%s]: ", *file***,** *object-file*

If **-A** and **-r** are not specified, and if more than one *file* operand is specified, or if a single *file* operand that names a library is specified, then **nm** prints a line identifying the object containing the symbols before the lines containing those symbols, in one of the following forms:

- If the corresponding *file* operand does not name a library:

  "%s:\n", *file*

- If the corresponding *file* operand names a library, *object-file* names the object file in the library containing the symbol being described:

  "%s[%s]:\n", *file***,** *object-file*

EXTERNAL INFLUENCES

Environment Variables

The following internationalization variables affect the execution of **nm**:

LANG determines the locale category for native language, local customs and coded character set in the absence of **LC_ALL** or other **LC_*** environment variables. If **LANG** is not specified or is null, it defaults to **C** (see *lang*(5)).

LC_ALL, if set to a nonempty string value, determines the values for all locale categories and has precedence over **LANG** and other **LC_*** environment variables.

LC_COLLATE determines the locale category for character collation.

LC_CTYPE determines the locale category for character handling functions.

LC_MESSAGES determines the locale that should be used to affect the format and contents of diagnostic messages written to standard error.

LC_NUMERIC determines the locale category for numeric formatting.

ST_NMCAT and **NLSPATH** determine the location of message catalogues for processing **LC_MESSAGES**.

If an internationalization variable is not specified or is null, it defaults to the value of **LANG**.

If **LANG** is not specified or is null, it defaults to **C** (see *lang*(5)).

If any internationalization variable contains an invalid setting, then all internationalization variables default to **C** (see *environ*(5)).

International Code Set Support

Single-byte character code sets are supported.

EXAMPLES

Display which object files have undefined references for the symbol **leap**:

```
nm -rup *.o | grep leap
```

Display which object files have a definition for the text symbol leap:

```
nm -rp *.o | awk '{ if ($3 == "T" && $4 == "leap") { print $0 } }'
```

WARNINGS

By default, **nm** now sorts symbols by name (the **-n** option). To turn off sorting, use the **-N** option.

Some options added for standards conformance duplicate the functionality of options that previously existed. This duplication has been retained for backward compatibility.

SEE ALSO

System Tools

cc_bundled(1)	HP-UX C compiler
ld(1)	Link editor

Miscellaneous

crt0(3)	Execution startup routine
end(3C)	Symbol of the last locations in program

STANDARDS CONFORMANCE

nm: SVID2, SVID3, XPG2, XPG3, XPG4

NAME

nohup - run a command immune to hangups

SYNOPSIS

nohup *command* [*arguments*]

DESCRIPTION

nohup executes *command* with hangups and quits ignored. If output is not redirected by the user, both standard output and standard error are sent to **nohup.out**. If **nohup.out** is not writable in the current directory, output is redirected to **$HOME/nohup.out**; otherwise, **nohup** fails. If a file is created, the file's permission bits will be set to **S_IRUSR | S_IWUSR**.

If output from **nohup** is redirected to a terminal, or is not redirected at all, the output is sent to **nohup.out**.

EXTERNAL INFLUENCES

Environment Variables

LC_MESSAGES determines the language in which messages are displayed.

If **LC_MESSAGES** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**.

If any internationalization variable contains an invalid setting, **nohup** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported.

EXAMPLES

It is frequently desirable to apply **nohup** to pipelines or lists of commands. This can be done only by placing pipelines and command lists in a single file, called a shell script. To run the script using **nohup**:

```
nohup sh file
```

nohup features apply to the entire contents of *file*. If the shell script *file* is to be executed often, the need to type **sh** can be eliminated by setting execute permission on *file*. The script can also be run in the background with interrupts ignored (see *sh*(1)):

```
nohup file &
```

file typically contains normal keyboard command sequences that one would want to continue running in case the terminal disconnects, such as:

```
tbl ofile | eqn | nroff > nfile
```

WARNINGS

Be careful to place punctuation properly. For example, in the command form:

```
nohup command1; command2
```

nohup applies only to *command1*. To correct the problem, use the command form:

```
nohup (command1; command2)
```

Be careful of where standard error is redirected. The following command may put error messages on tape, making it unreadable:

```
nohup cpio -o <list >/dev/rmt/c0t0d0BEST&
```

whereas

```
nohup cpio -o <list >/dev/rmt/c0t0d0BEST 2>errors&
```

puts the error messages into file **errors**.

EXIT STATUS

The following exit values are returned:

126 The command specified by *command* was found but could not be invoked

n

127 An error occurred in the nohup utility or the specified *command* could not be found

Otherwise, the exit status of nohup will be that of the command specified.

SEE ALSO

chmod(1), nice(1), sh(1), signal(5).

STANDARDS CONFORMANCE

nohup: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

NAME

nroff - format text

SYNOPSIS

nroff [*options*] *file* ...

DESCRIPTION

nroff is a text formatting program that interprets source text contained in *file* and prepares it for printing on typewriter-like devices and line printers. If *file* name is **-** or not specified, standard input is used as source text.

If the file contains plain text with no formatter requests, **nroff** uses default line lengths and page dimensions to produce readable output, outputting a blank line for each blank line encountered in the input, and filling and adjusting text to both margins. **nroff** ignores any lines in the source text that begin with a period (**.**) but are not valid **nroff** formatter requests.

nroff formatting capabilities are described in the tutorial cited below.

Source File Preparation

Document source file preparation is usually easier when text is coded using macro packages such as *mm*(1) which provide a high-level interface for headings, page footers, lists, and other features, rather than coding the file with inherently low-level **nroff** requests.

Options

nroff recognizes the following command-line *options*, which can appear in any order but must appear before the *file* argument:

-o*list* Print only pages whose page numbers appear in the *list* of numbers and ranges, separated by commas. A range *n***-***m* means pages *n* through *m*; an initial **-***n* means from the beginning to page *n*; and a final *n***-** means from *n* to the end. (See WARNINGS below.)

-n*n* Number first generated page *n*.

-s*n* Stop every *n* pages. **nroff** halts *after* every *n* pages (default *n*=1) to allow paper loading or changing, and resumes upon receipt of a line-feed or new-line (new-lines do not work in pipelines, such as with **mm**). When **nroff** halts between pages, an ASCII **BEL** is sent to the terminal.

-r*aN* Set register *a* (which must have a one-character name) to *N*.

-i Read standard input after *files* are exhausted.

-q Invoke the simultaneous input-output mode of the **.rd** request.

-z Print only messages generated by **.tm** (terminal message) requests.

-m*name* Precede the input *files* with the non-compiled (ASCII text) macro file

> **/usr/lib/nls/***LANG***/tmac/tmac.***name*

where *LANG* is the value of the **LANG** environment variable. If **LANG** is not set or

> **/usr/lib/nls/***LANG***/tmac/tmac.***name*

does not exist, the following file is used instead:

> **/usr/share/lib/tmac/tmac.***name*

-T*name* Prepare output for specified terminal. Known *name*s are as follows:

37	for the (default) TELETYPE Model 37 terminal
tn300	for the GE TermiNet 300 (or any terminal without half-line capability)
300s	for the DASI 300s
300	for the DASI 300
450	for the DASI 450
lp	for a (generic) ASCII line printer
382	for the DTC-382
4000A	for the Trendata 4000A
832	for the Anderson Jacobson 832

X	for a (generic) EBCDIC printer
2631	for the Hewlett-Packard 2631 line printer
klp	for a (generic) 16-bit character printer having ratio of 2 to 3 in 8-bit and 16-bit character width
lj	for Hewlett-Packard PCL3 and newer laser printers.

-e Produce equally-spaced words in adjusted lines, using the full resolution of the particular terminal.

-h Use output tabs during horizontal spacing to speed output and reduce output character count. Tab settings are assumed to be every eight nominal character widths.

-u*n* Set the emboldening factor (number of character overstrikes) for the third font position (bold) to *n*, or to zero if *n* is missing.

-P If this option is specified on the command line, it allows the use of the special feature provided by some Asian printers which prints two column wide characters in 3/2 column wide boxes.

EXTERNAL INFLUENCES

Environment Variables

LC_CTYPE determines the interpretation of text as single and/or multi-byte characters.

LANG is used to determine the search path for the **-m** option. **LANG** also determines the language in which messages are displayed.

If **LC_CTYPE** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**. If any internationalization variable contains an invalid setting, **nroff** behaves as if all internationalization variables are set to "C". See *environ*(5).

EXAMPLES

The following command prints the first five pages of the document whose **nroff** source file is *filename*:

nroff -o-5 *filename*

Note that there should not be a space between the **o** and the **-** or the **-** and the **5**.

To print only pages 1, 3, and 4 type:

nroff -o1,3,4 *filename*

WARNINGS

When **nroff** is used with the **-o***list* option inside a pipeline, it may cause a harmless "broken pipe" diagnostic if the last page of the document is not specified in *list*.

FILES

/usr/share/lib/macros/*	Standard macro files
/usr/share/lib/term/*	Terminal driving tables for **nroff**
/usr/share/lib/suftab	Suffix hyphenation tables
/usr/share/lib/tmac/tmac.*	Standard macro files and pointers

SEE ALSO

col(1), mm(1), neqn(1), soelim(1), ul(1), man(5).

nroff/troff tutorial in the *Text Formatting Users Guide*.

NAME

nslookup - query name servers interactively

SYNOPSIS

nslookup [-*option* ...] *host-to-find* [*server*]

nslookup [-*option* ...] [**-** [*server*]]

DESCRIPTION

nslookup is a program to query Internet domain name servers. **nslookup** has been extended to follow the configured name resolution algorithm of the host and to query NIS, as well as, DNS and host tables.

Both an interactive and non-interactive mode are available with **nslookup.** Interactive mode allows the user to query a name server for information about various hosts and domains, or print a list of hosts in the domain. Non-interactive mode is used to query a name server for information about one host or domain.

By default, **nslookup** accesses name services for name and address resolution based on the policy information obtained from the switch configuration file **/etc/nsswitch.conf** When the policy is set to use NIS or **/etc/hosts** first, or when DNS is first but unavailable, then **nslookup** will only provide a limited command set (a **help** command while in this situation will show what actions are possible when querying NIS or **/etc/hosts**.) To override the switch policy and query DNS servers directly, the **server** command can be used to specify a nameserver. This same overriding of the switch policy can also be done by providing a nameserver as the second argument on the command line. In this case, **nslookup** will ignore the switch policy and directly query nameservers, until a *reset* command is issued. Whenever an action is taken that causes the switch policy to be overridden, a warning message is displayed.

Note, NIS+ is not supported by nslookup. If the hosts source **nisplus** is found in the **/etc/nsswitch.conf** file. It will be ignored.

ARGUMENTS

Interactive mode is entered in the following cases:

- No arguments are given.
- The first argument is a hyphen (**-**). The optional second argument is a host name or Internet (IP) address of a name server.

Non-interactive mode is used when the name of the host to be looked up is given as the first argument. The optional second argument is a host name or Internet address of a name server.

Options listed under the **set** command below can be specified one per line in the **.nslookuprc** file in the user's home directory. Alternatively, these options may be specified on the command line by prefixing them with a hyphen and they must precede other command line arguments. For example, to change the default query type to host information, and the initial timeout to 10 seconds, type:

```
nslookup -query=hinfo -timeout=10
```

The command line option **-swdebug** may be used to debug syntactic errors in the switch configuration file. This option turns on tracing during initialization, causing the switch module to print out a trace of the scan and parse actions on the "hosts" entry (see *nsswitch.conf*(4)) in the **/etc/nsswitch.conf** file.

Interactive Commands

Commands can be interrupted at any time by using the interrupt character. To exit, type a Ctrl-D (EOF) or type **exit**. To treat a built-in command as a host name, precede it with an escape character (****). When using NIS or the host table, only host names and Internet addresses are allowed as commands. An unrecognized command is interpreted as a host name.

host [*server*] Look up information for *host* using the current default server or using *server* if specified. If *host* is an Internet address and the query type is **A** or **PTR**, the name of the host is returned. If *host* is a name and does not have a trailing period, one or more domains are appended to the name (this behavior depends on the state of the **set** options **domain**, **srchlist**, **defname**, and **search**). Answers from a name server's cache are labeled "non-authoritative."

server *domain*
lserver *domain*

Change the default server to *domain*. **lserver** uses the initial server to look up information about *domain* while **server** uses the current default server. When **server** is used while the current name service being pointed to is either NIS or **/etc/hosts**, then the switch policy will be overridden until a **reset** is issued.

root Changes the default server to the server for the root of the domain name space. Currently, the host **ns.nic.ddn.mil** is used (this command is a synonym for **lserver** ns.nic.ddn.mil). The name of the root server can be changed with the **set root** command.

policy Prints out the policy read from the switch configuration file. The number of name services specified in the file are shown, as well as the order and criteria on how the name services are to be used. The four statuses of the criteria are represented by the four positions within the square brackets. The order of the statuses are: SUCCESS, NOTFOUND, UNAVAIL and TRYAGAIN. The two actions of the criteria are represented by the two possible letters used in the four status positions: **R** for return and **C** for continue. However, if no criteria is specified between two sources, then the default actions are assigned to the statuses:

SUCCESS=	return
NOTFOUND=	return
UNAVAIL=	continue
TRYAGAIN=	return

finger [*name*] [**>** *filename*]
finger [*name*] [**>>** *filename*]
Connects with the finger server on the current host. The current host is defined when a previous lookup for a host was successful and returned address information (see the **set querytype=A** command). *name* is optional. **>** and **>>** can be used to redirect output in the usual manner.

ls [*option*] *domain* [**>** *filename*]
ls [*option*] *domain* [**>>** *filename*]
List the information available for *domain*, optionally creating or appending to *filename*. The default output contains host names and their Internet addresses. *option* can be one of the following:

-t *querytype*	lists all records of the specified type (see *querytype* below).
-a	lists aliases of hosts in the domain (synonym for **-t CNAME**).
-d	lists all records for the domain (synonym for **-t ANY**).
-h	lists CPU and operating system information for the domain (synonym for **-t HINFO**).
-s	lists well-known services of hosts in the domain (synonym for **-t WKS**).

When output is directed to a file, **#** characters are printed for every 50 records received from the server.

view *filename*
Sorts and lists the output of previous **ls** command(s) using **more** (see *more*(1)).

help
? Prints a brief summary of commands.

exit Exits the program.

reset Returns to the use of the configured name service switch policy and resets to use the original nameservers.

set *keyword*[**=***value*]
This command is used to change state information that affects the lookups. Valid keywords are:

all Prints the current values of the various options to **set**. Information about the current default server and host is also printed.

cl[**ass**]**=***value*
Change the query class to one of:

n

IN	the Internet class.
CHAOS	the Chaos class.
HESIOD	the MIT Athena Hesiod class.
ANY	wildcard (any of the above).

The class specifies the protocol group of the information. (Default = **IN**)

[**no**]**deb**[**ug**] Turn debugging mode on. More information is printed about the packet sent to the server and the resulting answer. (Default = **nodebug**)

[**no**]**d2** Turn exhaustive debugging mode on. Essentially all fields of every packet are printed. (Default = **nod2**)

[**no**]**def**[**name**]
If set, append the default domain name to a single-component lookup request (i.e., one that does not contain a period character). (Default = **defname**)

do[**main**]**=***name*
Change the default domain name to *name*. The default domain name is appended to a lookup request, depending on the state of the **defname** and **search** options. The domain search list contains the parents of the default domain if it has at least two components in its name. For example, if the default domain is **CC.Berkeley.EDU**, the search list is **CC.Berkeley.EDU** and **Berkeley.EDU**. Use the **set srchlist** command to specify a different list. Use the **set all** command to display the list. (Default = value from hostname, **/etc/resolv.conf** or **LOCALDOMAIN**)

[**no**]**ig**[**nore**] Ignore truncation errors. (Default = **noignore**)

q[**uerytype**]**=***value*
ty[**pe**]**=***value* Change the type of information returned from a query to one of:

A	Host's Internet address
ANY	All types of data
CNAME	Canonical name for an alias
GID	Group ID
HINFO	Host CPU and operating system type
MB	Mailbox domain name
MG	Mail group member
MINFO	Mailbox or mail list information
MR	Mail rename domain name
MX	Mail exchanger
NS	Name server for the named zone
PTR	Host name if the query is an Internet address, otherwise the pointer to other information.
SOA	Start of authority record
TXT	Text information
UID	User ID
UINFO	User information
WKS	Well-known service description

po[**rt**]**=***value* Change the default TCP/UDP name server port to *value*. (Default = 53)

[**no**]**rec**[**urse**]
: Tell the name server to query other servers if it does not have the information. (Default = **recurse**)

ret[**ry**]**=***number*
: Set the number of retries to *number*. When a reply to a request is not received within a certain amount of time (changed with **set timeout**), the timeout period is doubled and the request is resent. The retry value controls how many times a request is resent before giving up. (Default = 4)

ro[**ot**]**=***host*
: Change the name of the root server to *host*. This affects the **root** command. (Default = **ns.nic.ddn.mil**)

[no]sea[rch]
: If the lookup request contains at least one period but doesn't end with a trailing period, append the domain names in the domain search list to the request until an answer is received. See *hostname*(5). (Default = **search**)

srchl[**ist**]=*name1*/*name2*/...
: Change the default domain name to *name1* and the domain search list to *name1*, *name2*, etc. A maximum of 6 names separated by slashes (**/**) can be specified. For example,

```
set srchlist=lcs.MIT.EDU/ai.MIT.EDU/MIT.EDU
```

sets the domain to **lcs.MIT.EDU** and the search list to the three names. This command overrides the default domain name and search list of the **set domain** command. Use the **set all** command to display the list. (Default = value based on hostname, **/etc/resolv.conf** or **LOCALDOMAIN**)

[no]swtr[ace]
: When set, this flag causes **nslookup** to print out information about the sources used for resolving a name or an address lookup. This flag traces the behavior generated by the switch policy. (Default = **noswtrace**)

t[**imeout**]**=***number*
: Change the initial timeout interval for waiting for a reply to *number* seconds. Each retry doubles the timeout period. (Default = 5 seconds)

[**no**]**v**[**c**]
: Always use a virtual circuit when sending requests to the server. (Default = **novc**)

DIAGNOSTICS

If the lookup request was not successful, an error message is printed. Possible errors are:

Time-out
: The server did not respond to a request after a certain amount of time (changed with **set timeout=***value*) and a certain number of retries (changed with **set retry=***value*).

No response from server
: No name server is running on the server machine.

No records
: The server does not have resource records of the current query type for the host, although the host name is valid. The query type is specified with the **set querytype** command.

Non-existent domain
: The host or domain name does not exist.

Connection refused
Network is unreachable
: The connection to the name server could not be made at the present time.

Server failure
: The name server found an internal inconsistency in its database and could not return a valid answer.

n

`Refused`
The name server refused to service the request.

`Format error`
The name server found that the request packet was not in the proper format.

AUTHOR

`nslookup` was developed by the University of California, Berkeley.

FILES

`/etc/resolv.conf`	Initial domain name and name server addresses
`$HOME/.nslookuprc`	User's initial options

SEE ALSO

named(1M), resolver(3N), resolver(4), nsswitch.conf(4), hostname(5),

RFC1034, RFC1035

NAME

nsquery - query the Name Service Switch backend libraries

SYNOPSIS

nsquery *lookup_type lookup_query* [*lookup_policy*]

DESCRIPTION

nsquery is used to find the Name Service that returned the response to a **gethostbyname()**, **gethostbyaddr()**, **getpwnam()**, **getpwuid()**, **getgrnam()**, or **getgrgid()** function call. This application is Name Service Switch aware and follows the lookup policies in **/etc/nsswitch.conf**. The lookup types supported are:

hosts	Used to resolve host name or IP Address lookups.
passwd	Used to resolve user name or UID lookups.
group	Used to resolve group name or GID lookups.

The lookup query can either be a host name, IP Address, user name, user ID, group name or group ID.

The lookup policy must be a valid lookup policy described in **nsswitch.conf(4)**. If the policy is invalid, the system default policy will be used.

The default policies are:

hosts	dns [NOTFOUND=return TRYAGAIN=return] nis [NOTFOUND=return] files
passwd	files nis
group	files nis

nsquery will display the lookup policy being used, the name of the service being queried, and the result of the query.

If the result of the query was successful, the appropriate structure will be displayed.

EXAMPLES

nsquery hosts hondo

nsquery hosts 15.204.204.204 "dns files"

nsquery passwd dog "nisplus"

nsquery passwd 105

nsquery group wayne "nis [NOTFOUND=RETURN] files"

nsquery group 22

RETURN VALUE

0:	Success.
1:	Invalid Usage.
2:	Unknown ACTION.
3:	No match found in any name services queried.

AUTHOR

nsquery was developed by Hewlett-Packard.

SEE ALSO

nsswitch.conf(4).

NOTES

Changing the default behavior for SUCCESS is not recommended.

NAME

od, xd - octal and hexadecimal dump

SYNOPSIS

od [**-v**] [**-A** *address_base*] [**-j** *skip*] [**-N** *count*] [**-t** *type_string*] ... [*file* ...]

xd [**-v**] [**-A** *address_base*] [**-j** *skip*] [**-N** *count*] [**-t** *type_string*] ... [*file* ...]

Supported Pre-POSIX Usage

od [**-bcdosx**] [*file*] [[**+**][**0x**]*offset*[**.**][**b**]]

xd [**-bcdosx**] [*file*] [[**+**][**0x**]*offset*[**.**][**b**]]

DESCRIPTION

od and **xd** concatenate one or more input *file*s and write their contents to standard output in a user-specified format. If *file* is not specified, the standard input is used.

Options and Arguments

od and **xd** recognize the following options and command-line arguments:

-A *address_base* Specify the input offset base. *address_base* is a single character that defines which format the offset base is written in:

d	Decimal format.
o	Octal format.
x	Hexadecimal format.
n	Do not write the offset.

-j *skip* Jump over *skip* bytes from the beginning of the input. **od** seeks past the first *skip* bytes in the concatenated input files. If the combined input is not at least *skip* bytes long, **od** writes a diagnostic message to standard error and exits with a non-zero exit status. By default, *skip* is interpreted as a *decimal* number. If *skip* has a leading **0x** or **0X**, it is interpreted as a *hexadecimal* number; a leading **0** indicates that *skip* is an *octal* number.

If the value of *skip* is followed by a **b**, **k**, or **m**, it is interpreted as a multiple of 512, 1024, or 1048576, respectively.

O

-N *count* Format no more than *count* bytes of input.

By default, *count* is interpreted as a *decimal* number. A leading **0x** or **0X** indicates that *count* is a *hexadecimal* number; a leading **0** identifies an *octal* value.

If *count* bytes of input are not available (after successfully skipping if **-j***skip* is specified), the input that is available is formatted.

-t *type_string* *type_string* is a string defining the types to be used when writing the input data.

The string can contain any of the following type-specification characters:

a	named character ,
c	character ,
d	signed decimal ,
f	floating point ,
o	octal ,
u	unsigned decimal ,
x	hexadecimal ,

Type specification characters **d**, **f**, **o**, **u**, and **x** can be followed by an optional *unsigned decimal* integer specifying the number of bytes to be transformed by each instance of the output type, or by an optional **C**, **S**, **I**, or **L** indicating that the conversion should be applied to an item of type *char*, *short*, *int*, or *long*, respectively.

Type specification character **f** can be followed by an optional **F**, **D**, or **L** indicating that the conversion should be applied to an item of type *float*, *double*, or *long double*, respectively.

Multiple types can be concatenated within the same *type_string* and multiple **-t** options can be specified. Output lines are written for each type specified in the order in which the type specification characters appear.

-v Write all input data. Without the **-v** option, any number of groups of output lines, that would be identical to the immediately preceding group of output lines (except for the byte offsets), are replaced with a line containing only an asterisk (*****).

file Pathname of one or more input files to be processed. If *file* is not specified, the standard input is used.

Input files can be any file type.

DESCRIPTION OF PRE-POSIX USAGE

od and **xd** dump *file* in one or more formats as selected by the first argument. If the first argument is missing, the default is **-o** for **od**; **-x** for **xd**. An offset field is inserted at the beginning of each line. For **od**, the offset is in octal, for **xd**, the offset is in hexadecimal.

Options

od and **xd** recognize the following format options:

-b Interpret bytes in octal (hexadecimal).

-c Interpret bytes in ASCII. Certain non-graphic characters appear as C escapes: null=**\0**, backspace=**\b**, form-feed=**\f**, new-line=**\n**, return=**\r**, tab=**\t**; others appear as 3-digit octal numbers.

-d Interpret 16-bit words in decimal.

-o Interpret 16-bit words in octal.

-s Interpret 16-bit words in signed decimal.

-x Interpret 16-bit words in hexadecimal.

file specifies which file is to be dumped. If *file* is not specified, the standard input is used.

offset specifies the offset in the file where dumping is to commence, and is normally interpreted as octal bytes. Interpretation can be altered as follows:

- *offset* must be preceded by **+** if the file argument is omitted.
- *offset* preceded by **0x** is interpreted in hexadecimal.
- *offset* followed by **.** is interpreted in decimal.
- *offset* followed by **b** is interpreted in blocks of 512 bytes.

Dumping continues until end-of-file.

EXAMPLES

Write hexadecimal bytes and the corresponding octal values to the standard output in blocks of 16 bytes in one line, by transforming the data from the input file **file1**:

```
od -tx1oC file1
```

The following commands write one line each of the types *character*, *signed decimal integer*, and *float*, in the order given, transforming 100 bytes of data starting from fifteenth byte offset in the file **file1**:

```
od -j14 -N100 -tc -tdfF file1
od -j0xe -N100 -tcd4fF file1
```

Write one line each of the types *unsigned integer*, *named character*, and *long double*, with the offsets written in hexadecimal and forcing a write, even on lines that are identical to the immediately preceding group of output lines:

```
od -v -Ax -tuafL file1
```

WARNINGS

When the output format is of floating-point type; i.e., when using the **-t fD**, **-t fL**, or **-t f** options:

- If the input bytes cannot be transformed into a valid floating point number, a floating point exception might occur. In that case, the output is printed as a string containing some non-numeric characters and program execution continues.

O

- When the number of input bytes used for transformation is set to 1 with the type specifier characters **d**, **o**, **u**, or **x**, only the least-significant seven bits of each byte are used.
- When one or more of the **-A**, **-j**, **-N**, or **-t** options is specified, an operand starting with the first character as a plus-sign (**+**) or the first character as numeric is interpreted as a file name.

(XPG4 only. Multiple types can be specified by using multiple **-bcdox** options. Output lines are written for each type specified in the order in which the types are specified.)

EXTERNAL INFLUENCES

Environment Variables

LANG provides a default value for the internationalization variables that are unset or null. If **LANG** is unset or null, the default value of "C" (see *lang*(5)) is used. If any of the internationalization variables contains an invalid setting, **od** will behave as if all internationalization variables are set to "C". See *environ*(5).

LC_ALL If set to a non-empty string value, overrides the values of all the other internationalization variables.

LC_CTYPE determines the interpretation of text as single and/or multi-byte characters, the classification of characters as printable, and the characters matched by character class expressions in regular expressions.

LC_MESSAGES determines the locale that should be used to affect the format and contents of diagnostic messages written to standard error and informative messages written to standard output.

NLSPATH determines the location of message catalogues for the processing of **LC_MESSAGES**.

International Code Set Support

Single- and multi-byte character code sets are supported. Multi-byte data is displayed as multi-byte values.

RETURN VALUE

Exit values are:

0	Successful completion.
>0	Error condition occurred.

SEE ALSO

adb(1).

STANDARDS CONFORMANCE

od: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

NAME

on - execute command on remote host with environment similar to local

SYNOPSIS

on [**-i** | **-n**] [**-d**] *host* [*command* [*argument*] ...]

DESCRIPTION

on executes a command on a remote host, using an environment similar to that of the invoking user where:

host	specifies the name of the host on which to execute the command.
command	specifies the command to execute on *host*

If *command* is not specified, **on** starts a shell on *host*. *argument* ... is a list of arguments for *command*.

The user's environment variables are copied to the remote host, and the file system containing the user's current working directory is NFS mounted on the remote host (see *nfs*(7)). The command is executed on the remote host in the user's current working directory.

Commands using relative path names that reference file system objects within the user's current working file system have the same behavior as running the command on the client. The behavior of commands using relative path names that cross the file system boundary or commands using absolute path names depends on the organization of the remote host's file system.

Implicit and explicit use of environment variables may also cause a command's behavior to be dependent on the organization of the remote host's file system. For example, the **$PATH** environment variable usually contains absolute path names.

Standard input, output and error of the remote command are connected to the appropriate file descriptors on the client.

The remote execution daemon (**rexd**) does not allow **root** to execute a remote command.

The signals **SIGINT**, **SIGTERM**, and **SIGQUIT** are propagated to the remote command. **SIGTSTP** and **SIGSTOP** are ignored by the remote command. All other signals are delivered to the **on** command.

In order to execute a remote command, the remote host must be configured to execute **rexd** (see *rexd*(1M)).

Options

on recognizes the following options:

-i Interactive mode. This option is required for commands that must communicate with a terminal such as **vi**, **ksh**, or **more**. Terminal mode changes are propagated to the **rexd** server. The standard input for an interactive **on** command must be a tty device. The **-i** and **-n** options are mutually exclusive.

-d Debug mode. Print diagnostic messages during startup of the **on** command. These messages are useful for detecting configuration problems if the **on** command to a specific host is failing.

-n No input mode. This option causes the remote command to get end-of-file (EOF) when it reads from standard input, instead of connecting the standard input of the **on** command to the standard input of the remote command. The **-n** option is required when running commands in the background. The **-n** and **-i** options are mutually exclusive.

DIAGNOSTICS

on: unknown host *host*
The host name *host* was not found in the hosts database.

on: cannot connect to server on *host*
The host *host* is down, unreachable on the network, or not running **rexd**.

on: can't find *current_dir*
A problem occurred trying to find the user's current working directory (*current_dir*).

on: can't locate mount point for *current_dir*
A problem occurred trying to determine the mount point of the user's current working directory (*current_dir*).

on: standard input (stdin) is not a tty
The standard input (stdin) of the **on** command with the **-i** option is not a tty device.

on *server* **: rexd:** *message*
Errors that occur on the server *server* are propagated back to the client. These messages are documented in the DIAGNOSTICS section of *rexd*(1M).

AUTHOR

on was developed by Sun Microsystems, Inc.

SEE ALSO

exports(4), rexd(1M).

NAME

pack, pcat, unpack - compress and expand files

SYNOPSIS

pack [**-**] [**-f**] *name* ...

pcat *name* ...

unpack *name* ...

DESCRIPTION

pack attempts to store the specified files in a compressed form. Wherever possible, each input file *name* is replaced by a packed file *name***.z** with the same ownership, modes, and access and modification times. The **-f** option forces packing of *name*. This is useful for causing an entire directory to be packed even if some of the files do not benefit. If **pack** is successful, *name* is removed. Packed files can be restored to their original form using **unpack** or **pcat**.

pack uses Huffman (minimum redundancy) codes on a byte-by-byte basis. If the **-** argument is used, an internal flag is set that causes the number of times each byte is used, its relative frequency, and the code for the byte to be printed on the standard output. Additional occurrences of **-** in place of *name* cause the internal flag to be set and reset.

The amount of compression obtained depends on the size of the input file and the character frequency distribution. Because a decoding tree forms the first part of each **.z** file, it is usually not worthwhile to pack files smaller than three blocks unless the character frequency distribution is very skewed such as in printer plots or pictures.

Typically, text files are reduced to 60-75% of their original size. Load modules, which use a larger character set and have a more uniform distribution of characters, show little compression, the packed versions being about 90% of the original size.

pack returns a value that is the number of files that it failed to compress.

No packing occurs if:

- The file appears to be already packed.
- The file name has more than 12 characters and the file system is configured as a short filename system.
- The file has links.
- The file is a directory.
- The file cannot be opened.
- The file is empty.
- No disk storage blocks will be saved by packing.
- A file called *name***.z** already exists.
- The **.z** file cannot be created.
- An I/O error occurred during processing.

On short-filename systems, the last segment of the file name must contain no more than 12 characters to allow space for the appended **.z** extension. Directories cannot be compressed.

pcat does for packed files what *cat*(1) does for ordinary files, except that **pcat** cannot be used as a filter. The specified files are unpacked and written to the standard output. Thus to view a packed file named *name***.z** use:

```
pcat name.z
```

or simply:

```
pcat name
```

To make an unpacked copy (named *file*) of a packed file named *name***.z** without destroying *name***.z**) use the command:

```
pcat name >file
```

pcat returns the number of files it was unable to unpack. Failure may occur if:

- The file name (exclusive of the **.z**) has more than 12 characters;
- The file cannot be opened;
- The file does not appear to have been created by *pack*.

p

unpack expands files created by **pack**. For each file *name* specified in the command, a search is made for a file called *name*.**z** (or just *name* if *name* ends in **.z**). If this file appears to be a packed file, it is replaced by its expanded version. The new file has the **.z** suffix stripped from its name, and has the same access modes, access and modification dates, and owner as those of the packed file.

unpack returns a value that is the number of files it was unable to unpack. Failure may occur for the reasons given for **pcat**, as well as for the following:

- A file with the "unpacked" name already exists;
- The unpacked file cannot be created.

Access Control Lists (ACLs)

pack retains all entries in a file's access control list when compressing and expanding it (see *acl*(5)).

DEPENDENCIES

NFS

Optional access control list entries of networked files are summarized (as returned in **st_mode** by **stat()**, but not copied to the new file (see *stat*(2)).

WARNINGS

This command is likely to be withdrawn from X/Open standards. Applications using this command might not be portable to other vendors' systems. Instead of **pack** it is recommended to use **compress** utility as it has the following advantages:

- The algorithm used to create the output files is frequently more effective in reducing the size of files
- The compress utility can compress data from its standard input, not just a named regular file. Thus it is useful in pipelines

zcat is recommended instead of **pcat** and, **uncompress** is recommended instead of **unpack**.

SEE ALSO

cat(1), compact(1), compress(1), acl(5).

STANDARDS CONFORMANCE

pack: SVID2, SVID3, XPG2, XPG3

pcat: SVID2, SVID3, XPG2, XPG3

unpack: SVID2, SVID3, XPG2, XPG3

NAME

passwd - change login password and associated attributes

SYNOPSIS

passwd [*name*]

passwd -r files [**-F** *file*] [*name*]

passwd -r files [**-e** [*shell*]] [**-gh**] [*name*]

passwd -r files -s [**-a**]

passwd -r files -s [*name*]

passwd -r files [**-d**|**-l**] [**-f**] [**-n** *min*] [**-w** *warn*] [**-x** *max*] *name*

passwd -r nis [**-e** [*shell*]] [**-gh**] [*name*]

passwd -r nisplus [**-e** [*shell*]] [**-gh**] [**-D** *domain*] [*name*]

passwd -r nisplus -s [**-a**]

passwd -r nisplus -s [**-D** *domain*] [*name*]

passwd -r nisplus [**-l**] [**-f**] [**-n** *min*] [**-w** *warn*] [**-x** *max*] [**-D** *domain*] *name*

passwd -r dce [**-e** [*shell*]] [**-gh**] [*name*]

DESCRIPTION

The **passwd** command modifies the password as well as the attributes associated with the login *name*. If *name* is omitted, it defaults to the invoking user's login *name*, which is determined using *getlogin*(3C)

The default password file is **/etc/passwd**. The **-F** option can be used to choose an alternate password file, where read and write permissions are required. This option is only available using the **files** repository.

Ordinary users can only change passwords corresponding to their login *name*. If an old password has been established, it is requested from the user. If valid, a new password is obtained. Once the new password is entered, it is determined if the old password has "aged" sufficiently. If password aging is not sufficient, the new password is rejected and **passwd** terminates (see *passwd*(4)).

If password aging and construction requirements are met, the password is re-entered to ensure consistency. If the new copy differs, **passwd** repeats the new password prompting cycle three times.

A superuser, whose effective user ID is zero (see *id*(1) and *su*(1)), is allowed to change any password and is not forced to comply with password aging. Superusers are not prompted for old passwords unless they are attempting to change the superuser's password. In addition, on untrusted systems, superusers are not forced to comply with password construction requirements. Null passwords can be created by entering a carriage return in response to the prompt for a new password.

The DCE repository (**-r dce**) is only available if Integrated Login has been configured, see *auth.adm*(1M). If Integrated Login has been configured, other considerations apply. A user with appropriate DCE privileges is capable of modifying a user's password, shell, geco or home directory - this is not dependent upon superuser privileges.

If the repository is not specified, i.e. **passwd** [*name*], the password is changed in all existing repositories configured in **/etc/nsswitch.conf**. If password options are used, and no repository is specified, the default repository is **files**.

Options

The following options are recognized:

-D *domain* Use the **passwd.org_dir** in the specified *domain*. This option is for **nisplus** repositories only. If not specified, the default *domain* is returned.

-e *shell* Modify the default shell for the user's login *name* in the password file. If the *shell* is not provided, the user will be prompted to enter the default login shell.

-F *name* Choose an alternative password file, where read and write permissions are required. This option is available for the **files** repository only.

-g Change the gecos information in the password file, which is used by the **finger** command. The user is prompted for each subfield: name, location, work phone, and home

p

	phone.
-h	Modify the default home directory in the password file. Only superuser is allowed to exercise this option.
-r	Specify the repository to which the operation is to be applied. Supported repositories include **files**, **nis**, **nisplus**, and **dce**. If repository is not specified, the default is **files**.
-s *name*	Display password attributes associated with the specified *name*. Superuser privilege is required if the **files** repository is specified. For **nisplus**, there are no restrictions.
-s [**-a**]	Display password attributes for all users in the password file. The **-a** option must be used in conjunction with the **-s** option when no *name* is specified. For **nisplus**, this will display entries in the NIS+ **passwd** table in the local domain. For **files**, this is restricted to superuser.

Privileged User Options

A superuser can modify password aging characteristics associated with the user *name* using the following options:

-d	Allow user to login without a password by deleting it.
-f	Force user to change password upon next login by expiring the current password.
-l	Lock user account.
-n *min*	Determine the minimum number of days, *min*, that must transpire before the user can change the password.
-w *warn*	Specify the number of days, *warn*, prior to the password expiring when the user will be notified that the password needs to be changed. This option is only enabled when the system has been converted to a trusted, secure system. Refer to the *Managing Systems and Workgroups* manual for how to convert your HP-UX to a trusted, secure system.
-x *max*	Determine the maximum number of days, *max*, a password can remain unchanged. The user must enter another password after that number of days has transpired, known as the password *expiration time*.

The *min* and *max* arguments are each represented in units of days. These arguments will be rounded up to the nearest week on a nontrusted HP-UX system. If the system is then converted to a trusted system, the number of days will be based on those weeks. If only one of the two arguments is supplied, then, if the other one does not exist, it is set to zero.

Password Construction Requirements

Passwords must be constructed to meet the following requirements:

- A password must have at least six characters. Only the first eight characters are significant in an untrusted system.
- Characters must be from the 7-bit US-ASCII character set; letters from the English alphabet.
- A password must contain at least two letters and at least one numeric or special character.
- A password must differ from the user's login *name* and any reverse or circular shift of that login *name*. For comparison purposes, an uppercase letter and its corresponding lowercase equivalent are treated as identical.
- A new password must differ from the old one by at least three characters (one character in a trusted system). For comparison purposes, an uppercase letter and its corresponding lowercase equivalent are treated as identical.

If the above restrictions are met, the **/etc/nsswitch.conf** file specifies the repositories for which the password must be modified. The following configurations are supported:

- passwd: files
- passwd: files nisplus
- passwd: files nis
- passwd: compat (--> files nis)

- passwd: compat (--> files nisplus)
- passwd_compat: nisplus

Smart Card Login

If the user account is configured to use a Smart Card, the user password is stored in the card. This password has characteristics identical to a normal password stored on the system.

The password it is retrieved automatically from the Smart Card when a valid PIN is entered. Therefore, it is not necessary to know the password, only the PIN.

Since `passwd` can be used with a Smart Card account, the Smart Card must be inserted into the Smart Card reader. The user is prompted for a PIN instead of a password during authentication.

```
Enter PIN:
```

If the system retrieves a valid old password from the card, a new password is requested (twice). If the new password meets all requirements, the system automatically overwrites the old password stored on the card with the new password.

Therefore, the new dialog resembles:

```
Enter PIN:
New password:
Re-enter new password:
```

A Smart Card account can be shared among users. If one user modifies the password, other users must use the `scsync` command to write the new password onto their cards.

The `scpin` command is used to change the Smart Card PIN.

SECURITY FEATURES

This section applies only to trusted systems. It describes additional capabilities and restrictions.

When `passwd` is invoked on a trusted system, the existing password is requested (if one is present). This initiates the password solicitation dialog which depends upon the type of password generation that has been enabled on the account. There are four possible options for password generation:

Random syllables	A pronounceable password made up of meaningless syllables.
Random characters	An unpronounceable password made up of random characters from the character set.
Random letters	An unpronounceable password made up of random letters from the alphabet.
User-supplied	A user-supplied password, subject to length and triviality restrictions.

Passwords can be greater than eight characters. The system administrator can specify the password length guidelines for the system generated options (random syllables, random characters, and random letters). The minimum password length depends upon several parameters set by the system administrator in the authentication database. System warnings are displayed if passwords lengths are either too long or short.

The system requires a *minimum time* to elapse before a password can be changed. This prevents reuse of an old password within an undesirable period of time.

A password expires after a period of time known as the *expiration time*. System warnings are displayed as expiration time approaches.

A password dies after a time period known as the *password lifetime*. After the lifetime passes, the account is locked until it is re-enabled by a system administrator. Once unlocked, the user is forced to change the password before account use.

The system administrator can enable accounts without passwords. If a user account is allowed to function without a password, the user can choose a null password by typing a carriage-return when prompted for a new password.

EXTERNAL INFLUENCES

International Code Set Support

Characters from single-byte character code sets are supported in passwords.

EXAMPLES

Change the password expiration date of **user** to 42 days in the **files** repository:

```
passwd -r files -x 42 user
```

Modify the minimum time between password changes of **user1** to 7 days in the **nisplus** repository:

```
passwd -r nisplus -n 7 user1
```

Force **user2** to establish a new password on the next login which will expire in 70 days and prohibit the user from changing the password until 7 days have transpired:

```
passwd -r files -f -x 70 -n 7 user2
```

DEPENDENCIES

Pluggable Authentication Modules (PAM)

PAM is an Open Group standard for user authentication, password modification, and account validation. In particular, **pam_chauthtok()** is invoked to perform all functions related to **passwd**. This includes establishing and changing a password, using **passwd** options, and displaying error messages.

FILES

/etc/passwd	Standard password file used by HP-UX.
/tcb/files/auth/*/*	Protected password database used when system is converted to trusted system.

SEE ALSO

chfn(1), id(1), login(1), su(1), crypt(3C), getlogin(3C), passwd(4), auth(5), auth.adm(1M), auth.dce(5).

Managing Systems and Workgroups

Pluggable Authentication Modules (PAM)

pam_chauthtok(3), pam(3), pam.conf(4), pam_user.conf(4).

HP-UX Smart Card Login

scpin(1), scsync(1).

STANDARDS CONFORMANCE

passwd: SVID2, SVID3, XPG2

NAME

paste - merge same lines of several files or subsequent lines of one file

SYNOPSIS

paste *file1 file2 ...*

paste -d *list file1 file2 ...*

paste -s [**-d** *list*] *file1 file2 ...*

DESCRIPTION

In the first two forms, **paste** concatenates corresponding lines of the given input files *file1*, *file2*, etc. It treats each file as a column or columns in a table and pastes them together horizontally (parallel merging). In other words, it is the horizontal counterpart of *cat*(1) which concatenates vertically; i.e., one file after the other. In the **-s** option form above, **paste** replaces the function of an older command with the same name by combining subsequent lines of the input file (serial merging). In all cases, lines are glued together with the *tab* character, or with characters from an optionally specified *list*. Output is to standard output, so **paste** can be used as the start of a pipe, or as a filter if **-** is used instead of a file name.

paste recognizes the following options and command-line arguments:

-d Without this option, the new-line characters of all but the last file (or last line in case of the **-s** option) are replaced by a *tab* character. This option allows replacing the *tab* character by one or more alternate characters (see below).

list One or more characters immediately following **-d** replace the default *tab* as the line concatenation character. The list is used circularly; i.e., when exhausted, it is reused. In parallel merging (that is, no **-s** option), the lines from the last file are always terminated with a new-line character, not from the *list*. The list can contain the special escape sequences: **\n** (new-line), **\t** (tab), \\ (backslash), and **\0** (empty string, not a null character). Quoting may be necessary if characters have special meaning to the shell. (For example, to get one backslash, use **-d"\\\\"**).

-s Merge subsequent lines rather than one from each input file. Use *tab* for concatenation, unless a *list* is specified with the **-d** option. Regardless of the *list*, the very last character of the file is forced to be a new-line.

- Can be used in place of any file name to read a line from the standard input (there is no prompting).

EXTERNAL INFLUENCES

Environment Variables

LC_CTYPE determines the locale for the interpretation of text as single- and/or multi-byte characters.

LC_MESSAGES determines the language in which messages are displayed.

If **LC_CTYPE** or **LC_MESSAGES** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**.

If any internationalization variable contains an invalid setting, **paste** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported.

RETURN VALUE

These commands return the following values upon completion:

0 Completed successfully.

>0 An error occurred.

EXAMPLES

List directory in one column:

```
    ls | paste -d" " -
```

List directory in four columns

```
    ls | paste - - - -
```

Combine pairs of lines into lines

```
    paste -s -d"\t\n" file
```

Notes

pr -t -m... works similarly, but creates extra blanks, tabs and new-lines for a nice page layout.

DIAGNOSTICS

too many files Except for the **-s** option, no more than **OPEN_MAX** − 3 input files can be specified (see *limits*(5)).

AUTHOR

paste was developed by OSF and HP.

SEE ALSO

cut(1), grep(1), pr(1).

STANDARDS CONFORMANCE

paste: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

NAME

patch - a program for applying a diff file to an original

SYNOPSIS

Non-XPG4 version

patch [*options*] *orig patchfile* [**+**[*options*] *orig*]

patch <*patchfile* **# usual form**

XPG4 version

patch [**-blNR**] [**-c|-e|-n**] [**-d** *dir*] [**-D** *define*] [**-i** *patchfile*] [**-o** *outfile*] [**-p** *num*]
[**-r** *rejectfile*] [*file*]

DESCRIPTION

patch will take a patch file containing any of the three forms of difference listing produced by the *diff* program (normal, context or in the style of ed) and apply those differences to an original file, producing a patched version. By default, the patched version is put in place of the original, with the original file backed up to the same name with the extension "**.orig**", or as specified by the **-b** switch. Note that functionality of this option varies for XPG4 version. You may also specify where you want the output to go with a **-o** switch. If *patchfile* is omitted, or is a hyphen, the patch will be read from standard input. For XPG4 version, patchfile has to be specified as argument to **-i** switch. If this option is omitted or a hyphen is specified as argument, the patch will read from standard input.

Upon startup, patch will attempt to determine the type of the diff listing, unless over-ruled by a **-c**, **-e**, or **-n** switch. Context diffs and normal diffs are applied by the *patch* program itself, while ed diffs are simply fed to the *ed* editor via a pipe.

patch will try to skip any leading garbage, apply the diff, and then skip any trailing garbage. Thus you could feed an article or message containing a diff listing to *patch*, and it should work. If the entire diff is indented by a consistent amount, this will be taken into account.

With context diffs, and to a lesser extent with normal diffs, *patch* can detect when the line numbers mentioned in the patch are incorrect, and will attempt to find the correct place to apply each hunk of the patch. As a first guess, it takes the line number mentioned for the hunk, plus or minus any offset used in applying the previous hunk. If that is not the correct place, *patch* will scan both forwards and backwards for a set of lines matching the context given in the hunk. First *patch* looks for a place where all lines of the context match. If no such place is found, and it's a context diff, and the maximum fuzz factor is set to 1 or more, then another scan takes place ignoring the first and last line of context. If that fails, and the maximum fuzz factor is set to 2 or more, the first two and last two lines of context are ignored, and another scan is made. (The default maximum fuzz factor is 2.) Note that for XPG4 version maximum fuzz factor can not be specified as an option and the default maximum fuzz factor is used. If *patch* cannot find a place to install that hunk of the patch, it will put the hunk out to a reject file, which normally is the name of the output file plus "**.rej**". (Note that the rejected hunk will come out in context diff form whether the input patch was a context diff or a normal diff. If the input was a normal diff, many of the contexts will simply be null.) The line numbers on the hunks in the reject file may be different than in the patch file: they reflect the approximate location patch thinks the failed hunks belong in the new file rather than the old one.

As each hunk is completed, you will be told whether the hunk succeeded or failed, and which line (in the new file) *patch* thought the hunk should go on. If this is different from the line number specified in the diff you will be told the offset. A single large offset MAY be an indication that a hunk was installed in the wrong place. You will also be told if a fuzz factor was used to make the match, in which case you should also be slightly suspicious. Note that XPG4 version does not support verbose option. So, most of the diagnostic messages are not printed for this version. However user queries will always be displayed.

If no original file is specified on the command line, *patch* will try to figure out from the leading garbage what the name of the file to edit is. In the header of a context diff, the filename is found from lines beginning with "*******" or "**---**", with the shortest name of an existing file winning. Only context diffs have lines like that, but if there is an "**Index:**" line in the leading garbage, *patch* will try to use the filename from that line. The context diff header takes precedence over an Index line. If no filename can be intuited from the leading garbage, you will be asked for the name of the file to patch.

(If the original file cannot be found, but a suitable SCCS or RCS file is handy, *patch* will attempt to get or check out the file.)

Additionally, if the leading garbage contains a "**Prereq:** " line, *patch* will take the first word from the prerequisites line (normally a version number) and check the input file to see if that word can be found. If not, *patch* will ask for confirmation before proceeding.

The upshot of all this is that you should be able to say, while in a news interface, the following:

```
| patch -d /usr/src/local/blurfl
```

and patch a file in the blurfl directory directly from the article containing the patch.

If the patch file contains more than one patch, *patch* will try to apply each of them as if they came from separate patch files. This means, among other things, that it is assumed that the name of the file to patch must be determined for each diff listing, and that the garbage before each diff listing will be examined for interesting things such as filenames and revision level, as mentioned previously. You can give switches (and another original file name) for the second and subsequent patches by separating the corresponding argument lists by a “**+**”. (The argument list for a second or subsequent patch may not specify a new patch file, however.)

With XPG4 version, processing of multiple patches varies considerably. You can not specify different options for different patches. Options remain same for all the patches. This also affects the contents of output file specified with the **-o** option. See the description of this option for more details.

patch recognizes the following switches:

-b causes the next argument to be interpreted as the backup extension, to be used in place of “**.orig**”. (For XPG4 version this option varies. With this option no argument is required and the option only enables the backup process. Always default extension is used.)

-c forces *patch* to interpret the patch file as a context diff.

-d causes *patch* to interpret the next argument as a directory, and cd to it before doing anything else.

-D causes *patch* to use the "#ifdef...#endif" construct to mark changes. The argument following will be used as the differentiating symbol. Note that, unlike the C compiler, there must be a space between the **-D** and the argument. (For XPG4 version this option varies. With this version "#ifndef" constructor is not used.)

-e forces *patch* to interpret the patch file as an ed script.

-f forces *patch* to assume that the user knows exactly what he or she is doing, and to not ask any questions. It does not suppress commentary, however. Use **-s** for that. (This option is not supported by XPG4 version.)

p

-F *number*
sets the maximum fuzz factor. This switch only applied to context diffs, and causes *patch* to ignore up to that many lines in looking for places to install a hunk. Note that a larger fuzz factor increases the odds of a faulty patch. The default fuzz factor is 2, and it may not be set to more than the number of lines of context in the context diff, ordinarily 3. (This option is not supported by XPG4 version.)

-i This option is supported only by XPG4 version. It causes next argument to be interpreted as the patch file name.

-l causes the pattern matching to be done loosely, in case the tabs and spaces have been munged in your input file. Any sequence of whitespace in the pattern line will match any sequence in the input file. Normal characters must still match exactly. Each line of the context must still match a line in the input file.

-n forces *patch* to interpret the patch file as a normal diff.

-N causes *patch* to ignore patches that it thinks are reversed or already applied. See also **-R**.

-o causes the next argument to be interpreted as the output file name. There are some added features for the XPG4 version. Multiple patches for a single file will be applied to the intermediate versions of the file created by any previous patches, and will result in multiple,concatenated versions of the file being written to output file.

-p *number*
sets the pathname strip count, which controls how pathnames found in the patch file are treated, in case the you keep your files in a different directory than the person who sent out the patch. The strip count specifies how many backslashes are to be stripped from the front of the pathname. (Any intervening directory names also go away.) For example, supposing the filename in the patch file was

```
/u/howard/src/blurfl/blurfl.c
```

setting **-p** or **-p0** gives the entire pathname unmodified, **-p1** gives

```
u/howard/src/blurfl/blurfl.c
```

without the leading slash, **-p4** gives

```
blurfl/blurfl.c
```

and not specifying **-p** at all just gives you "blurfl.c". Whatever you end up with is looked for either in the current directory, or the directory specified by the **-d** switch.

-r causes the next argument to be interpreted as the reject file name.

-R tells *patch* that this patch was created with the old and new files swapped. (Yes, I'm afraid that does happen occasionally, human nature being what it is.) **patch** will attempt to swap each hunk around before applying it. Rejects will come out in the swapped format. The **-R** switch will not work with ed diff scripts because there is too little information to reconstruct the reverse operation.

If the first hunk of a patch fails, *patch* will reverse the hunk to see if it can be applied that way. If it can, you will be asked if you want to have the **-R** switch set. If it can't, the patch will continue to be applied normally. (Note: this method cannot detect a reversed patch if it is a normal diff and if the first command is an append (i.e. it should have been a delete) since appends always succeed, due to the fact that a null context will match anywhere. Luckily, most patches add or change lines rather than delete them, so most reversed normal diffs will begin with a delete, which will fail, triggering the heuristic.)

-s makes *patch* do its work silently, unless an error occurs. (This option is not supported by XPG4 version.)

-S causes *patch* to ignore this patch from the patch file, but continue on looking for the next patch in the file. Thus

```
patch -S + -S + <patchfile
```

will ignore the first and second of three patches. (This option is not supported by XPG4 version.)

-v causes *patch* to print out it's revision header and patch level. (This option is not supported by XPG4 version.)

-x *number*
sets internal debugging flags, and is of interest only to *patch* patchers. (This option is not supported by XPG4 version.)

p

EXTERNAL INFLUENCES

Environment Variables

UNIX95 determines which version of patch is used. If this variable is set, patch exhibits XPG4 behaviour.

RETURN VALUE

The following exit values are returned for XPG4 version:

0	Successful completion.
1	One or more lines were written to a reject file.
>1	An error occurred.

For non-XPG4 version exit values vary as follows:

0	Successful completion or one or more lines were written to a reject file.
1	An error occurred.

DIAGNOSTICS

Too many to list here, but generally indicative that *patch* couldn't parse your patch file.

The message "**Hmm...**" indicates that there is unprocessed text in the patch file and that *patch* is attempting to intuit whether there is a patch in that text and, if so, what kind of patch it is.

Note that only few diagnostic messages are printed for XPG4 version, since it does not support verbose option.

WARNINGS

patch cannot tell if the line numbers are off in an ed script, and can only detect bad line numbers in a normal diff when it finds a "**change**" or a "**delete**" command. A context diff using fuzz factor 3 may have the same problem. Until a suitable interactive interface is added, you should probably do a context diff in these cases to see if the changes made sense. Of course, compiling without errors is a pretty good indication that the patch worked, but not always.

patch usually produces the correct results, even when it has to do a lot of guessing. However, the results are guaranteed to be correct only when the patch is applied to exactly the same version of the file that the patch was generated from.

The result obtained from the XPG4 options **-c**, **-e**, **-n** which forces the patch command to interpret the diff file either as a context diff or as an ed script or as a normal diff respectively is unspecified. For example, if one forces the patch command to treat the context diff file as an ed script, the result is unspecified. The same is true if one forces patch to treat an ed script as a context file and so on.. When a diff is forced with the above options, the diff file is searched for patterns that are specific to that type of diff file. If the diff file is not what was specified by the option, the file is checked for ed commands. If ed commands are present in the diff file, then the file is assumed to be an ed_diff file and the patch proceeds.

FILES

/var/tmp/patch*

SEE ALSO

diff(1), ed(1).

NOTES FOR PATCH SENDERS

There are several things you should bear in mind if you are going to be sending out patches. First, you can save people a lot of grief by keeping a patchlevel.h file which is patched to increment the patch level as the first diff in the patch file you send out. If you put a Prereq: line in with the patch, it won't let them apply patches out of order without some warning. Second, make sure you've specified the filenames right, either in a context diff header, or with an Index: line. If you are patching something in a subdirectory, be sure to tell the patch user to specify a **-p** switch as needed. Third, you can create a file by sending out a diff that compares a null file to the file you want to create. This will only work if the file you want to create doesn't exist already in the target directory. Fourth, take care not to send out reversed patches, since it makes people wonder whether they already applied the patch. Fifth, while you may be able to get away with putting 582 diff listings into one file, it is probably wiser to group related patches into separate files in case something goes haywire.

BUGS

Could be smarter about partial matches, excessively deviant offsets and swapped code, but that would take an extra pass.

If code has been duplicated (for instance with #ifdef OLDCODE ... #else ... #endif), *patch* is incapable of patching both versions, and, if it works at all, will likely patch the wrong one, and tell you that it succeeded to boot.

If you apply a patch you've already applied, *patch* will think it is a reversed patch, and offer to un-apply the patch. This could be construed as a feature.

One more thing to be noted with respect to XPG4 version of *patch*. If you are using multiple patches for different files, group patches that have to be applied to a single file. Otherwise, intermediate versions of the previous patches of a file will not be used for the current patch.

STANDARDS CONFORMANCE

patch: XPG4

NAME

pathalias - electronic address router

SYNOPSIS

pathalias [**-ivcDf**] [**-l** *host*] [**-d** *link*] [**-t** *link*] [*files*]

DESCRIPTION

pathalias computes the shortest paths and corresponding routes from one host (computer system) to all other known, reachable hosts. **pathalias** reads host-to-host connectivity information on standard input or in the named *files*, and writes a list of host-route pairs on the standard output.

Options

pathalias recognizes the following options and command-line arguments:

-i	Ignore case: map all host names to lowercase. By default, case is significant.
-c	Print costs. Print the path cost (see below) before each host-route pair.
-v	Verbose. Report some statistics on the standard error output.
-D	Terminal domains. Domain members are terminal.
-f	First hop cost. The printed cost is the cost to the first relay in a path instead of the cost of the path itself; implies (and overrides) the **-c** option.
-l *host*	Set local host name to *host*. By default, **pathalias** discovers the local host name in a system-dependent way.
-d *link*	Declare a dead link, host, or network (see below). If *link* is of the form **host1!host2**, the link from host1 to host2 is treated as an extremely high cost (i.e., **DEAD**) link. If *link* is a single host name, that host is treated as dead and is used as an intermediate host of last resort on any path. If *link* is a network name, the network requires a gateway.
-t *link*	Trace input for link, host, or network on the standard error output. The form of *link* is as above.

The public domain version of **pathalias** includes two undocumented options that are briefly described in the Special Options section below.

Input Format

A line beginning with white space continues the preceding line. Anything following **#** on an input line is ignored.

A list of host-to-host connections consists of a “from” host in column 1, followed by white space, followed by a comma-separated list of “to” hosts, called *links*. A link may be preceded or followed by a network character to use in the route. Valid network characters are **!** (default), **@**, **:**, and **%**. A link (and network character, if present) may be followed by a “cost” enclosed in parentheses. Costs can be arbitrary arithmetic expressions involving numbers, parentheses, **+**, **-**, *****, and **/**. Negative costs are prohibited. The following symbolic costs are recognized:

LOCAL	25	(local-area network connection)
DEDICATED	100	(high speed dedicated link)
DIRECT	200	(toll-free call)
DEMAND	300	(long-distance call)
HOURLY	500	(hourly poll)
EVENING	2000	(time restricted call)
DAILY	5000	(daily poll, also called POLLED)
WEEKLY	30000	(irregular poll)

In addition, **DEAD** is a very large number (effectively infinite), and **HIGH** and **LOW** are –5 and +5 respectively, for baud-rate or quality bonuses/penalties, and **FAST** is -80, for adjusting costs of links that use high-speed (9.6 Kbaud or more) modems. These symbolic costs represent an imperfect measure of bandwidth, monetary cost, and frequency of connections. For most mail traffic, it is important to minimize the number of hosts in a route, thus, *e.g.*, **HOURLY** is far greater than **DAILY** divided by 24. If no cost is given, a default of 4000 is used.

For the most part, arithmetic expressions that mix symbolic constants other than **HIGH**, **LOW**, and **FAST** make no sense. For example, if a host calls a local neighbor whenever there is work, and additionally polls every evening, the cost is **DIRECT**, *not* **DIRECT+EVENING**.

Some examples:

```
down         princeton!(DEDICATED), tilt,
             %thrash(LOCAL)
princeton    topaz!(DEMAND+LOW)
topaz        @rutgers(LOCAL+1)
```

If a link is encountered more than once, the least-cost occurrence dictates the cost and network character. Links are treated as bidirectional but asymmetric: for each link declared in the input, a **DEAD** reverse link is assumed.

If the "to" host in a link is surrounded by angle brackets, the link is considered *terminal*, and further links beyond this one are heavily penalized. For example, with input

```
seismo       <research>(10), research(100), ihnp4(10)
research     allegra(10)
ihnp4        allegra(50)
```

the path from **seismo** to **research** is direct, but the path from **seismo** to **allegra** uses **ihnp4** as a relay; not **research**.

The set of names by which a host is known by its neighbors is called its *aliases*. Aliases are declared as follows:

name=*alias*, *alias* ...

The name used in the route to or through aliased hosts is the name by which the host is known to its predecessor in the route.

Fully connected networks, such as the ARPANET or a local-area network, are declared as follows:

net = {*host, host, ...***}**

The host-list can be preceded or followed by a routing character (**!** by default), and can be followed by a cost (4000 by default). The network name is optional; if not given, **pathalias** creates one.

```
etherhosts = {rahway, milan, joliet}!(LOCAL)
ringhosts = @{gimli, alida, almo}(DEDICATED)
= {etherhosts, ringhosts}(0)
```

The routing character used in a route to a network member is the one encountered when "entering" the network. See also the sections on *gateways* and *domains*.

p

Connection data can be given while hiding host names by declaring

private {*host, host, ...***}**

pathalias does not generate routes for private hosts, but can produce routes through them. The scope of a private declaration extends from the declaration to the end of the input file in which it appears, or to a private declaration with an empty host list, whichever comes first. The latter scope rule offers a way to retain the semantics of private declarations when reading from the standard input.

Dead hosts, links, or networks can be presented in the input stream by declaring

dead {*arg*, ...**}**

where *arg* has the same form as the argument to the **-d** option.

To force a specific cost for a link, delete all prior declarations with

delete {*host1***!***host2***}**

and declare the link as desired. To delete a host and all its links, use

delete {*host***}**

Error diagnostics refer to the file in which the error was found. To alter the file name, use

file {*filename***}**

Fine-tuning is possible by adjusting the weights of all links from a given host, as in

adjust {*host1*, *host-2***(LOW)**, *host3***(-1)}**

If no cost is given, a default of 4000 is used.

Input from compressed (and uncompressed) files can be piped into **pathalias** with the following script.

```
for i in $*; do
        case $i in
        *.Z)    echo "file {‘expr $i : ’).Z’‘}
                zcat $i ;;
        *)      echo "file {$i}"
                cat $i ;;
        esac
        echo "private {}"
done
```

Output Format

A list of host-route pairs is written to the standard output, where route is a string appropriate for use with **printf()** (see *printf*(3S)), *such* as

```
rutgers     princeton!topaz!%s@rutgers
```

The **%s** in the route string should be replaced by the user name at the destination host (this task is normally performed by a mailer).

Except for *domains* (see below), the name of a network is never used in routes. Thus, in the earlier example, the path from **rahway** to **milan** would be **milan!%s**, not **etherhosts!milan!%s**.

Gateways

A network is represented by a pseudo-host and a set of network members. Links from the members to the network have the weight given in the input, while the cost from the network to the members is zero. If a network is declared dead, the member-to-network links are marked dead, which effectively prohibits access to the network from its members.

However, if the input also shows an explicit link from any host to the network, then that host can be used as a gateway (in particular, the gateway need not be a network member).

For example, suppose **CSNET** is declared dead on the command line and the input contains

```
CSNET = {...}
csnet-relay     CSNET
```

Then routes to **CSNET** hosts will use **csnet-relay** as a gateway.

Domains

A network whose name begins with **.** is called a domain. Domains are presumed to require gateways; i.e., they are **DEAD**. The route given by a path through a domain is similar to that for a network, but here the domain name is appended to the end of the name of the next host. Subdomains are permitted. For example:

```
harvard     .EDU            # harvard is gateway to .EDU domain
.EDU        = {.BERKELEY, .UMICH}
.BERKELEY   = {ernie}
```

yields

```
ernie       ...!harvard!ernie.BERKELEY.EDU!%s
```

Output is given for the nearest gateway to a domain; e.g., the example above gives

```
.EDU          ...!harvard!%s
```

Output is given for a subdomain if it has a different route than its parent domain, or if all its ancestor domains are private.

If the **-D** option is given on the command line, **pathalias** treats a link from a domain to a host member of that domain as terminal. This property extends to host members of subdomains, etc., and discourages routes that use any domain member as a relay.

Special Options

The public domain version of **pathalias** includes two undocumented options that rewrite named files with intermediate data of limited usage. Here are brief descriptions:

-g *file* Dump graph edges into *file* in the form *host***>***host* for simple connections and *host***@**<tab>*host* for network connections (from hosts to networks only).

-s *file* Dump shortest path tree into *file* in the form *host*<tab>[**@**]*host*[**!**]**(***cost***)**, including both connections from hosts to networks and from networks to hosts. This data may be useful for generating lists of one-way connections.

BUGS

The **-i** option should be the default.

The order of arguments is significant. In particular, **-i** and **-t** should appear early in the command line.

pathalias can generate hybrid (i.e., ambiguous) routes, which are abhorrent and most certainly should not be given as examples in a manual entry. Experienced mappers largely shun '@' when preparing input; this is historical, but also reflects UUCP's simplistic syntax for source routes.

Mixed-mode paths are ambiguous because the precedence of @ versus ! is not specified, varies from host to host, and is configurable. They should rarely be used.

Multiple @s in routes are prohibited by many mailers. To circumvent this restriction, mailers instead support the "magic %" rule, described below. When **pathalias** would otherwise generate a path containing multiple @s, it instead generates a path to which the "magic %" rule can be correctly applied.

Basically, the "magic %" rule for generating paths is "when constructing a path that would require multiple @s, replace all but the right-most @ with **%**.

When a mailer that supports the "magic %" rule receives a message that was routed to it via ..path..@host, it processes the route as follows:

1. Remove the trailing "@host" part of the route.
2. Examine the remaining route from right to left, proceeding to the next step when a "!" is seen. If a '%' is seen, change it to '@' and proceed to the next step immediately.
3. Continue processing the message using the modified route. If the modified route contains both '!' and '@' characters, the exact selection of the next host to route the message is governed by the specific precedence of '!' vs. '@' at this host.

   For example, if a host, **jazz.nonesuch.com**, received a message with a path **foo!joe%castle.hrh.gov.uk@jazz.nonesuch.com**, the mailer would convert the path to **foo!joe@castle.hrh.gov.uk**, and then forward it appropriately. If the host were configured such that '!' were of higher precedence than '@', the message would be forwarded to host **foo**, which would then deliver the message to **joe@castle.hrh.gov.uk**. If instead **jazz.nonesuch.com** were configured with '@' as higher in precedence, it would forward the message to host **castle.hrh.gov.uk**, which would then deliver it to **foo!joe**. (Clearly, **pathalias** could only correctly generate such a path if it knew the precedence at host **jazz.nonesuch.com**; since the database does not contain that information, such paths from **pathalias** should be viewed with suspicion.)

The **-D** option suppresses insignificant routes to domain members. This is benign, perhaps even beneficial, but confusing, since the behavior is undocumented and somewhat unpredictable.

AUTHOR

pathalias was developed by Peter Honeyman and Steven M. Bellovin.

FILES

newsgroup comp.mail.maps Likely location of some input files.

SEE ALSO

P.Honeyman and S.M. Bellovin, *PATHALIAS or The Care and Feeding of Relative Addresses*, in *Proc. Summer USENIX Conf.*, Atlanta, 1986.

p

NAME

pathchk - check path names

SYNOPSIS

pathchk [**-p**] *pathname*...

DESCRIPTION

The **pathchk** command checks that one or more path names are valid and portable. By default, the **pathchk** command checks each component of each path name specified by the *pathname* parameter based on the underlying file system. An error message is written for each path name operand that:

- is longer than that allowed by the system.
- contains any component longer than that allowed by the system.
- contains any component in a directory that is not searchable.
- contains any character in any component that is not valid in its containing directory.

It is not considered an error if one or more components of a path name do not exist, as long as a file matching the path name specified by the *pathname* parameter could be created that does not violate any of the checks above.

More extensive portability checks are performed when the **-p** flag is specified.

Options

The **pathchk** command supports the following option:

-p Performs path name checks based on POSIX portability standards instead of the underlying file system. An error message is written for each path name that:

- is longer than **_POSIX_PATH_MAX** bytes.
- contains any component longer than **_POSIX_NAME_MAX** bytes.
- contains any character in any component that is not in the portable file name character set.

EXTERNAL INFLUENCES

Environment Variables

LC_MESSAGES determines the language in which messages are displayed.

If **LC_MESSAGES** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**.

If any internationalization variable contains an invalid setting, **pathchk** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single-byte and multi-byte character code sets are supported.

RETURN VALUE

Upon successful completion, **pathchk** returns zero; otherwise it returns nonzero to indicate an error.

EXAMPLES

To check the validity and portability of the

```
/users/mary/work/tempfiles
```

path name on your system, use:

```
pathchk /users/mary/work/tempfiles
```

To check the validity and portability of the

```
/users/mary/temp
```

path name for POSIX standards, use:

```
pathchk -p /users/mary/temp
```

STANDARDS CONFORMANCE

pathchk: XPG4, POSIX.2

NAME

pax - Extracts, writes, and lists archive files; copies files and directory hierarchies

SYNOPSIS

Listing Member Files of Archived Files

pax [**-cdnv**] [**-f** *archive*] [**-s** *replstr*] ... [*pattern* ...]

Extracting Archive Files

pax -r [**-cdiknuvy**] [**-f** *archive*] [**-p** *string*] ... [**-s** *replstr*] ... [*pattern* ...]

Writing Archive Files

pax -w [**-adituvXy**] [**-b** *blocking*] [**-f** *archive*] [**-s** *replstr*] ... [**-x** *format*] [*file* ...]

Copying Files

pax -r -w [**-diklntuvXy**] [**-p** *string*] ... [**-s** *replstr*] ... [*file* ...] *directory*

DESCRIPTION

The **pax** command extracts and writes member files of archive files; writes lists of the member files of archives; and copies directory hierarchies. The **-r** and **-w** flags specify the archive operation performed by the **pax** command.

The *pattern* argument specifies a pattern that matches one or more paths of archive members. A **** (backslash) character is not recognized in the *pattern* argument and it prevents the subsequent character from having any special meaning. If no *pattern* argument is specified, all members are selected in the archive.

If a *pattern* argument is specified, but no archive members are found that match the pattern specified, the **pax** command detects the error, exits with a nonzero exit status, and writes a diagnostic message.

The **pax** command can read both **tar** and **cpio** archives. In the case of **cpio**, this means that **pax** can read ASCII archives (which are created with **cpio -c**) and binary archives (which are created without the **-c** flag). The supported archive formats are automatically detected on input.

pax can also write archives that **tar** and **cpio** can read; by default, **pax** writes archives in the **ustar** extended **tar** interchange format. **pax** also writes ASCII **cpio** archives; use the **-x cpio** flag to specify this extended **cpio** output format.

Options

-a Appends files to the end of the archive. Certain devices might not support appending.

-b *blocking* Specifies the block size for output to be the positive decimal integer of bytes specified by the *blocking* argument. The block size value cannot exceed 32,256. Blocking is automatically determined on input.

Do not specify a value for the *blocking* argument larger than 32768. Default blocking when creating archives depends on the archive format. (See the **-x** flag description.)

-c Matches all file or archive members except those specified by the *pattern* or *file* arguments.

-d Causes directories being copied or archived, or archived directories being extracted, to match only the directory or archived directory itself and not the contents of the directory or archived directory.

-f *archive* Specifies the path of an archive file to be used instead of standard input (when the **-w** flag is not specified) or the standard output (when the **-w** flag is specified but the **-r** flag is not). When specified with the **-a** flag, any files written to the archive are appended to the end of the archive.

-i Renames files or archives interactively. For each archive member that matches the *pattern* argument or file that matches a *file* argument, a prompt is written to the terminal (**/dev/tty**) that contains the name of a file or archive member. A line is then read from the terminal. If this line is empty, the file or archive member is skipped. If this line consists of a dot, the file or archive member is processed with no modification to its name. Otherwise, its name is replaced with the contents of the line. The **pax** command immediately exits with a nonzero exit status if an End-of-File is encountered when reading a response or if it cannot read or write to the terminal.

-k Prevents the **pax** command from writing over existing files.

-l Links files when copying files. When both **-r** and **-w** are specified, hard links are established between the source and destination file hierarchies whenever possible.

-n Selects the first archive member that matches each *pattern* argument. No more than one archive member is matched for each pattern (although members of type directory will still match the file hierarchy rooted at that file).

-p *string* Specifies one or more file characteristics to be retained or discarded on extraction. The *string* argument consists of the characters **a**, **e**, **m**, **o**, and **p**. Multiple characteristics can be concatenated within the same string and multiple **-p** flags can be specified. The specification flags have the following meanings:

- **a** Does not retain file-access times.
- **e** Retains the user ID, group ID, access permission, access time, and modification time.
- **m** Does not retain file-modification times.
- **o** Retains the user ID and the group ID.
- **p** Retains the access permission.

Note that "retain" means that an attribute stored in the archive is given to the extracted file, subject to the permissions of the invoking process; otherwise, the attribute is determined as part of the normal file creation action.

If neither the **e** nor the **o** flag is specified, or the user ID and group ID are not retained, the **pax** command does not set the **S_ISUID** and **S_ISGID** bits of the access permission. If the retention of any of these items fails, the **pax** command writes a diagnostic message to standard error. Failure to retain any of the items affects the exit status, but does not cause the extracted file to be deleted. If specification flags are duplicated or conflict with each other, the ones given last take precedence. For example, if **-p eme** is specified, file-modification times are retained.

-r Reads an archive file from the standard input.

-s Modifies file-member or archive-member names specified by the *pattern* or *file* arguments according to the substitution expression *replstr*, using the syntax of the **ed** command. The substitution expression has the following format:

-s/*old***/***new***/**[**gp**]

where as in the **ed** command, *old* is a basic regular expression and *new* can contain an **&** (ampersand), **\***n* (*n* is a digit) back references, or subexpression matching. The *old* string can also contain newline characters.

Any nonnull character can be used as a delimiter (the **/** (slash) character is the delimiter in the previous format). Multiple **-s** flag expressions can be specified; the expressions are applied in the order specified, terminating with the first successful substitution. The optional trailing **g** character performs as in the **ed** command. The optional trailing **p** character causes successful substitutions to be written to the standard error. File-member or archive-member names that substitute to the empty string are ignored when reading and writing archives.

-t Causes the access times of the archived files to be the same as they were before being read by the **pax** command.

-u Ignores files that are older (having a less recent file modification time) than a preexisting file or archive member with the same name.

When extracting files (**-r** flag), an archive member with the same name as a file in the file system is extracted if the archive member is newer than the file.

When writing files to an archive file (**-w** flag), an archive member with the same name as a file in the file system is superseded if the file is newer than the archive member.

When copying files to a destination directory (**-rw** flags), the file in the destination hierarchy is replaced by the file in the source hierarchy or by a link to the file in the source hierarchy if the file in the source hierarchy is newer.

-v Writes information about the process. If neither the **-r** or **-w** flags are specified, the **-v** flag produces a verbose table of contents that resembles the output of **ls** -l**;** otherwise, archive-member pathnames are written to standard error.

-w Writes files to the standard output in the specified archive format.

-x *format* Specifies the output archive format. The **pax** command recognizes the following formats:

cpio	Extended **cpio** interchange format. The default blocking value for this format for character special archive files is 5120. Blocking values from 512 to 32,256 in increments of 512 are supported.
ustar	Extended **tar** interchange format. This is the default output archive format. The default blocking value for this format for character special archive files is 10240. Blocking values from 512 to 32,256 in increments of 512 are supported.

Any attempt to append to an archive file in a format different from the existing archive format causes the **pax** command to exit immediately with a nonzero exit status.

-X When traversing the file hierarchy specified by a pathname, the **pax** command does not descend into directories that have a different device ID.

-y Prompts interactively for the disposition of each file. Substitutions specified by **-s** flags are performed before you are prompted for disposition. An EOF marker or an input line starting with the character **q** causes **pax** to exit. Otherwise, an input line starting with anything other than **y** causes the file to be ignored. This flag cannot be used in conjunction with the **-i** flag.

Option Interaction and Processing Order

The flags that operate on the names of files or archive members (**-c**, **-i**, **-n**, **-s**, **-u**, and **-v**) interact as follows.

When extracting files (**-r** flag), archive members are selected, using the modified names, according to the user-specified pattern arguments as modified by the **-c**, **-n**, and **-u** flags. Then, any **-s** and **-i** flags modify, in that order, the names of the selected files. The **-v** flag writes the names resulting from these modifications.

When writing files to an archive file (**-w** flag), or when copying files, the files are selected according to the user-specified pathnames as modified by the **-n** and **-u** flags. Then, any **-s** and **-i** flags modify, in that order, the names resulting from these modifications. The **-v** flag writes the names resulting from these modifications.

If both the **-u** and **-n** flags are specified, the **pax** command does not consider a file selected unless it is newer than the file to which it is compared.

Listing Member Files of Archived Files

When neither the **-r** nor the **-w** flags are specified, the **pax** command writes the names of the members of the archive file read from the standard input, with pathnames matching the specified patterns, to the standard output. If a named file is a directory, the file hierarchy contained in the directory is also written. You can specify the **pax** command without the **-r** or **-w** flags with the **-c**, **-d**, **-f**, **-n**, **-s**, and **-v** flags, and with the *pattern* argument.

If neither the **-r** or **-w** flags are included, **pax** lists the contents of the specified archive, one file per line. **pax** lists hard link pathnames as follows:

*pathname***==***linkname*

pax lists symbolic link pathnames as follows:

*pathname***->***linkname*

In both of the preceding cases, *pathname* is the name of the file that is being extracted, and *linkname* is the name of a file that appeared earlier in the archive.

If the **-v** flag is specified, the listing of hard link pathnames is output in the *ls -l* command format.

Extracting Archive Files

When the **-r** flag is specified, but the **-w** flag is not, the **pax** command extracts the members of an archive file read from the standard input, and with pathnames matching the *pattern* argument if one is

p

specified. If an extracted file is a directory, the file hierarchy contained in the directory is also extracted. The extracted files are created relative to the current file hierarchy. The **-r** flag can be specified with the **-c**, **-d**, **-f**, **-n**, **-s**, and **-v** flags, and a *pattern* argument.

The access and modification times of the extracted files are the same as the archived files. The access permissions of the extracted files remain as archived unless affected by the user's default file creation mode. The **S_ISUID** and **S_ISGID** bits of the extracted files are cleared.

If intermediate directories are necessary to extract an archive member, the **pax** command creates the directories with access permissions set as the bitwise inclusive OR of the values of the **S_IRWXU**, **S_IRWXG**, and **S_IRWXO** options.

If the selected archive format supports the specification of linked files (both the **tar** and **cpio** formats do), it is an error if these files cannot be linked when the archive is extracted. **pax** informs you of the error and continues processing.

Writing Archive Files

When the **-w** flag is specified and the **-r** flag is not, the **pax** command writes the contents of the files specified by the *file* arguments to the standard output in an archive format. If no *file* arguments are specified, a list of files to copy, one per line, is read from the standard input. When the *file* argument specifies a directory, all of the files contained in the directory are written. The **-w** flag can be specified with the **-b**, **-d**, **-f**, **-i**, **-s**, **-t**, **-u**, **-v**, **-x**, and **-X** flags and with *file* arguments.

If **-w** is specified, but no files are specified, standard input is used. If neither **-f** or **-w** are specified, standard input must be an archive file.

Copying Files

When both the **-r** and **-w** flags are specified, the **pax** command copies the files specified by the *file* arguments to the destination directory specified by the *directory* argument. If no file arguments are specified, a list of files to copy, one per line, is read from the standard input. If a specified file is a directory, the file hierarchy contained in the directory is also copied. The **-r** and **-w** flags can be specified with the **-d**, **-i**, **-k**, **-l**, **-p**, **-n**, **-s**, **-t**, **-u**, **-v**, and **-X** flags and with the *file* arguments. A *directory* argument must be specified.

Copied files are the same as if they were written to an archive file and subsequently extracted, except that there may be hard links between the original and the copied files.

RETURN VALUE

The **pax** command returns a value of 0 (zero) if all files were successfully processed; otherwise, **pax** returns a value greater than 0 (zero).

EXAMPLES

To copy the contents of the current directory to the tape drive, enter:

```
pax -w -f /dev/rmt/0m  .
```

To copy the **olddir** directory hierarchy to **newdir** enter:

```
mkdir newdir
cd olddir
pax -rw olddir newdir
```

To read the archive **a.pax**, with all files rooted in the directory **/usr** in the archive extracted relative to the current directory, enter:

```
pax -r -s ',//*usr//*,,' -f a.pax
```

All of the preceding examples create archives in **tar** format.

The following pairs of commands demonstrate conversions from **cpio** and **tar** to **pax**. In all cases, the examples show comparable command-line usage rather than identical output formats. The **-x** flag can be specified to the **pax** commands shown here, producing archives to select specific output formats:

```
ls * | cpio -ocv
pax -wdv *

find /mydir -type f -print | cpio -oc
find /mydir -type f -print | pax -w
```

```
cpio -icdum < archive
pax -r < archive

(cd /fromdir;find . -print) | cpio -pdlum /todir
pax -rwl /fromdir /todir

tar cf archive *
pax -w -f archive *

tar xfv - < archive
pax -rv < archive

(cd /fromdir; tar cf - . ) | (cd /todir; tar xf -)
pax -rw /fromdir /todir
```

Notes

When you use the **-i** flag (interactively renames files) on files to which there are hard links, **pax** does *not* create hard links to the renamed files.

WARNINGS

Because of industry standards and interoperability goals, **pax** does not support the archival of files larger than 2GB or files that have user/group IDs greater than 60K. Files with user/group IDs greater than 60K are archived and restored under the user/group ID of the current process.

AUTHOR

pax was developed by Mark H. Colburn, OSF, and HP.

SEE ALSO

ed(1), tar(4).

STANDARDS CONFORMANCE

pax: XPG4, POSIX.2

This implementation of pax is based upon a *POSIX.2 draft* specification. HP intends to update pax to meet the final POSIX.2 Standard once it completes, and thus the pax implementation is likely to change in a future release of HP-UX, possibly in ways incompatible with the current implementation. HP recommends using the current implementation only if absolutely necessary.

NAME

pdclean - removes all jobs from the specified object

SYNOPSIS

pdclean -h

pdclean [**-c** *ObjectClass*] [**-m "***MessageText***"**] [**-x "***AttributeValuePairs***"**]
[**-X** *AttributesFileName*]
{ *ServerName* ...
| [*ServerName***:**]*PrinterName* ...
| [*ServerName***:**]*QueueName* ... }

DESCRIPTION

Use this administrative command, **pdclean**, to remove all jobs from the specified servers, logical printers, queues, or physical printers. If you are removing jobs from a server, any **job-retention-period** attribute values are ignored. For all other conditions, the **job-retention-period** is honored. If you are removing jobs from a logical printer, all jobs that have been submitted to it are removed from the queue. If you are removing jobs from a queue, all jobs that are contained in that queue are removed from the queue.

Jobs that are in the middle of printing are aborted, if possible.

HPDPS issues a confirmation message prior to cleaning the object, unless your environment variable **PD_CONFIRM_DELETE** has a value of **no**.

Options

Use the following options with the **pdclean** command:

-c *ObjectClass*
Specify the object class you want for this command. The *ObjectClass* can be one of the following:

printer (default)
queue
server

Within the valid classes, **server** is for a spooler or a supervisor, and **printer** is for a logical or physical printer.

This option is equivalent to the command-attribute **class**.

p

-h Display a command-specific help message containing information about command syntax and options. This option cannot be used with another option or with an attribute.

-m "*MessageText***"**
Specify the message that is to be associated with the *ObjectClass* specified: **printer**, **queue**, or **server**. You can use this message to indicate the reason why the **printer**, **queue**, or **server** is being cleaned or to provide other comments.

This option is equivalent to specifying the command-attribute **message**.

List this message by specifying **requested-attributes=message** with the **pdls** command.

-x "*AttributeValuePairs***"**
A single attribute string, consisting of one or more attribute-value pairs.

-X *AttributesFileName*
The name of a file containing attribute-value pairs that you want inserted at the current point in the command line. This option is equivalent to specifying the command-attribute **attributes**.

Command Attributes

You may specify these attributes in a **-x "***AttributeValuePairs***"** string or in an attributes file designated with the **-X** *AttributesFileName* option.

attributes=*AttributesFileName*
Cause the designated attributes file to be read.

class=*ObjectClass*
Specify the *ObjectClass* that you want for this command. Valid object class names for the **pdclean** command are: **printer**, **queue**, or **server**. Within the valid classes, **server** is for a spooler or supervisor, and **printer** is for a logical or physical printer.

message=" *MessageText* **"**

Specify the message you want associated with the **printer**, **queue**, or **server**. You can use this message to indicate the reason why the **printer**, **queue**, or **server** is being cleaned or to provide other comments.

You can list this message by specifying **requested-attributes=message** with the **pdls** command.

Arguments

Use the argument to specify the printer, queue, or server to clean. If you specify multiple objects, all must be of the same object class, and each must be separated by spaces.

You can use the following arguments with **pdclean** command:

[*ServerName*:]*PrinterName*

Specify which printer you want cleaned of jobs.

Cleaning a physical printer removes only those jobs that have been assigned to that physical printer.

Cleaning a logical printer removes all jobs that were submitted through that logical printer that have not yet been completed. They are removed from the queue associated with the logical printer. Any jobs currently printing are aborted if this is supported by the physical printer device.

[*ServerName*:]*QueueName*

Specify which queue you want cleaned of jobs.

Cleaning a queue deletes all jobs that reside in that queue. Any jobs currently printing will be aborted if this is supported by the physical printer device.

ServerName

Specify which server you want cleaned of jobs.

Cleaning a spooler deletes all jobs that have been submitted to any of the logical printers residing in that spooler including any retained jobs. Any jobs currently printing are aborted if this is supported by the physical printer device.

Cleaning a supervisor deletes all jobs that are assigned to any of the physical printers residing in that supervisor. Any jobs currently printing are aborted, if this is supported by the physical printer device.

EXAMPLES

Clean a Printer

To clean printer lj4si of all jobs, enter the command:

```
pdclean lj4si
```

Clean a Server

To clean server DSuper1, enter the command:

```
pdclean -c server DSuper1
```

Clean a Physical Printer

To clean printer PhysPrt2 of all jobs and leave a message enter the command:

```
pdclean -m "Printer PhysPrt2 is down for repairs" SUPER1:PhysPrt2
```

SEE ALSO

pdcreate(1), pddelete(1), pddisable(1), pdenable(1), pdls(1), pdmod(1), pdpause(1), pdpr(1), pdpromote(1), pdq(1), pdresubmit(1), pdresume(1), pdrm(1), pdset(1), pdshutdown(1)

STANDARDS CONFORMANCE

pdclean: POSIX 1387.4

NAME

pdcreate - creates print objects

SYNOPSIS

pdcreate -h

pdcreate [**-c** *ObjectClass*] [**-g**] [**-m "***MessageText***"**] [**-r** *RequestedAttributes*]
[**-s** *StyleName*] [**-x "***AttributeValuePairs***"**] [**-X** *AttributesFileName*]
{ *ServerName***:***InitialValueDocumentName* ...
| *ServerName***:***InitialValueJobName* ...
| *ServerName***:***PrinterName* ...
| *ServerName***:***QueueName* ... }

DESCRIPTION

Use this administrative command, **pdcreate**, to create print objects (except servers, logs, documents and jobs) and to set their attributes to specific values.

Note: Servers and logs are created when the server is started. Documents and jobs are created when the files are submitted for printing with the **pdpr** command.

You must submit the **pdcreate** command to the appropriate server. HPDPS spoolers and supervisors support different sets of object classes.

Spooler	**Supervisor**
document	**document**
initial-value-document	**job**
initial-value-job	**log**
job	**printer (physical)**
log	**server**
printer (logical)	
queue	
server	

You can use the **pdcreate** command to create objects for the following object classes:

- **printer** (logical and physical)
- **queue**
- **initial-value-job**
- **initial-value-document**

When you create a printer object, it remains in the disabled state so that print jobs cannot be accepted. You must issue the **pdenable** command to place the printers in an enabled state.

An object you create with the **pdcreate** command still exists even if the server in which it is contained is terminated normally (shutdown) or abnormally. A logical printer or queue returns to its previous state when its server is restarted. A physical printer attempts to return to its previous state when its server is restarted.

Options

pdcreate accepts the following options:

-c *ObjectClass*
Specify the object class you want for this command. The *ObjectClass* can be:

printer (default)
queue
initial-value-job
initial-value-document

You can only specify one class per command invocation. This option is equivalent to specifying the command-attribute **class**.

-g Turn off headings. This option is equivalent to specifying the command-attribute **headings=false**.

-h Display a command-specific help message containing information about command syntax and options. This option cannot be used with another option or with an attribute.

-m "*MessageText***"**
Specify the message that is to be associated with the *ObjectClass* that is being created. You may indicate the reason for creating the *ObjectClass* or provide other comments.

You can list this message by specifying **requested-attributes=***message* with the **pdls** command. This option is equivalent to specifying the command-attribute **message**.

-r *RequestedAttributes*
Specify the attribute values you want displayed for the specified objects. Values can be:

none (default)
brief
verbose

This option is equivalent to specifying the command-attribute **requested-attributes**.

-s *StyleName*
Specify the format in which you want the attributes displayed. *StyleName* can be one of:

column (default)
line

This option is equivalent to specifying the command-attribute **style**.

-x "*AttributeValuePairs***"**
A single attribute string, consisting of one or more attribute-value pairs.

-X *AttributesFileName*
The name of a file containing attribute-value pairs to be inserted at the current point in the command line. This option is equivalent to specifying the command-attribute **attributes**.

Command Attributes

You can specify these attributes in a **-x "***AttributeValuePairs***"** string or in an attributes file designated with the **-X** *AttributesFileName* option.

attributes=*AttributesFileName*
Cause the designated attributes file to be read.

class=*ObjectClass*
Specify the *ObjectClass* to be created. Valid object class names are specified in the **-c** option description. You can only specify one class per command invocation.

force=*Boolean*
Force the creation of an object. The value can be:

false (default)
true

If you (as an administrator) are authorized and the object already exists, the specified object replaces the existing object and no warning or error messages are returned.

If the designated object already exists and you do not specify **force=true**, an error is returned and the command is rejected.

headings=*Boolean*
Specify if you want headings displayed in the output. *Boolean* can be:

false
true (default)

message="*MessageText***"**
Specify the message that is to be associated with the *ObjectClass* that is being created. You can use this message to indicate the reason for creating this object or to provide other comments.

You can list this message by specifying **requested-attributes=message** with the **pdls** command.

requested-attributes=*AttributeType*
Specify which output attributes you want displayed. The command-attribute value can be:

none (default)
brief
verbose

p

style=*StyleName*
Specify the presentation format that you want for the displayed output. *StyleName* can be:

column (default)
line

Object Attributes

You can specify these attributes in a **-x "***AttributeValuePairs***"** string or in an attributes file designated with the **-X** *AttributesFileName* option.

You can specify any settable or specifiable attribute with the **pdcreate** command. You can specify an attribute only when creating the object.

Arguments

Use the argument to specify the object to create. If you specify multiple objects, each must be separated by spaces.

Note: Regardless of which object you are creating, you must use *ServerName***:** with the **pdcreate** command.

Valid argument values you can use are:

*ServerName***:***PrinterName*
*ServerName***:***QueueName*
*ServerName***:***InitialValueDocumentName*
*ServerName***:***InitialValueJobName*

EXAMPLES

Create a Queue

To create the queue **QUEUE1** on the spooler **SPOOL1**, enter the command:

```
pdcreate -c queue SPOOL1:QUEUE1
```

Create a Logical Printer

To create the logical printer **LogPrt1** on the spooler **SPOOL1** and specify the queue that the printer is to be associated with, enter the command:

```
pdcreate -x "associated-queue=QUEUE1" SPOOL1:LogPrt1
```

Create a Physical Printer

To create a physical printer **PhyPrt1** in the supervisor **SUPER1** and specify the queue that the printer is to be associated with, enter the command:

```
pdcreate -x "associated-queue=QUEUE1 printer-model=LaserJet3Si  \
     attachment-type=tcpip printer-internet-address=15.0.64.97"  \
     SUPER1:PhyPrt1
```

Create an Initial Value Object (IVO)

To create an **initial-value-document** called **LP3ivd**, enter the command:

```
pdcreate -c initial-value-document -m "Created 11/15/94" \
     -x "copy-count=1 document-format=postscript\
     sides=1 descriptor='IVD for LogPrt3'" SPOOL1:LP3ivd
```

To create an **initial-value-job** object called **MyJobTemplate** using the attributes specified in the attributes file named **MyJobAttributes**, enter the command:

```
pdcreate -c initial-value-job -X MyJobAttributes  \
     server3:MyJobTemplate
```

The server **server3** must be a spooler; if not, the command is rejected.

Create a Queue and its Notification Profile

To create a notification profile for **QUEUE1** in server **SPOOL1**, enter the command:

```
pdcreate -c queue -x \
     "notification-profile={event-identifiers=job-modified \
     queue-status-changed delivery-method=e-mail \
     event-comment='This is a modification of job or status event' \
        delivery-address=dave@cowboy locale=C}" SPOOL1:QUEUE1
```

SEE ALSO

pdclean(1), pddelete(1), pddisable(1), pdenable(1), pdls(1), pdmod(1), pdpause(1), pdpr(1), pdpromote(1), pdq(1), pdresubmit(1), pdresume(1), pdrm(1), pdset(1), pdshutdown(1), pd_att(5).

To view information about all supported attributes, see manpage *pd_att*(5). It contains a list of files by object from which you can select the attribute listing that you want.

STANDARDS CONFORMANCE

pdcreate: POSIX 1387.4

NAME

pddelete - deletes print objects

SYNOPSIS

pddelete -h

pddelete [**-c** *ObjectClass*] [**-m "***MessageText***"**] [**-x "***AttributeValuePairs***"**]
[**-X** *AttributesFileName*]
{ ServerName ...
| *ServerName***:***InitialValueDocumentName* ...
| *ServerName***:***InitialValueJobName* ...
| [*ServerName***:**]*PrinterName* ...
| [*ServerName***:**]*QueueName* ... }

pddelete -c job [**-m "***MessageText***"**] [**-r** *JobRetentionPeriod*]
[**-x "***AttributeValuePairs***"**] [**-X** *AttributesFileName*] *LocalJobId* ... | *GlobalJobId* ...

DESCRIPTION

Use this administrative command, **pddelete**, to permanently delete print objects from the printing system.

A confirmation message is issued before deleting objects, unless the **PD_CONFIRM_DELETE** environment variable is set to **no**.

Options

Use the following options with the **pddelete** command:

-c *ObjectClass*
Specify the object class that the command is to operate upon. The *ObjectClass* can be one of the following:

printer (default)
job
queue
initial-value-job
initial-value-document
server

This option is equivalent to specifying the command-attribute **class**.

p

-h Display a command-specific help message containing information about command syntax and options. This cannot be used with another option or an attribute.

-m "*MessageText***"**
Specify the message that is to be associated with the *ObjectClass* that is being deleted: **printer**, **job**, **queue**, **initial-value-job**, or **initial-value-document**. You can use this message to indicate the reason for deleting the **printer**, **job**, **queue**, **initial-value-job**, or **initial-value-document**, or to provide other comments.

If the command is to operate on a job with a non-zero **job-retention-period**, list this message by specifying **requested-attributes=job-message-from-administrator** with the **pdls** command. If the command is to operate on a **printer**, **queue**, **initial-value-job**, **initial-value-document**, or **job** with a **job-retention-period** of zero (0), this message is deleted with the object and cannot be retrieved.

If the **-m** is not specified, the message already stored with the object remains unchanged and is deleted as just described.

This option is equivalent to specifying the command-attribute **message**.

-r *JobRetentionPeriod*
Can only be used if object-class is **job**. The command will be rejected if used with any other class.

If the job currently has a retention-period or is currently retained, this option must be specified with a zero (0) value to delete the job. If not specified, the present retention-period for the job is used.

This option is equivalent to specifying the object-attribute **job-retention-period**.

-x "*AttributeValuePairs***"**
A single attribute string, consisting of one or more attribute-value pairs.

-X *AttributesFileName*
The name of a file containing attribute-value pairs you want inserted at the current point in the command line. This option is equivalent to specifying the command-attribute **attributes**.

Command Attributes

You may specify these attributes in a **-x "***AttributeValuePairs***"** string or in an attributes file designated with the **-X** *AttributesFileName* option.

attributes=*AttributesFileName*
Cause the designated attributes file to be read.

class=*ObjectClass*
Specify the *ObjectClass* that you want for this command. Valid object class names for the **pddelete** command are: **printer** (default), **job**, **queue**, **initial-value-job**, **initial-value-document**, and **server**.

message="*MessageText***"**
Specify the message that you want associated with the **printer**, **job**, **queue**, **initial-value-job**, or **initial-value-document**, that is being deleted. You can use this message to indicate the reason for deleting the **printer**, **job**, **queue**, **initial-value-job**, **initial-value-document**, or to provide other comments.

If the command is to operate on a **job** with a non-zero **job-retention-period**, you can list this message by specifying a value of **requested-attributes=job-message-from-administrator** with the **pdls** command. If the command is to operate on **printer**, **job**, **queue**, **initial-value-job**, **initial-value-document**, or **job** with job-retention-period of zero (0), this message is deleted with the object and cannot be retrieved.

If the **message** attribute is not specified, the message stored with the object remains unchanged and is deleted as just described.

Object Attribute

You can only use the object-attribute if object-class is **job**. This command will be rejected if used with any other class. You may specify this attribute in a **-x "***AttributeValuePairs***"** string or in an attributes file designated with the **-X** *AttributesFileName* option.

job-retention-period=*time*
If the job currently has a retention-period or is currently retained, this attribute must be specified with a zero (0) value to delete the job. If not specified, the present retention-period for the job is used.

p

Arguments

Use the argument to specify the object to delete. If you specify multiple objects, each must be separated by spaces.

You can use the following arguments with the **pddelete** command:

ServerName
You must remove all jobs contained within the named server before you can delete the server.

*ServerName***:***InitialValueDocumentName*
If any of the logical printers that reference this initial value document are enabled, you cannot delete this **initial value** document.

*ServerName***:***InitialValueJobName*
If any of the logical printers that reference this initial value **job** are enabled, you cannot delete this initial value **job**.

LocalJobID or *GlobalJobId*
A job object is deleted based on the specified value in its **job-retention-period** attribute.

[*ServerName***:**]*PrinterName*
All the jobs must be removed from physical printers. Physical and logical printers must be disabled before they can be deleted.

[*ServerName***:**]*QueueName*
You must disable all logical and physical printers associated with the queue before you can delete the queue. Before disabling physical printers, you must first remove all of the jobs from them. All of the jobs (including paused jobs) must be removed from the queue before the queue can be deleted.

EXAMPLES

Delete a job using a Local ID

To delete job 5, enter the command:

```
pddelete -c job 5
```

Delete Logical Printers

To delete logical printers lp11 and lp15, enter the command:

```
pddelete  lp11  lp15
```

Delete an IVO

To delete the initial-value-job object, **`IVJ_2`**, from the spooler, **`DivSPOOL2`**, enter the command:

```
pddelete -c initial-value-job  DivSPOOL2:IVJ_2
```

Delete a Server

To delete the server SPOOL1, enter the command:

```
pddelete -c server SPOOL1
```

SEE ALSO

pdclean(1), pdcreate(1), pddisable(1), pdenable(1), pdls(1), pdmod(1), pdpause(1), pdpr(1), pdpromote(1), pdq(1), pdresubmit(1), pdresume(1), pdrm(1), pdset(1), pdshutdown(1)

STANDARDS CONFORMANCE

`pddelete`: POSIX 1387.4

NAME

pddisable - stops printers from accepting jobs and logs from logging

SYNOPSIS

pddisable -h

pddisable [**-c** *ObjectClass*] [**-m "***MessageText***"**] [**-x "***AttributeValuePairs***"**]
[**-X** *AttributesFileName*]
{ *ServerName* ...
| *ServerName***:***LogName* ...
| [*ServerName***:**]*PrinterName* ...
| [*ServerName***:**]*QueueName* ... }

DESCRIPTION

Use this administrative command, **pddisable**, to stop physical printers or logical printers from accepting jobs, or to stop logs from logging data.

When you disable a printer, it does not accept jobs submitted with **pdpr** or **pdresubmit** commands. The printer still accepts other commands. All previously submitted jobs and currently-printing jobs continue unaffected.

You use the **pdenable** command to enable a printer to accept jobs again and to enable the logging function of a log again.

Note: The **pddisable** and **pdenable** commands control input, whereas **pdpause** and **pdresume** control output.

Options

Use the following options with the **pddisable** command:

-c *ObjectClass*
Specify the object class you want for this command. The *ObjectClass* can be one of the following:

printer (default)
log
queue
server

Within the valid classes, **queue** disables all logical printers associated with it, **printer** is for a logical or a physical printer, **server** is for a spooler or a supervisor and disables all printers residing in the server, and **log** is for error log. See the *HPDPS Administration Guide* for more detail on error logging.

This option is equivalent to specifying the command-attribute **class**.

-h Display a command-specific help message containing information about command syntax and options. This option cannot be used with another option or with an attribute.

-m "*MessageText***"**
Specify the message that is to be associated with the *ObjectClass* specified: **printer**, **log**, **queue**, or **server**. You can use this message to indicate the reason for disabling the **printer**, **log**, **queue**, or **server** or to provide other comments.

If the **-m** option is not specified, the message already stored with the object remains unchanged.

When the command is issued against a server, HPDPS propagates the message to the **message** attribute of the printers residing in that server. The **message** attribute for that server is not changed. When the command is issued against a queue, the message is propagated to the **message** attribute of the logical printers associated with the queue. The **message** attribute for the queue is not changed.

You can list this message by specifying **requested-attributes=message** with the **pdls** command.

This option is equivalent to specifying the command-attribute **message**.

-x "*AttributeValuePairs***"**
A single attribute string, consisting of one or more attribute-value pairs.

p

-X *AttributesFileName*
The name of a file containing attribute-value pairs. This option is equivalent to specifying the command-attribute **attributes**.

Command Attributes

You may specify these attributes in a **-x "***AttributeValuePairs***"** string or in an attributes file designated with the **-X** *AttributesFileName* option.

attributes=*AttributesFileName*
Cause the designated attributes file to be read.

class=*ObjectClass*
Specify the *ObjectClass* that you want for this command. Valid object class names for the **pddisable** command are: **printer** (default), **log**, **queue**, or **server**. Within the valid classes, **queue** disables all associated logical printers, **printer** is for a logical or a physical printer, **server** is for a spooler or a supervisor and disables all printers residing in the server, and **log** is for error log.

message="*MessageText***"**
Specify the message you want associated with the **printer**, **log**, **queue**, or **server**. You can use this message to indicate the reason for disabling the **printer**, **log**, **queue**, or **server** or to provide other comments.

If this attribute is not specified, the message already stored with the object remains unchanged.

When the command is issued against a server, HPDPS propagates the message to the **message** attribute of the printers residing in that server. The **message** attribute for that server is not changed. When the command is issued against a queue, the message is propagated to the **message** attribute of the logical printers associated with the queue. The **message** attribute for the queue is not changed.

You can list this message by specifying **requested-attributes=message** with the **pdls** command.

Object Attributes

There are no object attributes for this command.

Arguments

Use the argument to specify the object to disable. If you specify multiple objects, each must be separated by spaces.

p

You can use the following arguments with the **pddisable** command:

*ServerName***:***LogName*
Specify the log you want disabled. Disabling a log stops it from logging data.

[*ServerName***:**]*PrinterName*
Specify the printer you want disabled. Disabling a printer stops the acceptance of print requests. Any jobs currently assigned to a physical printer continue printing.

[*ServerName***:**]*QueueName*
Specify the queue you want disabled. Issuing the **pddisable** command against a queue disables all of the logical printers associated with that queue.

ServerName
Specify the server you want disabled. Issuing the **pddisable** command against a server disables all of printers residing in the server. When issued against a:

Spooler	HPDPS attempts to disable all of the logical printers residing in that server.
Supervisor	HPDPS attempts to disable all of the physical printers residing in that server.

EXAMPLES

Disable Printers

To disable printer lj4si (logical or physical) on server Serve1, enter one of these commands:

```
pddisable lj4si
pddisable Serve1:lj4si
```

The server name is not required.

To disable the physical printers contained in the supervisor SuprG1 and assign a message to the printers, enter the command:

```
pddisable -c server -m "Unavailable due to testing" SuprG1
```

To disable the logical printers associated with the queue production-q1 on server spoolera, enter one of the following commands:

```
pddisable -c queue production-q1
pddisable -c queue spoolera:production-q1
```

Again, the server name is not required.

SEE ALSO

pdclean(1), pdcreate(1), pddelete(1), pdenable(1), pdls(1), pdmod(1), pdpause(1), pdpr(1), pdpromote(1), pdq(1), pdresubmit(1), pdresume(1), pdrm(1), pdset(1), pdshutdown(1)

STANDARDS CONFORMANCE

pddisable: POSIX 1387.4

NAME

pdenable - enables printers to accept jobs and logs to log

SYNOPSIS

pdenable -h

pdenable [**-c** *ObjectClass*] [**-m** "*MessageText*"] [**-x** "*AttributeValuePairs*"] [**-X** *AttributesFileName*]
{ *ServerName* ...
  | [*ServerName*:]*LogName* ...
  | [*ServerName*:]*PrinterName* ...
  | [*ServerName*:]*QueueName* ... }

DESCRIPTION

Use this administrative command, **pdenable**, to enable the logging function of logs or to enable logical printers or physical printers to accept jobs.

To discontinue acceptance of print jobs, use the **pddisable** command. To discontinue logs from logging data, use the **pddisable** command.

Note: The **pdenable** and **pddisable** commands control input, whereas **pdpause** and **pdresume** control output.

Options

You can use the following options with the **pdenable** command:

-c *ObjectClass*
Specify the object class you want for this command. The *ObjectClass* can be one of the following:

printer (default)
log
queue
server

Within the valid classes, **queue** enables all associated logical printers, **printer** is for a logical or a physical printer, **server** is for a spooler or a supervisor and enables all printers residing in that server, and **log** is for error log. See the *HPDPS Administration Guide* for more detail on error logging.

p

This option is equivalent to specifying the command-attribute **class**.

-h Display a command-specific help message containing information about command syntax and options. This option cannot be used with another option or with an attribute.

-m "*MessageText*"
Specify the message that is to be associated with the **printer**, **log**, **queue**, or **server** that is being enabled. You can use this message to indicate the reason for enabling the **printer**, **log**, **queue**, or **server** or to provide other comments.

If the **-m** option is not specified, the message already stored with the **printer**, **log**, **queue**, or **server** remains unchanged.

When the command is issued against a **server**, HPDPS propagates the message to the **message** attribute of the printers residing in that server. The **message** attribute for that server is not changed. When the command is issued against a **queue**, the message is propagated to the **message** attribute of the logical printers associated with the queue. The message attribute for that queue is not changed.

You can list this message by specifying **requested-attributes=message** with the **pdls** command.

This option is equivalent to specifying the command-attribute **message**.

-x "*AttributeValuePairs*"
A single attribute string, consisting of one more attribute-value pairs.

-X *AttributesFileName*
The name of a file containing attribute-value pairs you want inserted at the current point in the command line. This option is equivalent to specifying the command-attribute **attributes**.

Command Attributes

You can specify these attributes in a **-x** **"***AttributeValuePairs***"** string or in an attributes file designated with the **-X** *AttributesFileName* option.

attributes=*AttributesFileName*
: Specifies the designated attributes file that HPDPS is to read and insert at the current point in the command line. This file contains attribute and value pairs that HPDPS uses to expand on the command being entered.

class=*ObjectClass*
: Specify the *ObjectClass* that you want for this command. Valid object class names for the **pdenable** command are: **printer** (default), **log**, **queue**, or **server**. Within the valid classes, **queue** enables all associated logical printers, **printer** is for a logical or a physical printer, **server** is for a spooler or a supervisor and enables all printers residing in that server and **log** is for error log.

message="*MessageText***"**
: Specify the message you want associated with the **printer**, **log**, **queue**, or **server** that is being enabled. You may use this message to indicate the reason for enabling the **printer**, **log**, **queue**, or **server** or to provide other comments. If the **message** attribute is not specified, the message already stored with the object remains unchanged.

  When the command is issued against a server, HPDPS propagates the message to the **message** attribute of the printers residing in that server. The **message** attribute for that server is not changed. When the command is issued against a queue, the message is propagated to the **message** attribute of the logical printers associated with the queue. The **message** attribute for that queue is not changed.

  You can list this message by specifying **requested-attributes=message** with the **pdls** command.

Object Attributes

There are no object attributes for this command.

Arguments

Use the arguments to identify the specific object you want to enable. If you specify multiple objects, each must be separated by spaces.

You can use the following arguments with the **pdenable** command:

[*ServerName***:**]*LogName*
: Specify the log you want enabled. Enabling a log allows it to begin accepting input (logging).

[*ServerName***:**]*PrinterName*
: Specify the printer you want enabled. When first created, printers are in the disabled state. Printers cannot be enabled unless they are associated with an existing queue. Any other objects referenced by the printer also must exist. These are for logical printers:

  initial-value-job objects
  initial-value-document objects

[*ServerName***:**]*QueueName*
: Specify the queue you want enabled. When the **pdenable** command is issued against a queue, HPDPS attempts to enable all of its associated logical printers.

ServerName
: Specify the server you want enabled. When the **pdenable** command is issued against a server, HPDPS attempts to enable all of its printers. When issued against a:

Spooler	HPDPS attempts to enable all of its logical printers.
Supervisor	HPDPS attempts to enable all of its physical printers.

When a server is initialized again after being shut down, the printers are either enabled or disabled. This depends on the state the printer was in when HPDPS was shut down and on ability of the server to communicate with its associated queue.

EXAMPLES

Enable Printers

To enable logical printers lj4si and lj5si on spooler SPOOL1, enter one of these commands:

```
pdenable  lj4si  lj5si
pdenable  SPOOL1:lj4si  SPOOL1:lj5si
```

The server name is not required.

To enable all physical printers contained in the supervisor SUPERG1, enter the command:

```
pdenable -c server SUPERG1
```

SEE ALSO

pdclean(1), pdcreate(1), pddelete(1), pddisable(1), pdls(1), pdmod(1), pdpause(1), pdpr(1), pdpromote(1), pdq(1), pdresubmit(1), pdresume(1), pdrm(1), pdset(1), pdshutdown(1)

STANDARDS CONFORMANCE

pdenable: POSIX 1387.4

NAME

pdls - lists selected attribute values

SYNOPSIS

pdls [**-c** *ObjectClass*] [**-f** **"***FilterCriteria***"**] [**-F**] [**-g**] [**-j**] [**-r** **"***RequestedAttributes***"**]
[**-s** *StyleName*] [**-U**] [**-x** **"***AttributeValuePairs***"**] [**-X** *AttributesFileName*]
[*ServerName...* |
*ServerName***:***InitialValueDocumentName...* |
*ServerName***:***InitialValueJobName...* |
*ServerName***:***LogName...* |
[*ServerName***:**]*PrinterName...* |
[*ServerName***:**]*QueueName...* |
LocalJobId[**.***DocNumber*]... |
GlobalJobId[**.***DocNumber*]...]

pdls -h

DESCRIPTION

You enter the **pdls** command to request that selected attribute values be displayed for one or more print jobs or other HPDPS objects.

- By default, a filter is created when listing jobs which only allows you to see your jobs. The jobs have a predefined value for the filter that is equal to the job attribute **user-name**; this value is your login identity when you submit a job.
- You add to this filter to further restrict the jobs for which information is to be returned.
- You must suppress the default filter by using the **-U** option or turn off all filtering with the **-F** option to see more than your own jobs.
- If you only specify the *ServerName* as the command argument, the attribute values for all objects belonging to the object class you specify are displayed.
- You can list the attribute values for specific jobs by using the local ID or the global ID. You must have submitted the job to use the local ID.

Note: There is a situation when you must use the global ID. If the client daemon responsible for mapping local ID to global ID is not available, HPDPS may not be able to tie the local ID to the global ID. If this situation exists, you must use the global ID, the name of the server (spooler), or an argument specification using global characters.

There is a possible situation that can occur if your administrator has set the PDIDTABLE environment variable to a low value, for example 10. You submit a series of jobs during a short time span such that the number of jobs you have in process is larger than the value set, say 14. You will have two jobs with the local IDs of 1, 2, 3, and 4. However, HPDPS no longer associates the first four jobs with a local ID because those local IDs have been given to the 11th, 12th, 13th, and 14th jobs. Therefore, you must use the global ID to take action for any of the first four jobs.

Options

pdls recognizes the following options:

-c *ObjectClass*
Specifies the object class of the object whose attributes you want listed. *ObjectClass* can be one of:

job (default)
document
initial-value-document
initial-value-job
log
printer
queue
server

This option is equivalent to the command-attribute **class**.

-f **"***FilterCriteria***"**
Specifies the filter criteria you want to use in selecting from the candidate objects. Among the candidate objects, only those matching the filter expression are returned. See the command-attribute

p

filter for filter expression details. Using this option is equivalent to specifying the command-attribute **filter**.

-F Turns off all filtering (both specified and default). See the **-U** option for suppressing only the default.

-g Turns off headings. Using this option is equivalent to specifying the command-attribute **headings=false**.

-h Displays a command-specific help message containing information about command syntax and options. This option cannot be used with another option or with an attribute.

-j Use this option to display only job attributes.

-r "*RequestedAttributes***"**
Specifies the group of attributes that you want displayed for the object class (see **-c**).

RequestedAttributes can be:

brief (default)
verbose
archive
"*AttributesList***"**
all
None

Within the valid classes, **archive** only displays settable and specifiable attributes. **all** lists all attributes listing first those that do not have values assigned and then those with values assigned.

These values are additive. This option is equivalent to the command-attribute **requested-attributes**.

-s *StyleName*
Specifies the format you want the attributes displayed in.

StyleName can be **column** (default) or **line**.

column is the default for **-r brief** and **-r verbose**.

line is the default for **-r all**, **-r "***AttributesList***"** and **-r archive**.

See the command-attribute **style** for detail explanation of the column and line formats. This option is equivalent to the command-attribute **style**.

p

-U Suppresses the default user-name filter.

-x "*AttributeValuePairs***"**
Specifies command attributes.

AttributeValuePairs consists of a single attribute string containing of one or more attribute-value pairs.

-X *AttributesFileName*
Specifies the designated attributes file to be read and inserted at the current point in the command line. This file contains attribute-value pairs that are used to expand on the command being entered. This option is equivalent to the command-attribute **attributes**.

Command Attributes

You can also specify these command attributes in a **-x "***AttributeValuePairs***"** string or in an attributes file designated with the **-X** *AttributesFileName* option.

attributes=*AttributesFileName*
Specifies the designated attributes file to be read and inserted at the current point in the command line. This file contains attribute-value pairs that are used to expand on the command being entered.

class=*ObjectClass*
Specifies the object class you want for this command. *ObjectClass* can be one of:

job (default)
document
log
printer
queue
initial-value-job

initial-value-document
server

filter="*FilterCriteria***"**

Specifies the selection criteria you want used to select a subset from the candidate objects (if you request attribute values for multiple objects). A filter is a logical expression consisting of relations of attributes to attribute values. Among the objects you specify, only objects whose attribute values match the filter expression are returned.

You can only use attributes for the object class you specify in the command (see the **-c** option or the **class** command attribute). The filter may contain an attribute other than one of those you are requesting.

FilterCriteria is a text string delimited by quotes. The filter syntax is one of the following:

1. A filter item consisting of an "*attribute operator value*".

   Table 1 shows the operators and the data formats they can be used with to separate the attribute and value.

   Table 1. Attribute Operators for Filters

Operation	Operator	Strings	Integers	Time Format
Equal	**==**	X	X	X
Match first part of a value	**=***	X	X	X
Match last part of a value	***=**	X	X	X
Match any part of a value, such as a substring	***=***	X	X	X
Attribute present (any value) (See Note 1)	**==***	X	X	X
Match approximately, for case-insensitive substring (See Notes 2 and 3)	**~=**	X	-	-
Match a value greater than that specified	**>**	-	X	X
Match a value less than that specified	**<**	-	X	X

   Note 1: Testing for attribute present returns true when the attribute has a value, not just when the attribute exists. A false value may be needed to satisfy the requirement such as using quotes, as long as the false value conforms to the general syntax.

   Note 2: An approximate match is when at least half of the target string, regardless of starting position, matches the filter value.

   Note 3: A case insensitive match is when the target string may have a mix of upper- and lowercase characters, but the character do match.

2. In the previous table the attribute present operation consists of an attribute name followed by the equality operator followed by an * in the place of an attribute value. For example:

   ```
   -f "printers-assigned==*"
   ```

   If the attribute has no value, the filter item is evaluated as false. It evaluates as true if the attribute has been assigned any value.

3. Each attribute in a filter item can be compared to only one attribute value. To compare an attribute to more than one value, or to filter on more than one attribute, separate the filter items with one of the following operators:

   && AND operator, as in: *filter-item* **&&** *filter-item*

   The expression evaluates to true only if both *filter-item*s evaluate to true.

   || OR operator, as in: *filter-item* **||** *filter-item*

   The expression evaluates to true if either of the *filter-item*s evaluate to true.

p

4. To evaluate a filter item as false, use the NOT operator before the filter item and enclose the filter item in parentheses.

   ! NOT operator, as in: **! (** *filter-item* **)**

   If *filter-item* evaluates to true, the expression is considered false, and vice versa.

5. When you use multiple logical operators in a filter, they are evaluated in an order of precedence. You can override the order of precedence by using parentheses (**( )**). See Table 2 for order of precedence.

Table 2 summarizes the filter syntax provided by the HPDPS command line interface. The operators are listed in the order of precedence from highest to lowest.

Table 2. Filter Syntax

Operator		Placement
( )	Parentheses	Around filter items
> <	Relational operators	Between attribute and value
==	Equality operators	Between attribute and value
=* *= *=* ~=	String matching	Between attribute and value
!	NOT Operator	Before (filter-item) only
&&	AND operator	Between two filter items
||	OR operator	Between two filter items

headings=*Boolean*
Specifies whether you want the output displayed with or without headings; *Boolean* can be **true** (default) or **false.** See the command-attribute **style** for examples.

message-count=*Number*
When you request the log-messages log attribute, this specifies the **Number** of previous messages you want to see starting from the last message logged. The value you can specify for **Number** can be an integer from 1 through 2147483647.

Use this command attribute in conjunction with the log object **log-messages** attribute to query for error log information. For example:

```
pdls -c log -r log-messages -x "message-count=4" SPOOL1:
```

displays the last four messages contained in the error log for server SPOOL1.

p

requested-attributes=*AttributeType*
Specifies the group of attributes you want displayed for the specified *ObjectClass*.

AttributeType can be: **brief** (default), **verbose**, **archive**, **"***AttributesList***"**, **all**, or **none**.

archive	displays only settable and specifiable attributes.
all	lists all attributes, listing first those that do not have values assigned and then those with values assigned.

The following list shows the **brief** and the **verbose** attributes that are displayed for the specified object. The **brief** attributes are a subset of the **verbose** attributes and are identified in the following list with a “B” to the left of the attribute name.

The attributes and their values are displayed in the specified order.

Document Attributes
B **document-sequence-number**
B **document-format**
  octet-count
  copy-count
  sides
  document-type
B **document-file-name**
Initial-Value-Document Attributes
B **initial-value-document-identifier**
B **associated-server**
B **logical-printers-ready**

B **copy-count**
B **sides**
B **document-format**

Initial-Value-Job Attributes

B **initial-value-job-identifier**
B **associated-server**
B **logical-printers-ready**
B **printer-locations-requested**
B **printer-models-requested**
job-retention-period

Job Attributes

B **job-client-id (local ID)**
B **job-identifier (global ID)**
B **job-name**
B **current-job-state**
job-state-reasons
B **intervening-job** (Only returned on the **pdls** and **pdq** commands.)
B **printer-name-requested** (Logical printer name to which the job was submitted.)
B **printers-assigned** (Physical printer name to which the job has been assigned, if such a scheduling decision has been made.)
total-job-octets (Sum of all printable files and copies.)
job-owner

Log Attributes

B **log-identifier**
B **log-type**
B **associated-server**
B **enabled**
log-size
log-wrap

Printer Attributes

B **printer-name**
B **printer-realization**
B **printer-state** (physical printers)
associated-server
printer-locations
printer-associated-printers (logical printers)
B **enabled**
B **associated-queue**

Queue Attributes

B **queue-name**
B **queue-state**
B **scheduler-ready**
B **associated-server**
logical-printer-assigned
physical-printer-assigned

Server Attributes

B **server-name**
B **server-state**
logical-printers-supported (for spoolers)
physical-printers-supported
B **server-type**
queues-supported

style=*StyleName*

The format in which you want the attributes displayed. *StyleName* may be either **column** (default) or **line.**

column is the default for:

requested-attributes=brief
requested-attributes=verbose

line is the default for:

```
requested-attributes=all
requested-attributes="AttributesList"

requested-attributes=archive
```

The following is an example of **column** style with headings for **brief** set of attributes for a **physical printer** object within a specified supervisor.

```
                                                Associated
Name          Realization     State    Enabled  Queue
--------      -----------     -----    -------  ----------
PhysPrt1      physical        Idle     True     Queue1
PhysPrt2      physical        Idle     True     Queue1
PhysPrt3      physical        Idle     True     Queue2
```

The following is an example of **column** style without headings for **brief** set of attributes for a **physical printer** object within a specified supervisor.

```
PhysPrt1      physical        Idle     True  Queue1
PhysPrt2      physcial        Idle     True  Queue1
PhysPrt3      physical        Idle     True  Queue2
```

The following is an example of **line** style with headings for **brief** set of attributes for a **logical printer** object within a specified spooler that contains a single logical printer.

```
LogPrt1: printer-name           =  LogPrt1
LogPrt1: printer-realization    =  logical
LogPrt1: printer-enabled        =  true
LogPrt1: associated-queue       =  Queue1
```

The following is an example of **line** style without headings for **brief** set of attributes for a **logical printer** object within a specified spooler that contains a single logical printer.

```
printer-name                  =  LogPrt1
printer-realization           =  logical
printer-enabled               =  true
associated-queue              =  Queue1
```

Object Attributes

There are no object attributes for this command.

p

Arguments

You use the argument value to identify the specific object you want the attributes displayed for. If you specify multiple objects, each must be separated by spaces.

You can use the following argument values with the **pdls** command:

ServerName
ServerName:*InitialValueDocumentName*
ServerName:*InitialValueJobName*
ServerName:*LogName*
[*ServerName*:]*PrinterName*
[*ServerName*:]*QueueName*
LocalJobId[.*DocNumber*]
GlobalJobId[.*DocNumber*]

Note: When you only specify *ServerName* as the argument of the command (without an object name), the attribute values are returned for all of the objects within the object class you specify for that server that meet the filter criteria.

EXAMPLES

Check on a Document

- To find out as much as you can about the first document in a job; if the local ID is 13, enter the command:

```
pdls -r all 13.1
```

  This will provide information about the job and the first document.

- To find the minimum about the second document in a job; if the local ID is 13, enter the command:

  ```
  pdls -c document 13.2
  ```

  This will provide information only about the second document.

Determine Document Formats Supported by Given Printers

- To find out the document formats supported by the logical printers in SPOOL1, enter one of the following commands:

  ```
  pdls -c printer -r document-format-supported -s line SPOOL
  
  pdls -x "class=printer
       requested-attributes=document-format-supported
       style=line" SPOOL1
  ```

  HPDPS will respond with information that looks something like this:

  ```
  LogPrt1:  document-formats-supported=document-format-ASCII
                                       document-format-postscript
  
  LogPrt2:  document-formats-supported=document-format-ASCII
                                       document-format-HPL
                                       document-format-postscript
  ```

Using a Filter

- To determine the logical printers, physical printers, and queues that support selected attributes on any server, enter one of the following commands:

  ```
  pdls -c printer -f "content-orientation-supported==landscape &&
       sides-supported==2"
       -r printer-realization,associated-queue "*:"
  
  pdls -x "class=printer
       filter= content-orientation-supported==landscape &&
       sides-supported==2"
       requested-attributes=printer-realization associated-queue"
       "*:"
  ```

  The system will return information that looks something like this:

  ```
  LogPrtHi:  Printer-name          =  LogPrtHi
  LogPrtHi:  Printer-realization   =  logical
  LogPrtHi:  associated-queue      =  HiResQ
  LogPrtLo:  Printer-name          =  LogPrtLo
  LogPrtLo:  Printer-realization   =  logical
  LogPrtLo:  associated-queue      =  LoResQ
  PhysPrt1:  Printer-name          =  PhysPrt1
  PhysPrt1:  Printer-realization   =  physical
  PhysPrt1:  associated-queue      =  HiResQ
  PhysPrt2:  Printer-name          =  PhysPrt2
  PhysPrt2:  Printer-realization   =  physical
  PhysPrt2:  associated-queue      =  LoResQ
  ```

- To list all jobs that are owned by Smith, enter one of the following commands:

  ```
  pdls -f "job-owner==Smith" -U SPOOL1
  pdls -f "j-ow==Smith" -U SPOOL1
  ```

- To list all job that are not owned by Smith, enter the following command:

  ```
  pdls -f !("job-owner==Smith") -U SPOOL1
  ```

- To list jobs owned by someone with a given substring in their name (substring matching) use one of the following filters with the **pdls** command:

  ```
  pdls -U -f "job-owner=*Jones" SPOOL1: # Initial string match
  pdls -U -f "job-owner*=*one"  SPOOL1: # Any substring match
  pdls -U -f "job-owner*=nes"   SPOOL1: # Final string match
  ```

p

All of these will return jobs owned by "Jones".

- To list jobs owned by all users with a name close to "Jones" (approximate match), enter the command:

```
pdls -U -f "job-owner˜=jones" SPOOL1:
```

- To list all jobs which have requested more than one copy and that have been assigned to the **lj4pp** physical printer, enter the command:

```
pdls -U -f "copy-count>1 && printer-assigned==lj4pp" "*:"
```

To Determine the Server Associated with a Printer

- To query for the name of the spooler containing logical printer **LogPrt1**, enter one of the following commands:

```
pdls -c printer -r associated-server LogPrt1

pdls -x "class=printer requested-attributes=associated-server"
     LogPrt1
```

HPDPS returns information similar to the following:

```
LogPrt1:  associated-server=SPOOL1
```

To Determine the Attributes Specified in an IVO

- To query for the attributes specified in the initial value document **spl7ivd** contained in **SPOOL7**, enter one of the following commands:

```
pdls -c initial-value-document -r all SPOOL7:spl7ivd

pdls -x "class=initial-value-document requested-attributes=all"
     SPOOL17:spl17ivd
```

HPDPS returns information similar to the following:

```
spl7ivd:  initial-value-document-identifier = spl7ivd
spl7ivd:  associated-server                 = SPOOL7
spl7ivd:  object-class                      = initial-value-document
spl7ivd:  copy-count                        = 2
spl7ivd:  document-format                   = line-data
spl7ivd:  descriptor                        = "IVD for SPOOL7"
```

p

List Job Attributes

- To list all the attributes of jobs 10 and 12, enter the command:

```
pdls -r all 10 12
```

- To list just the job attributes of jobs 10 and 12, enter the command:

```
pdls -j 10 12
```

A Combined Example

In the following examples, which show printers as the command arguments, assume that the logical printer named **DepartPrt** is your default logical printer as defined in your **PDPRINTER** environment variable and that it sends input to the queue associated with the physical printers **LaserJet5** and **3825X**.

List Status of All Jobs

- To list the status of all jobs you have submitted to the logical printer **DepartPrt**, enter the command:

```
pdls -f "printer-name-requested==DepartPrt" SPOOL1:
```

The following is displayed:

```
                               Current    Intervening Printer   Printer
Job ID               Name        State      Jobs        Requested Assigned
--- -------------- ---------- --------- ----------- --------- --------
4   SplX: 1099222204 Mthly-report processing 0           DepartPrt 3825X
5   SplX: 1114222205 Test-report  processing 0           DepartPrt LaserJet3
6   SplX: 1224222206 Trip-report  pending    2           DepartPrt
1   SplX: 0988222201 Dept-memo12  retained               DepartPrt
```

List Status of All Pending Jobs

To list job status of all jobs that have been submitted to the logical printer **DepartPrt** and that are pending, enter the command:

```
pdls -f "printer-name-requested==DepartPrt &&
     current-job-state==pending" SPOOL1:
```

The following is displayed:

```
                                Current    Printer    Printer
Job   ID         Name           State      Requested  Assigned
---- ---------- ----------- ---------- ---------  ---------
5     SplX:1114 Trip-report   pending    DepartPrt
```

List the Brief Attributes of a Logical Printer

- To list the **brief** attributes of your default logical printer (assigned to the **PDPRINTER** environment variable), enter the following command:

  ```
  pdls -c printer DepartPrt
  ```

  It causes the following to be displayed:

  ```
  Printer     Realization  Enabled  Queue
  -------     -----------  -------  -----
  DepartPrt   logical      true     DepartQ
  ```

  If you do not specify a printer name, the command will list the brief attributes of all the logical printers that share the same spooler with your default logical printer (as defined in your **PDPRINTER** environment variable.

List Document Formats Supported

- To list the document-formats supported by the logical printer **DepartPrt** in the **line** style with headings (default style for an attribute list), enter the following command:

  ```
  pdls -c printer -r document-format-supported DepartPrt
  ```

  The following is an example of what might be displayed:

  ```
  DepartPrt: document-formats-supported=document-format-ASCII
  =document-format-ps
  ```

- To list the document-formats supported by the logical printer **DepartPrt** in the column style with headings, enter the following command:

  ```
  pdls -c printer -r document-formats-supported -s column DepartPrt
  ```

  The following is an example of what might be displayed:

  ```
  Document
  Formats
  -----------
  ASCII
  PostScript
  ```

Create an Archive File

- To create an archive file for a spooler, enter the command:

  ```
  pdls -c server -r archive  SPOOL1 > /attr/SPOOL1.archive
  ```

SEE ALSO

pdclean(1), pdcreate(1), pddelete(1), pddisable(1), pdenable(1), pdmod(1), pdpause(1), pdpr(1), pdpromote(1), pdq(1), pdresubmit(1), pdresume(1), pdrm(1), pdset(1), pdshutdown(1), pd_att(5).

To view information about all supported attributes, enter the command:

```
man pd_att
```

This will display a list of files by object from which you can select the attribute listing you want.

STANDARDS CONFORMANCE

pdls:POSIX 1387.4

NAME

pdmod - modifies attributes of submitted print jobs

SYNOPSIS

pdmod -h

pdmod [**-g**] [**-m "***MessageText***"**] [**-n** *CopyCount*] [**-r "***RequestedAttributes***"**] [**-s** *StyleName*] [**-t** *JobName*] [**-x "***AttributeValuePairs***"**] [**-X** *AttributesFileName*]
LocalJobId ... | *GlobalJobId ...*

DESCRIPTION

Use the **pdmod** command to modify the values of job and document attributes of previously submitted print jobs.

You can only modify preprocessing, pending, held, paused, or retained jobs. You cannot modify any jobs that are currently printing.

Modifying an existing job may affect the scheduling of the job.

Table 1-1 lists the four modification operators:

Table 1-1. Four Modification Operators:

Operator	Syntax	Description
replace	*attribute=value*	Replaces the entire value of the attribute *attribute* with value or, if not already present, adds the attribute-value pair to the job.
add-values	*attribute+=value*	Adds the value *value* to the attribute *attribute*. You cannot add values to single-valued attributes. An add request that duplicates values on a multi-valued attribute has no effect on the job.
remove-values	*attribute-=value*	Removes the values *value* from the attribute *attribute*. A remove request for a nonexistent value has no effect on the object. A remove request for the last or only value of an attribute is equivalent to a reset-to-default request.
reset-to-default	*attribute==*	Set the attribute *attribute* to the default values according to the job defaulting-hierarchy. If values are supplied with a reset request, they are ignored.

If you do not specify a value with a replace, add, or remove request, an error is issued and the request to change the attribute value for the object is rejected. Processing continues with the next argument on the command line, if any.

Note: If any modification is not accepted, then the whole request is rejected and the job continues as before.

You must be authorized to modify a job belonging to another person. Use the global job identifier to identify the job belonging to another person.

Options

You can use the following options with the **pdmod** command:

-g Turn off headings. This option is equivalent to specifying the command-attribute **headings=false**.

-h Display a command-specific help message containing information about command syntax and options. This option cannot be used with another option or with an attribute.

-m "*MessageText***"**
Specify the message you want stored in the **job-message-from-administrator** attribute. You can use the message to give the reason the job is being modified or to provide other comments. If you do not specify the **-m** option, the message already stored with the job remains unchanged.

You can list this message by specifying **requested-attributes=job-message-from-administrator** with the **pdls** command. This option is equivalent to specifying the command-

p

attribute **message**.

-n *CopyCount*
Specify the number of document copies. A copy count of zero (0) is an error. This option is equivalent to the object-attribute **copy-count**.

-r *RequestedAttributes*
Specify the attribute values that you want to display for the specified objects.

RequestedAttributes can be **none** (default), **brief**, or **verbose**.

This option is equivalent to the command-attribute **requested-attributes**.

-s *StyleName*
Specify the style in which you want the attributes displayed.

StyleName may be **column** (default), or **line**.

This option is equivalent to the command-attribute **style**.

-t *JobName*
Specify the new name that you want for the job. This option is equivalent to the object-attribute **job-name**.

-x "*AttributeValuePairs***"**
A single attribute string, consisting of one or more attribute-value pairs. Prefix the attribute value with the **=** character to replace a value, the **+=** characters to add a value, and the **-=** characters to remove a value. Use the **==** characters with no attribute value to set the attribute to its default value.

-X *AttributesFileName*
The name of a file containing attribute-value pairs to be inserted at the current point in the command line. This option is equivalent to the command-attribute **attributes**.

Command Attributes

You can specify these attributes in a **-x "***AttributeValuePairs***"** string or in an attributes file designated with the **-X** *AttributesFileName* option.

attributes=*AttributesFileName*
Specifies the designated attributes file that HPDPS is to read and insert at the current point in the command line. This file contains attributes and value pairs that HPDPS uses to expand on the command being issued.

p

headings=Boolean
Boolean can be **true** (default), or **false**. Specifies if you want headings displayed.

message="*MessageText***"**
Specify the message that you want stored in the **job-message-from-administrator** attribute. You can use the message to give the reason the job is being modified or to provide other comments. If you do not specify the **message** attribute, the message already stored with the job remains unchanged.

You can list this message by specifying **requested-attributes=job-message-from-administrator** with the **pdls** command.

requested-attributes=*AttributeType*
Specify which attributes you want displayed. *AttributeType* can be **none** (default), **brief**, or **verbose**.

style=*StyleName*
Specify the presentation format that you want for the displayed output. *StyleName* can be **column** (default) or **line**.

Object Attributes

You can specify these attributes in a **-x "***AttributeValuePairs***"** string or in an attributes file designated with the **-X** *AttributesFileName* option.

copy-count=*number*
Specify the number of document copies you want. A copy count of zero (0) is an error.

job-hold=*Boolean*
Specify whether you want to put the job in a hold state.

Boolean can be **true** or **false** (default).

job-name=*name*
Specify the new name that you want for the job.

Also, you may specify any other settable attributes for the job or document.

Note: You can set the value for a settable attribute with either the **pdset** or **pdmod** command.

Arguments

Use the arguments to identify the specific objects that you want to modify. If you specify multiple objects, each must be separated by spaces.

LocalJobId or *GlobalJobId*	Specify the local or global job identifier. If you modify more than one job with a single command, each must be separated with one or more white spaces.

EXAMPLES

Modify Content Orientation

To modify the job (local id of 10) and change the orientation to landscape, enter the command:

```
pdmod -x content-orientation=landscape 10
```

Modify Job Hold Condition

To modify the job-attribute **job-hold** to **false** for job 10, so that the job (which was previously submitted with **job-hold** set to **true**) can be scheduled, enter the command:

```
pdmod -x job-hold=false 10
```

Modify Job Comment

To modify the job-attribute **job-comment** to be **"Test Results 100"** for job 10, enter:

```
pdmod -x job-comment="Test Results 100" 10
```

Modify Number of Sides to be Printed

To modify the document-attribute **sides** to be 2 for all documents in the global job 1011222243 on server DivSpool2 enter:

```
pdmod -x sides=2 DivSpool2:1011222243
```

To modify the sides document attribute to specify to print documents on both sides for jobs 10 and 20:

```
pdmod -x sides=2 10 20
```

Add a Value

To add the value **building47** to the attribute **Printer-locations-requested** for job 10, enter:

```
pdmod -x Printer-locations-requested+=building47 10
```

Remove a Value

To remove the value **building47** from the **Printer-locations-requested** attribute in my job 10, enter:

```
pdmod -x Printer-locations-requested-=building47 10
```

Reset to Default Values

To reset the **job-originator** job attribute to the default values in my job 10, enter:

```
pdmod -x job-originator==10
```

Combining Modifications

To modify job 17, notifying the operator, and placing the job on hold, enter the command:

```
pdmod -x "job-message-to-operator="
This job requires blue legal-sized paper.
Please advise when the paper is
available
```

```
        job-hold=yes
        17
```

SEE ALSO

pdclean(1), pdcreate(1), pddelete(1), pddisable(1), pdenable(1), pdls(1), pdpause(1), pdpr(1), pdpromote(1), pdq(1), pdresubmit(1), pdresume(1), pdrm(1), pdset(1), pdshutdown(1)

For information about:

- Headings, see the **pdls** command-attribute **style**.
- Requested attributes, the **pdls** command-attribute, **requested-attributes**.
- Style, see the **pdls** command-attribute, **style**.
- Job defaulting-hierarchy, see the **pdpr** command.

To view information about all supported attributes, see manpage *pd_att*(5). It contains a list of files by object from which you can select the attribute listing that you want.

STANDARDS CONFORMANCE

pdmod: POSIX 1387.4

NAME

pdmsg - displays the text and description of an HPDPS message at the command line

SYNOPSIS

pdmsg [**-d**] [**-t**] *message-number*...

DESCRIPTION

The **pdmsg** utility displays the text and description of an HPDPS message at the command line. The **pdmsg** utility extracts the text and description from the appropriate message catalogs. If you do not specify the **-d** or **-t** option, **pdmsg** displays both the text and the message description.

Options

pdmsg uses the following options:

-d Display only the description of a message, which consists of a message number, an explanation of the message, the system action, and the response.

-t Display only the text of the message, which consists of a message number along with the actual words of the message itself.

message-number
The number of the HPDPS message you want displayed. Valid message numbers range from 5010-002 through 5010-1500 for HPDPS. However, not every number in these ranges has an associated message.

EXAMPLES

To View Text and Description for a HPDPS Message

To view text and description for the HPDPS message 5010-096, enter the command:

```
pdmsg 5010-096
```

The following information appears:

```
5010-096 The value %1$AV is not supported for attribute %2$AN.

5010-096

EXPLANATION: The server or printer does not support this value.

SYSTEM ACTION:  HPDPS could not process the request.

RESPONSE:  Enter the command again and specify a value that is
supported by the server and printer, or specify a printer that
supports (by association) the value.
```

To View Only the Text for a HPDPS Message

To view only the text for HPDPS message 5010-096, enter the command:

```
pdmsg -t 5010-096
```

The following message appears:

```
5010-096 The value %1$AV is not supported for attribute %2$AN.
```

To View Only the Description for a HPDPS Message

To view only the description for HPDPS message 5010-096, enter the command:

```
pdmsg -d 5010-096
```

The following messages appears:

```
EXPLANATION: The server or printer does not support this value.

SYSTEM ACTION:  HPDPS could not process the request.

RESPONSE:  Enter the command again and specify a value that is
supported by the server and printer, or specify a printer that
supports (by association) the value.
```

NOTE: **%1$AV** and **%2$AN** are two types of substitution variables that will be replaced by actual values at runtime.

NAME

pdpause - pauses jobs, physical printers, servers, or queues

SYNOPSIS

pdpause -h

pdpause [**-c** *ObjectClass*] [**-m** "*MessageText*"] [**-x** "*AttributeValuePairs*"] [**-X** *AttributesFileName*]
{ *LocalJobId* ...
| *GlobalJobId* ...
| *ServerName* ...
| [*ServerName*:]*PrinterName* ...
| [*ServerName*:]*QueueName* ... }

pdpause -j [**-m** "*MessageText*"] [**-x** "*AttributeValuePairs*"] [**-X** *AttributesFileName*] [*ServerName*:]*PrinterName* ...

DESCRIPTION

Use this administrative command, **pdpause**, to pause an object that holds jobs or to pause a job.

Objects that can be paused are as follows:

- pending print jobs
- held print jobs
- currently-processing print jobs
- physical printers
- queues
- servers (pauses all of the queues contained in a spooler or all physical printers contained in a supervisor)

To resume a paused object, use the **pdresume** command.

Note: The **pdpause** and **pdresume** commands control output, whereas the **pdenable** and **pddisable** commands control input.

Options

Use the following options with the **pdpause** command:

-c *ObjectClass*
Specify the object class that you want to pause. The *ObjectClass* can be one of the following:

printer (default)
queue
job
server

Within the valid classes, **printer** is a physical printer, and a **server** is either a spooler or a supervisor. This option is equivalent to specifying the command-attribute **class**.

-h Display a command-specific help message containing information about command syntax and options. This option cannot be used with another option or with an attribute.

-j Only valid when used with object-class **printer**. Use this option to pause the currently printing job on the specified physical printer.

-m "*MessageText*"
Specify the message that is to be associated with the *ObjectClass* that is being paused: **printer**, **queue**, **job**, or **server**. You can use this message to indicate the reason that a pause has taken place or to provide other comments.

When pausing a supervisor, HPDPS propagates the message to the **message** attribute of the physical printers residing in a supervisor. The **message** attribute for the supervisor is not changed.

When pausing a spooler, the message is propagated to the **message** attribute of the queues residing in the spooler. The **message** attribute for the spooler is not changed.

If the command operates on a **printer** or a **queue**, you can list this message by specifying **requested-attributes=message** with the **pdls** command.

When the command operates on a job, the specified text becomes the value of the **job-message-from-administrator** attribute. You can list this message by specifying **requested-attributes=job-message-from-administrator** with the **pdls** command.

If the **-m** option is not specified, the message already stored with the **printer**, **queue**, **job**, or **server** remains unchanged.

This option is equivalent to specifying the command-attribute **message**.

-x "*AttributeValuePairs***"**
A single attribute string, consisting of one or more attribute-value pairs.

-X *AttributesFileName*
The name of a file containing attribute-value pairs you want inserted at the current point in the command line. This option is equivalent to specifying the command-attribute **attributes**.

Command Attributes

You can specify these attributes in a **-x "***AttributeValuePairs***"** string or in an attributes file designated with the **-X** *AttributesFileName* option.

attributes=*AttributesFileName*
Specifies the designated attributes file that HPDPS is to read and insert at the current point in the command line. This file contains attribute and value pairs that HPDPS uses to expand on the command being entered.

class=*ObjectClass*
Specify the object class that you want to pause. Valid object class names for the **pdpause** command are: **printer**, **queue**, **job**, or **server**.

Within the valid classes, **printer** is a physical printer, and a **server** is either a spooler or a supervisor.

message="*MessageText***"**
Specify the message that you want associated with the **printer**, **queue**, **job**, or **server** that is being paused. You can use this message to indicate the reason that a pause has taken place or to provide other comments.

When pausing a supervisor, HPDPS propagates the message to the **message** attribute of the physical printers residing in a supervisor. The **message** attribute for the supervisor is not changed. When pausing a spooler, the message is propagated to the message attribute of the queues residing in the spooler. The **message** attribute for the spooler is not changed.

If the command operates on a **printer** or a **queue** you can list this message by specifying **requested-attributes=message** with the **pdls** command.

When the command operates on a **job**, the specified text becomes the value of the **job-message-from-administrator** attribute. You can list this message by specifying **requested-attributes=job-message-from-administrator** with the **pdls** command.

If the message attribute is not specified, the message that is already stored with the **printer**, **queue**, **job**, or **server** remains unchanged.

Object Attribute

There are no object attributes for this command.

Arguments

Use the argument to specify the object that you want to pause. If you specify multiple objects, all of the objects must be of the same class, and each must be separated by spaces.

You can use the following arguments with the **pdpause** command:

LocalJobID or *GlobalJobId*
Specify the jobs that you want to pause as determined by a local job identifier or global job identifier. Only you as an administrator have the authority to pause jobs and would generally use the global job identifier. However, for your own jobs, you can use the local job identifier. The following actions take place for these three cases:

A currently printing job is paused:

- The job stops at a "pause" point.

  Note: A pause point may be the next page, sheet, document, or job boundary depending on the type of printer on which the job is being printed
- The job-state changes to paused.
- The printer remains available to accept work.
- Other jobs can be assigned to the printer.

A pending job is paused:

The job is prevented from being scheduled but does not affect any printer.

A held job is paused:

The job is prevented from becoming pending, even if the reasons for the job being held are removed. For example, the specified **`print-after`** time expires.

Paused jobs remain in the queue until they are resumed or canceled. A paused job can be modified, but it cannot be resubmitted until a **`pdresume`** command is issued for the job.

[*ServerName*:]*PrinterName*
Specify the printers that you want to pause. The action taken differs whether the **`-j`** option is included in the command.

Without the **`-j`** option:

- Any currently-printing job stops at a "pause" point
- The printer-state is changed to paused
- The job-state is left at processing; the job is still assigned to the printer.

The physical printer still accepts print jobs from its associated queue up to the **`maximum-concurrent-jobs`** limit, but will not print them.

With the **`-j`** option:

- The job stops at a "pause" point,

Note: A pause point may be the next page, sheet, document, or job boundary depending on the type of printer on which the job is being printed

- The job-state changes to paused
- The printer remains available to accept work
- Other jobs can be assigned to the printer

The physical printer may still accept print jobs from its associated queue and process them.

Note: Logical printers cannot be paused because they do not hold jobs.

[*ServerName*:]*QueueName*
Specify the queue that you want to pause.

Pausing a queue halts the distribution of jobs from the queue to the physical printers associated with the queue. Pausing a queue does not prevent it from accepting jobs from its associated logical printers.

ServerName
Specify the server on which you want the command to operate. A server does not have a paused state. Issuing the command against a spooler pauses all queues contained within the spooler. Issuing the command against a supervisor pauses all of physical printers contained within the supervisor.

EXAMPLES

Pause a Physical Printer

To pause physical printer 3828C and include a printer message as to why the printer is being paused, enter the following command:

```
pdpause -m "Toner is low, refilling" 3828C
```

Pause a Currently Printing Job

To pause the currently-printing job on printer 3828C, enter the following command:

```
pdpause -j 3828C
```

Pause a Queue

To pause the queue Div1Q2, enter the following command:

```
pdpause -c queue  Div1Q2
```

To pause all of the queues in spooler DivSpool1, enter the following command:

```
pdpause -c server  DivSpool1
```

SEE ALSO

pdclean(1), pdcreate(1), pddelete(1), pddisable(1), pdenable(1), pdls(1), pdmod(1), pdpr(1), pdpromote(1), pdq(1), pdresubmit(1), pdresume(1), pdrm(1), pdset(1), pdshutdown(1)

STANDARDS CONFORMANCE

pdpause: POSIX 1387.4

NAME

pdpr - submits print jobs

SYNOPSIS

pdpr -h

pdpr [**-f** *FileName*] [**-g**] [**-l**] [**-n** *NumberOfCopies*] [**-N** *NotificationMethod*]
[**-p** *LogicalPrinterName*] [**-r** *RequestedAttributes*] [**-s** *StyleName*] [**-t** *JobName*]
[**-x "***AttributeValuePairs***"**] [**-X** *AttributesFileName*] [*FileName* ...]

DESCRIPTION

Use the **pdpr** command to submit print jobs to logical printers. Each print job can contain multiple printable documents and any number of print resources.

The target logical printer name defaults to the value of the **PDPRINTER** environment variable. You can override the default by specifying the name of another logical printer using the option **-p** *LogicalPrinterName* or the per-job-attribute **printer-name-requested**.

When the **print-job** request is accepted by the server, it is given a unique *global ID* (job identifier). The job also has a *local ID* that can only be used by you (the job submitter). The job submitter can use either the global ID or the local ID in subsequent commands, such as **pdmod, pdrm**, or **pdls**. Anyone else who accesses the job, such as an administrator, must use the global ID.

The values for job and document attributes are derived as follows:

1. Values specified in the **pdpr** command using the **-x "***AttributeValuePairs***"** option or specified in an attributes file.
2. The values of an *initial-value-job* object specified in the job attribute **initial-value-job** or the values for an *initial-value-document* object specified in the document attribute **initial-value-document**.
3. The values of an *initial-value-job* object specified in the **printer-initial-value-job** attribute or the values of an *initial-value-document* object specified in the **printer-initial-value-document** attribute of the logical printer the job is submitted to.
4. Server defaults for required attribute values not specified by the items in 1 through 3.

You must be authorized to submit print jobs to the specified printer if the logical printer is protected.

p

If the server cannot locate a physical printer (associated with the requested logical printer) supporting the job and document attributes, the job is rejected.

Options

Use the following options with the **pdpr** command:

-f *FileName*
Specify a file that you want included in a print job. This option is useful for identifying files whose names begin with a **-** character.

-g Turn off headings. This option is equivalent to specifying the command-attribute **headings=false**.

-h Display a command-specific help message containing information about command syntax and options. This option cannot be used with another option or with an attribute.

-l Create a job containing pointers (symbolic links) to the job files rather than taking snapshot copies. When the job is assigned to a physical printer and is ready to be printed, the supervisor uses the pointers in the job to locate the original files and copies them at that time. This can be useful when printing large files or jobs.

Notes: You can make changes to the files after the job is accepted and up to the time the supervisor makes a copy.

You must use caution when using this option because:

a. Depending on when you make changes to the files and the supervisor copies them, the printed output may or may not reflect the changes.

b. If you delete the file before the job prints or while the job is being printed, the printed output may fail or be incomplete.

-n *NumberOfCopies*
Specify the number of document copies you want printed. *NumberOfCopies* can be either **1** (default) or more copies.

If you do not specify the **-n** option, the copy count defaults to one (1). A copy count of zero (0) is an error. This option is equivalent to specifying the per-document-attribute **copy-count**.

-N *NotificationMethod*
Specify the delivery method that you want used in notification of job events. *NotificationMethod* can be **message**, **email** (default), or **none**.

This option is equivalent to specifying the command-attribute **notification-delivery-method**.

Note: Using this option and its value causes the **notification-profile** attribute to be built.

-p *LogicalPrinterName*
The logical printer to which you want the job submitted. If you do not specify this option, the **PDPRINTER** environment variable is used to determine the printer. This option is equivalent to specifying the object-attribute **printer-name-requested**.

-r *RequestedAttributes*
Identify the attribute values you want displayed for the specified objects. *RequestedAttributes* can be **none** (default), **brief**, or **verbose**.

This option is equivalent to specifying the command-attribute **requested-attributes**.

-s *StyleName*
Specify the format in which you want the attributes displayed. *StyleName* can be **column** (default) or **line**.

This option is equivalent to specifying the command-attribute **style**.

-t *JobName*
Specify the name you want to assign to the job. This option is equivalent to specifying the object-attribute **job-name**.

-x "*AttributeValuePairs***"**
A single attribute string, consisting of one or more attribute-value pairs.

Document attributes defined with the **-x** option affect all files whose names follow that **-x** option on the command line, unless the document attributes are reset by using the **-x** option again.

-X *AttributesFileName*
The name of a file containing attribute-value pairs to be inserted at the current point in the command line. This option is equivalent to specifying the command-attribute **attributes**.

- Causes the command to read from **stdin**.

Command Attributes

You can specify these attributes in a **-x "***AttributeValuePairs***"** string or in an attributes file designated with the **-X** *AttributesFileName* option.

attributes=*AttributesFileName*
Cause the designated attributes file to be read.

headings=*Boolean*
Specify if you want headings displayed. *Boolean* can be **true** (default) or **false**.

notification-delivery-method=*NotificationMethod*
Specify the delivery method you want used in notification of job events. *NotificationMethod* may be **message**, **email** (default), or **none**.

Note: Using this attribute and its value causes the **notification-profile** to be built.

requested-attributes=*AttributeType*
Specify which attributes you want displayed. *AttributeType* is **none** (default), **brief**, or **verbose**.

style=*StyleName*
Specify the presentation format in which you want the output displayed. *StyleName* can be **column** (default) or **line**.

p

Object Attributes

There are two types of object attributes for the **pdpr** command (per-job and per-document) that have some special conditions. However, you can specify any settable or specifiable job or document attribute with the **pdpr** command. You can only specify settable attributes with the **pdset** and **pdmod** commands.

Per-Job Attributes

Per-job attributes apply to the job as a whole and may occur anywhere in the **pdpr** command.

printer-name-requested=[*ServerName* **:**]*PrinterName*

Specify the name of a logical printer that you want the job submitted to.

If this attribute-value pair is not specified, the **PDPRINTER** environment variable is used to determine the printer.

job-name=*name*

Specify the name of the print job.

Per-Document Attributes

The per-document attributes apply to the individual document and precede the file name of the document that they affect. If no file name is specified, the per-document attributes are applied to the **stdin** file.

The value for a given attribute must be the same for all the documents in the job except for **initial-value-document**, **document-type**, and **document-comment**.

The only per-document attributes an **initial-value-document** can set are **document-type** and **document-comment**.

Note: Per-job attributes can be specified anywhere on the command line. Per-document attributes cannot be set after the final file name.

copy-count=*NumberOfCopies*

Specify the number of document copies that you want printed. *NumberOfCopies* can be **1** (default) or **number**.

If no value is specified, the value of the **copy-count** attribute defaults to one (1). A copy count of zero (0) is an error.

document-format=*format*

Specify the format that you want for the document. If the value for the document-format attribute is not provided, HPDPS attempts to determine the correct value for the document format by reading the file. If HPDPS cannot determine the document format, the value for the document-format attribute defaults to **ascii**.

p

Arguments

Use the argument to identify the name of a file to be printed. If you specify multiple file names, each must be separated by spaces. Each file becomes a document within the job.

[*FileName*...]

Specify the document that you want printed. You should preceded the file name by any per-document attributes.

If you do not specify a file name or do specify a **-** (hyphen) as the file name, **pdpr** reads from **stdin.**

If you specify a multi-document job, and if any document within the job cannot be supported, the job is rejected and an error message is generated.

EXAMPLES

Print a File

To submit the file **File1** to your default logical printer, enter one of the following commands:

```
pdpr File1
pdpr -f File1
```

To submit the file **File1** to the logical printer LogPrt2, enter one of the following commands:

```
pdpr -p LogPrt2 File1
pdpr -x printer-name-requested=LogPrt2 File1
```

Number of Copies

To submit the file **File1** to the default logical printer and to specify three copies of the print job, enter one of the following commands:

```
pdpr -n 3 File1
pdpr -x copy-count=3 File3
```

To submit the job to the default logical printer and specify two copies of each file with the **-n** option, enter one of the following commands:

```
pdpr -n 2 Title Contents Body1 Body2 Append
pdpr -x copy-count=2 Title Contents Body1 Body2 Append
```

HPDPS prints two copies of Title, followed by two of Contents, and so on for each file in the job.

To submit the job to the default logical printer and specify two copies of the complete job, enter the command:

```
pdpr -x results-profile=:::2 Title Contents Body1 Body2 Append
```

HPDPS prints a single copy of each file in the job, and then prints a second set in the same manner.

To submit the job to the default logical printer and specify two copies of the complete job with each copy of the job containing three copies of each file, enter the command:

```
pdpr -n 3 -x results-profile=:::2 Title Contents Body1 Body2 Append
```

There are printed three copies of Title, followed by three copies of Contents and so until the first copy of the job is done. HPDPS prints a second set in the same manner.

Specifying Job Name

To submit the file **File1** to your default logical printer and specify the job name, enter the command:

```
pdpr -t CmdRef File1
```

Double-sided Printing

To submit the file **File1** to the default logical printer and specify double-sided printing, enter the command:

```
pdpr -x "sides=2" File1
```

Page Orientation

To submit the file **File1** to the default logical printer and specify landscape orientation, enter the command:

```
pdpr -x content-orientation=landscape File1
```

Document Format

To submit the file **PSFile2** to the default logical printer and specify a document format of ASCII, enter the command:

```
pdpr -x document-format=ascii PSFile2
```

Multiple Documents

To submit the files **File1** and **File2** to the default logical printer, enter the command:

```
pdpr File1 File2
```

To submit the files **File1** and **File2** to the default logical printer and specify three copies of each file with double-sided printing, enter the command:

```
pdpr -n 3 -x sides=2 File1 File2
```

Requesting Status

To submit the file **File1** to the default logical printer and receive brief status information, enter the command:

```
pdpr -r brief File1
```

HPDPS displays information similar to the following about your job:

Job	ID	Name	Current State	Printer Requested	Printers Assigned
8	sp15:075410002	File1	pending	LogPrt1	

To submit the file `File1` to the default logical printer and receive brief status information without headings, enter one of the following commands:

```
pdpr -g -r brief File1
pdpr -x  "headings=no  requested-attributes=brief" File1
```

HPDPS displays information similar to the following about your job:

```
8    sp15:075410002  File1  pending     LogPrt1
```

Delay Printing of File

To submit the file `BigJob` to the default logical printer and delay printing until after 6:30 p.m., enter the command:

```
pdpr -x job-print-after=18:30:00 BigJob
```

Delay Printing of File and Specifying a Symbolic Link

To submit the file `BigJob` to a logical printer LogPrt4 without copying the file and delay printing until after 6:30 p.m. on April 4, 1995, enter the following command:

```
pdpr -l -x job-print-after="18:30:00 04/04/95"   BigJob
```

Job Discard Time

To submit the file `BigJob` to the default logical printer and specify that the job is to be discarded if it has not printed by 5:00 p.m., enter the command:

```
pdpr -x job-discard-time=17:00:00 BigJob
```

Job to a Specific Printer Location

To submit the file File5 to be printed on one of the printer devices located in building 20 room 17, enter the command:

```
pdpr -p LogPrt20 -x printer-locations-requested=bld20.rm17 File5
```

Job to a Specific Physical Printer

To submit the file File5 to be printed on physical printer PhysPrt1, enter the command:

```
pdpr -p LogPrt20 -x physical-printers-requested=PhysPrt1 File5
```

Retaining a Job

To submit the file `File1` to the default logical printer, request feedback of job attributes (brief group), and specifying a retention period of 90 minutes so you can print more copies after you have looked at the first copy, enter the command:

```
pdpr  -r brief -x job-retention-period=90 File1
```

You want to note the job number (local ID) so you can use that number when you want to print more copies within the time allotted.

Specifying an Initial Value Object (IVO)

To submit the file `File1` to the default logical printer and use the job attribute values specified in the initial value job ivj23, enter the command:

```
pdpr -x initial-value-job=ivj23 File1
```

To submit the file `File1` to the default logical printer and use the document attribute values specified in the initial value document ivd44, enter the command:

```
pdpr -x initial-value-document=ivd44 File1
```

To submit the file `File1` to the default logical printer and use both IVOs specified in the previous two examples, enter the command:

```
pdpr -x "initial-value-document=ivd44 initial-value-job=ivj23" File1
```

Specifying Attributes Files for a Job

To submit the file **File1** to the default logical printer and specify the attributes file **myjob.att**, enter the command:

```
pdpr -X myjobs.att File1
```

To submit the file File5 to the default logical printer and specify the two attributes files (**default.att** and **special.att**), enter the command:

```
pdpr -X default.att -X special.att File5
```

Overriding an Attribute Value in an Attributes File

To submit the file **File1** to the default logical printer and override the value of 2 for the **sides** attribute specified in a given attributes file, enter the command:

```
pdpr -X default.att -x sides=1 File1
```

Requesting Feedback Concerning the Job as Each Event Happens

To submit the file **File1** to the default logical printer and have all possible event notifications sent to your display screen, enter the command:

```
pdpr -x "notification-profile={delivery-method=message \
     event-identifiers=job-modified class-job-problem \
     class-job-attention}" File1
```

To submit the file **File1** to the default logical printer and have the default event notification sent to you by electronic-mail, enter the command:

```
pdpr  -x "notification-profile={delivery-method=electronic-mail}" File1
```

Specifying a Different Job Owner

To print the files **MyFile1** and **MyFile2** and specify **Marta** as the owner of the job, enter the command:

```
pdpr -x job-owner=Marta MyFile1 MyFile2
```

SEE ALSO

pdclean(1), pdcreate(1), pddelete(1), pddisable(1), pdenable(1), pdls(1), pdmod(1), pdpause(1), pdpromote(1), pdq(1), pdresubmit(1), pdresume(1), pdrm(1), pdset(1), pdshutdown(1)

For information about:

- Headings, see **pdls** command-attribute **style**.
- Requested attributes, see **pdls** command-attribute **requested-attributes**.
- Style, see **pdls** command-attribute **style**.

To view information about all supported attributes, see manpage *pd_att*(5). It contains a list of files by object from which you can select the attribute listing that you want.

STANDARDS CONFORMANCE

pdpr: POSIX 1387.4

NAME

pdpromote - advances a job to the top of a queue

SYNOPSIS

pdpromote -h

pdpromote [**-m "***MessageText***"**] [**-x "***AttributeValuePairs***"**] [**-X** *AttributesFileName*]
LocalJobId ... | *GlobalJobId* ...

DESCRIPTION

Use the **pdpromote** command to move a pending job before any currently-queued jobs. The job becomes the first job in the queue. If another job is then promoted, it becomes the first job in the queue (ahead of the job previously promoted).

A move to the beginning of the queue does not necessarily guarantee that the job will be the next job printed. The jobs currently printing on each of the physical printers associated with the queue continue printing. The server assigns the promoted job to the first physical printer that becomes available and is capable of handling the promoted job.

You must have at least read and write authority for the queue to promote your own jobs as well as the jobs belonging to other people.

The priority level of a print job can be changed by setting the **job-priority** attribute using the **pdmod** or the **pdset** command. However, a job is promoted to the top of the queue by the **pdpromote** command regardless of it priority.

Options

You can use the following options with the **pdpromote** command:

-h Display a command-specific help message containing information about command syntax and options. This option cannot be used with another option or with an attribute.

-m "*MessageText***"**
Specify the message you want stored in the **job-message-from-administrator** attribute. You can use the message to give the reason the job is being or has been promoted or to provide other comments. If you do not specify the **-m** option, the message already stored with the job remains unchanged.

You can list this message by specifying **requested-attributes=job-message-from-administrator** with the **pdls** command.

This option is equivalent to specifying the command-attribute **message**.

-x "*AttributeValuePairs***"**
A single attribute string, consisting of one or more attribute-value pairs.

-X *AttributesFileName*
The name of a file containing attribute-value pairs to be inserted at the current point in the command line. This option is equivalent to the command-attribute **attributes**.

Command Attributes

You can specify these attributes in a **-x "***AttributeValuePairs***"** string or in an attributes file designated with the **-X** *AttributesFileName* option.

attributes=*AttributesFileName*
Cause the designated attributes file to be read.

message="*MessageText***"**
Specify the message that you want stored in the **job-message-from-administrator** attribute. You can use the message to give the reason the job is being promoted or to provide other comments. If you do not specify the **message** attribute, the message already stored with the job remains unchanged.

You can list this message by specifying **requested-attributes=job-message-from-administrator** with the **pdls** command.

Object Attributes

There are no object attributes for this command.

Arguments

Use the arguments to identify the specific objects that you want to promote. You can use the following argument values with the **pdpromote** command:

LocalJobId or *GlobalJobId*

Specify the local or global job identifier of the job that you want to promote.

You as an administrator would generally use the global job identifier but you can promote your own jobs using the local job identifier.

When a job is specified with the **pdpromote** command, it becomes the first job in the queue. If a another job is then promoted, it becomes the first job in the queue (ahead of the job previously promoted).

EXAMPLES

Promote Job

To promote job 6450500001 on server DServe1, enter the command:

```
pdpromote DServe1:6450500001
```

To promote job 1099600001 on server SPOOL1, enter the command:

```
pdpromote -m "This job must be printed in 10 minutes"
SPOOL1:1099600001
```

SEE ALSO

pdclean(1), pdcreate(1), pddelete(1), pddisable(1), pdenable(1), pdls(1), pdmod(1), pdpause(1), pdpr(1), pdq(1), pdresubmit(1), pdresume(1), pdrm(1), pdset(1), pdshutdown(1)

STANDARDS CONFORMANCE

pdpromote: POSIX 1387.4

NAME

pdq - queries and lists the status of one or more print jobs

SYNOPSIS

pdq -h

pdq [**-f "***FilterCriteria***"**] [**-F**] [**-g**] [**-j**] [**-p** *LogicalPrinterName*] [**-r** *RequestedAttributes*] [**-s** *StyleName*] [**-U**] [**-x "***AttributeValuePairs***"**] [**-X** *AttributesFileName*] [*LocalJobId*[. *DocNumber*] ... | [*GlobalJobId*[. *DocNumber*] ...]

DESCRIPTION

Use the **pdq** command to list the status of some or all of the print jobs that have been submitted to a logical printer.

If you omit both the *LocalJobId* and *GlobalJobId* and do not name a logical printer (either with the option **-p** *LogicalPrinterName* or the command-attribute **printer-name-requested**), all print jobs in the queue associated with the default logical printer (as defined in the **PDPRINTER** environment variable) are listed.

If you do not specify a value for the **-r** option or the command-attribute **requested-attributes**, the **pdq** command defaults to the value **brief**. By default, you can list the attribute values only for jobs you submit.

Jobs will be listed in the order in which the queue considers them for printing.

You can use the filtering option so the status is returned for specific jobs only. The jobs have a predefined value for the filter that is equal to the job attribute **user-name**; this value is set to your login identity when you submit a job.

Options

Use the following options with the **pdq** command:

-f *FilterCriteria*
Specify the filter selection criteria that you want to use for the candidate objects or print jobs. Among the candidate object or print jobs, only those matching the filter expression are returned. This option is equivalent to specifying the command-attribute **filter**.

-F Turn off all filtering (both specified and default).

See the **-U** option for only turning the default filter off.

-g Turn off headings. This option is equivalent to specifying the **headings=false** command attribute.

-h Display a command-specific help message containing information about command syntax and options. This option cannot be used with another option or with an attribute.

-j Use this option to return only job attributes.

-p *LogicalPrinterName*
Query all jobs in queue associated with this logical printer. This option is equivalent to specifying the command-attribute **printer-name-requested**.

-r *RequestedAttributes*
Specify the attribute values you want displayed for the specified objects. Values can be: **brief** (default), **verbose**, **archive**, **"***attribute_list***"** , **all**, or **none**. **archive** displays only attributes that can be specified and set. This option is equivalent to specifying the command-attribute **requested-attributes**.

-s *StyleName*
Specify the format in which you want the attributes displayed. *StyleName* can be **column** (default) or **line**

This option is equivalent to specifying the command-attribute **style**.

-U Suppress the default user-name filter.

-x "*AttributeValuePairs***"**
A single attribute string, consisting of one or more attribute-value pairs.

-X *AttributesFileName*
The name of a file containing attribute-value pairs to be inserted at the current point in the command

p

line. This options is equivalent to specifying the command attributes **attribute**.

Command Attributes

You can specify these attributes in a **-x "***AttributeValuePairs***"** string or in an attributes file designated with the **-X** *AttributesFileName* option.

attributes=*AttributesFileName*
Cause the designated attributes file to be read.

filter="*FilterCriteria***"**
Specify the filter selection criteria that you want used if you request attribute values for multiple jobs. Only some of the candidate jobs are selected based on the filtering criteria. A filter is a logical expression consisting of relations of attributes to attribute values. Among the specified jobs, only those whose attribute values match the filter expression are returned.

The **filter** command attribute functions the same way for the **pdq** command as for the **pdls** command.

headings=*Boolean*
Specify whether or not you want headings displayed for the requested attributes. *Boolean* can be: **true** (default) or **false**.

printer-name-requested=*PrinterName*
Specify the logical printer for which you want a list of queued jobs.

If you do not define a printer name, the **PDPRINTER** environment variable is used to determine the printer.

requested-attributes=*AttributeType*
Specify which output attributes you want displayed. The command-attribute value can be: **brief** (default), **verbose**, **"***attribute_list***"** , **all**, or **none**.

archive only displays attributes that can be set and specified.

style=*StyleName*
Specify the presentation format in which you want attributes displayed. *StyleName* can be **column** (default) or **line**.

Object Attributes

There are no object attributes for this command.

Arguments

Use the argument value to identify the specific object for which you want the status. If you specify multiple objects, each must be separated by spaces.

The argument value identifies the specific object the command applies to. If multiple objects are specified, each instance must be separated by one or more white spaces.

You can use the following argument values with the **pdq** command:

LocalJobId[.*DocNumber*]
Specify the jobs or documents that you want listed as determined by a local ID or a local ID and document number. If you specify multiple arguments on the command line, each must be separated by spaces.

GlobalJobId[.*DocNumber*]
Specify the job or documents you want listed as determined by a global ID or a global ID and document number. If you specify multiple arguments on the command line, each must be separated by spaces.

If you omit the **-p** *LogicalPrinterName* option, or the command-attribute **printer-name-registered=***PrinterName*, and the *LocalJobId* or *GlobalJobId* argument, all of the jobs in the queue associated with your defaults logical printer (defined in your **PDPRINTER** environment variable) are listed.

EXAMPLES

Query All Jobs on the Default Printer

To list all your jobs sent to your default logical printer LogPrt4, enter the command:

```
pdq
```

which will display information similar to the following:

```
                                              Intervening  Printer    Printers
Job  ID              Name    State            Jobs         Requested  Assigned
13   spl4:1104221000 File1   processing       0            LogPrt4    PhysPrt1
14   spl4:1105226030 JobA    pending          7            LogPrt4
15   spl4:1133000058 MyJob   pending          8            LogPrt4
8    sp15:075410002  File1   pending          LogPrt1
```

Note: Status information is displayed for all of your jobs in the queue associated with your default logical printer, not just those that were submitted to your default logical printer.

Query all Jobs on a Logical Printer

To list all jobs sent to logical printer dizzy, enter the command:

```
pdq -p dizzy
```

SEE ALSO

pdclean(1), pdcreate(1), pddelete(1), pddisable(1), pdenable(1), pdls(1), pdmod(1), pdpause(1), pdpr(1), pdpromote(1), pdresubmit(1), pdresume(1), pdrm(1), pdset(1), pdshutdown(1)

For information about:

- Filters, see the **pdls** command-attribute *filter*
- Headings, see the **pdls** command-attribute *style*
- Requested attributes, see the **pdls** command-attribute *style*
- Style, see the **pdls** command-attribute *style*

To view information about all supported attributes, see manpage *pd_att*(5). It contains a list of files by object from which you can select the attribute listing that you want.

STANDARDS CONFORMANCE

pdq: POSIX 1387.4

NAME

pdresubmit - resubmits previously submitted print jobs

SYNOPSIS

pdresubmit -h

pdresubmit [**-c** *ObjectClass*] [**-g**] [**-r** *RequestedAttributes*] [**-s** *StyleName*]
[**-x "***AttributeValuePairs***"**] [**-X** *AttributesFileName*] { *LocalJobId* ... | *GlobalJobId* ...
| [*ServerName***:**]*TargetLogicalPrinterName* ...
| [*ServerName***:**]*PrinterName* ... | [*ServerName***:**]*QueueName* ... }

DESCRIPTION

Use the **pdresubmit** command to request that an existing job be resubmitted to a specific logical printer. The logical printer can be in the same spooler or a different spooler. You can only resubmit jobs that have the current job state of held, pending, timed-out, or retained.

If the specified logical printer is in a new server (different spooler than where the job had been), the old server (spooler) resubmits the job with all of the current attributes assigned to the job to the new spooler. Any attributes that the old spooler defaulted are included as well, so that the new job remains as similar as possible to the old job. If the new spooler accepts the print job, it assigns a new global job identifier and the old global job identifier becomes invalid (see *Notes* below).

Logical printers and queues can also be arguments for this command. If a logical printer is the argument, all of the jobs that have been submitted to the old logical printer are resubmitted to the specified new logical printer. If a queue is the argument, all of the jobs in the old queue are resubmitted to the specified logical printer. All of the resubmitted jobs are validated again. If they cannot be supported by the newly specified logical printer they remain in the queue that they were in originally.

If the new logical printer cannot accept the job for some reason, an error message is issued and the job stays where it was.

Notes:

1. The global job identifier for the job is not changed if the job is resubmitted to a logical printer on the same server.
2. The global job identifier for the job is changed if the job is resubmitted to a logical printer on a different server. However, if you are the job-owner, you can still use the same local job identifier.

Options

You can use the following options with the **pdresubmit** command:

-c *ObjectClass*
Specify the object class you want for this command. The *ObjectClass* can be **job** (default), **queue**, or **printer**.

Within the valid classes, printer only applies to logical printers. This option is equivalent to specifying the command-attribute **class**.

-g Turn off headings. This option is equivalent to specifying the command-attribute **headings=false**.

-h Display a command-specific help message containing information about command syntax and options. This option is cannot be used with another option or with an attribute.

-r *RequestedAttributes*
Specify the attribute values you want displayed for the specified objects. Values can be **none** (default), **brief**, or **verbose**.

This option is equivalent to specifying the command-attribute **requested-attributes**.

-s *StyleName*
Specify the format in which you want the attributes displayed. *StyleName* can be **column** (default) or **line**.

This option is equivalent to specifying the command-attribute **style**.

-x "*AttributeValuePairs***"**
A single attribute string, consisting of one or more attribute-value pairs.

-X *AttributesFileName*
The name of a file containing attribute-value pairs to be inserted at the current point in the command line. This option is equivalent to specifying the command-attribute **attributes**.

Command Attributes

You can specify these attributes in a **-x "***AttributeValuePairs***"** string or in an attributes file designated with the **-X** *AttributesFileName* option.

attributes=*AttributesFileName*
Cause the designated attributes file to be read.

class=*ObjectClass*
Specify the *ObjectClass* that you want for this command. *ObjectClass* can be **job**(default), **queue**, or **printer**. Within the valid classes, **printer** only applies to logical printers.

headings=*Boolean*
Specify whether or not you want headings displayed for the requested attributes. The value can be **true** or **false**.

requested-attributes=*RequestedAttributes*
Specify the group of attributes that you want displayed for the job, queue, or printer.

ObjectAttribute can be **none**(default), **brief**, or **verbose**.

style=*StyleName*
Specify the format in which you want the attributes displayed. *StyleName* can be **column** or **line**.

Object Attributes

There are no object attributes for this command.

Arguments

Use the argument value to identify the specific object that you want to resubmit. If you specify multiple objects, all must be of the same class and each must be separated by one or more spaces on the command line.

You must use the following argument:

[*ServerName*:]*TargetLogicalPrinterName*
Specify the name of the new logical printer.

p

You must use one of the following arguments with the **pdresubmit** command:

LocalJobId or *GlobalJobId*
Specify the local or global job identifier for the print job to resubmit. The global job identifier can be for another user's print job if you are authorized to do so.

[*ServerName*:]*PrinterName*
Resubmit all jobs that are currently submitted to the indicated logical printer.

[*ServerName*:]*QueueName*
Resubmit all jobs that are currently in the specified queue.

EXAMPLES

1. To resubmit jobs 3828PP:1098 and 3828PP:1099 to logical printer localLP, enter:

```
pdresubmit localLP 3828PP:1098 3828PP:1099
```

2. To resubmit all jobs submitted to logical printer DeptLP1 to logical printer localLP, enter:

```
pdresubmit localLP -c printer DeptLP1
```

SEE ALSO

pdclean(1), pdcreate(1), pddelete(1), pddisable(1), pdenable(1), pdls(1), pdmod(1), pdpause(1), pdpr(1), pdpromote(1), pdq(1), pdresume(1), pdrm(1), pdset(1), pdshutdown(1)

STANDARDS CONFORMANCE

pdresubmit: POSIX 1387.4

NAME

pdresume - enables paused objects to resume operation

SYNOPSIS

pdresume -h

pdresume [**-c** *ObjectClass*] [**-m** "*MessageText*"] [**-x** "*AttributeValuePairs*"] [**-X** *AttributesFileName*] { *LocalJobId* ... | *GlobalJobId* ...
| *ServerName* ... | [*ServerName*:]*PrinterName* ...
| [*ServerName*:]*QueueName* ... }

DESCRIPTION

Use the **pdresume** command to start up (resume) paused jobs, physical printers, queues, or servers.

Note: The **pdresume** and **pdpause** commands are used to allow or prevent output from the object, whereas **pddisable** and **pdenable** are used to prevent or allow input to the object.

Options

Use the following options with the **pdresume** command:

-c *ObjectClass*
Specify the object class that you want to resume. The *ObjectClass* can be one of the following:

printer (default)
queue
job
server

Within the valid classes, **printer** is a physical printer and **server** is either a spooler or a supervisor. This option is equivalent to specifying the command-attribute **class**.

-h Display a command-specific help message containing information about command syntax and options. This option cannot be used with another option or with an attribute.

-m "*MessageText*"
Specify the message that is to be associated with the *ObjectClass* that is being resumed: **printer**, **queue**, **job**, or **server**. You can use this message to indicate the reason for resuming or to provide other comments.

When resuming a supervisor, HPDPS propagates the message to the **message** attribute of the physical printers residing in the supervisor. The **message** attribute for the supervisor is not changed.

When resuming a spooler, the message is propagated to the **message** attribute of the queues residing in the spooler. The **message** attribute for the spooler is not changed.

If the command operates on a **job**, you can list this message by specifying **requested-attributes=job-message-from-administrator** with the **pdls** command. If the command is to operate on a **printer**, **queue**, or **server**, you can list this message by specifying **requested-attributes=message** with the **pdls** command.

If the **-m** option is not specified, the message already stored with the object remains unchanged.

This option is equivalent to specifying the command-attribute **message**.

-x "*AttributeValuePairs*"
A single attribute string, consisting of one or more attribute-value pairs.

-X *AttributesFileName*
The name of a file containing attribute-value pairs you want inserted at the current point in the command line. This option is equivalent to specifying the command-attribute **attributes**.

Command Attributes

You can specify these attributes in a **-x** "*AttributeValuePairs*" string or in an attributes file designated with the **-X** *AttributesFileName* option.

attributes=*AttributesFileName*
Cause the designated attributes file to be read.

class=*ObjectClass*
Specify the object class that you want to resume. Valid object class names for the **pdresume**

p

command are: **printer** (default), **queue**, **job**, or **server**.

Within the valid classes, **printer** is a physical printer and **server** is either a spooler or a supervisor.

message="*MessageText***"**

Specify the message that you want associated with the **printer**, **queue**, **job**, or **server**. You may use this message to indicate the reason for resuming or to provide other comments.

When resuming a supervisor, HPDPS propagates the message to the **message** attribute of the physical printers residing in the supervisor. The **message** attribute for the supervisor is not changed.

When resuming a spooler, the message is propagated to the **message** attribute of the queues residing in the spooler. The **message** attribute for the spooler is not changed.

If the command operates on a job, you can list this message by specifying **requested-attributes=job-message-from-administrator** with the **pdls** command. If the command is to operates on **printer**, **queue**, or **server**, you can list this message by specifying **requested-attributes=message** with the **pdls** command.

If the **message** attribute is not specified, the message already stored with the object remains unchanged.

Object Attributes

There are no object attributes for this command.

Arguments

Use the argument values to specify the object that you want to start processing again (resume operating). If you specify multiple objects, all of the objects must be of the same class, and each must be separated by spaces.

You can use the following argument values with the **pdresume** command:

LocalJobID or *GlobalJobId*

Specify the job you want to resume as determined by a local job identifier or global job identifier. Only an administrator has the authority to resume jobs, and generally use the global job identifier.

A resumed job does not automatically return to its previous state. For example, the **job-hold** or **print-after** attributes of a paused job may have been modified, and returning to its previous state would cause the job to be held even after it has been resumed.

p

Resuming a job that was not previously assigned to a physical printer allows it to be scheduled.

Resuming a job that was processing when it was paused restores it to the pending state. An attempt is made to schedule it on the original physical printer. If the original printer is not available, HPDPS schedules the job on another physical printer that can support the job checkpoint format and begins printing the job where it stopped. If another physical printer cannot be found that supports the checkpoint format, the job is placed in the held state. The job can be resubmitted, which means that the job will be started from the beginning.

[*ServerName*:]*PrinterName*

Specify the printer that you want to resume operating.

Resuming a physical printer allows the printer to start processing jobs that have been assigned to the printer.

[*ServerName*:]*QueueName*

Specify the queue that you want to resume operating.

Resuming a queue resumes the distribution of jobs to physical printers associated with that queue.

ServerName

Specify the server on which you want the command to operate. A server does not have a paused state. Issuing the command to a spooler resumes all of the paused queues that reside in that spooler. The queues can then continue to distribute jobs to physical printers.

Issuing the command to a supervisor resumes all of the physical printers that reside in that supervisor. The physical printers are then allowed to start processing jobs that have been assigned to them.

EXAMPLES

Resume a Physical Printer

To resume physical printer HPLJ6P and issue a message, enter:

```
pdresume -m "Toner refilled" HPLJ6P
```

Resume a Queue

To resume the queue Div1Q2, enter:

```
pdresume -c queue  Div1Q2
```

To resume all of the queues in the spooler Mrk-spooler3, enter:

```
pdresume -c server Mrk-spooler3
```

SEE ALSO

pdclean(1), pdcreate(1), pddelete(1), pddisable(1), pdenable(1), pdls(1), pdmod(1), pdpause(1), pdpr(1), pdpromote(1), pdq(1), pdresubmit(1), pdrm(1), pdset(1), pdshutdown(1)

STANDARDS CONFORMANCE

pdresume: POSIX 1387.4

NAME

pdrm - removes print jobs

SYNOPSIS

pdrm -h

pdrm [**-m** **"***MessageText***"**] [**-r** *JobRetentionPeriod*] [**-x** **"***AttributeValuePairs***"**] [**-X** *AttributesFileName*] {*LocalJobId* ... | *GlobalJobId* ...}

DESCRIPTION

Use the **pdrm** command to remove (delete) previously submitted print jobs.

1. If the job you specify is currently printing, it can only be removed at a pausable point in the job. The pausable point at which the job can be removed is dependent on the type of printer being used to print the job. A pausable point may be immediate or it may be the next page, sheet, document, or job boundary. If there is no such point before the end of job, the job is not removed.

2. When you request a job to be removed with:

   - a **job-retention-period** of zero, the job is deleted.
   - a **job-retention-period** of non-zero, the job is retained until the **job-retention-period** runs out. Then the job is deleted.

   Note: The job-retention-period applies to both a previously set period or one specified with this command.

3. You can remove your own jobs, and if you are authorized, you can remove a job belonging to another person by specifying the global ID for the job.

   HPDPS issues a confirmation message prior to deleting jobs, unless the environment variable **PD_CONFIRM_DELETE** for the person requesting the job removal has a value of **no**.

Options

You can use the following options with the **pdrm** command:

-h Display a command-specific help message containing information about command syntax and options. This option cannot be used with another option or with an attribute.

-m "*MessageText***"**
Specify the message to be associated with the specified job. The specified text becomes the **job-message-from-administrator** attribute. You can use this message to indicate the reason for removing the job or to provide other comments.

You can list this message by specifying **requested-attributes=job-message-from-administrator** with the **pdls** command.

If the **-m** option is not specified, the message already stored with the job remains unchanged.

This option is equivalent to specifying the command-attribute **message**.

-r *JobRetentionPeriod*
Specify the period of time that you want the job retained by the server before the job is deleted. This is the retention period. If a retention period was previously specified for the job, this new retention period takes precedence over the previous job retention period.

This option is equivalent to specifying the object-attribute **job-retention-period**.

-x "*AttributeValuePairs***"**
A single attribute string, consisting of one or more attribute-value pairs.

-X *AttributesFileName*
The name of a file containing attribute-value pairs you want inserted at the current point in the command line. This option is equivalent to specifying the command-attribute **attributes**.

Command Attributes

You may specify these attributes in a **-x "***AttributeValuePairs***"** string or in an attributes file designated with the **-X** *AttributesFileName* option.

attributes=*AttributesFileName*
Cause the designated attributes file to be read.

message="*MessageText***"**
Specify the message that you want stored in the **job-message-from-administrator** attribute. You can use this message to indicate the reason for removing the job or to provide other comments.

If the **message** attribute is not specified, the message already stored with the job remains unchanged.

You can list this message by specifying **requested-attributes=job-message-from-administrator** with the **pdls** command.

Object Attribute

You may specify this attribute in a **-x "***AttributeValuePairs***"** string or in an attributes file designated with the **-X** *AttributesFileName* option.

job-retention-period=*time*
Specify the amount of time for the server to retain the job before the job is deleted. If a retention period was previously specified for the job, this new retention period takes precedence over the previous job retention period.

Arguments

Use the argument to specify the job to delete.

You can use the following arguments with the **pdrm** command:

LocalJobID
Specify the local ID of the job that you want removed.

This argument must appear last in the command. If you want to cancel more than one job, each identifier must be separated by spaces.

GlobalJobId
Specify the global ID of the job that you want removed.

This argument must appear last in the command. If you want to cancel more than one job, each identifier must be separated by spaces.

EXAMPLES

Remove A Job

To remove your job number 15, enter the command:

```
pdrm 15
```

To remove global job number Super1:1011223002 on the spooler Super1, enter the command:

```
pdrm Super1:1011223002
```

Remove Job Regardless of Retention Time

To remove your job number 10 as soon as possible, regardless of any previously specified **job-retention-period** attribute value, enter one of the following commands:

```
pdrm -x job-retention-period=0 10
```

or

```
pdrm -r 0 10
```

SEE ALSO

pdclean(1), pdcreate(1), pddelete(1), pddisable(1), pdenable(1), pdls(1), pdmod(1), pdpause(1), pdpr(1), pdpromote(1), pdq(1), pdresubmit(1), pdresume(1), pdset(1), pdshutdown(1).

STANDARDS CONFORMANCE

pdrm: POSIX 1387.4

NAME

pdset - define the attribute values of the specified print object

SYNOPSIS

pdset -h

pdset [**-c** *ObjectClass*] [**-g**] [**-m "***MessageText***"**] [**-r "***RequestedAttributes***"**] [**-s** *StyleName*] [**-x "***AttributeValuePairs***"**] [**-X** *AttributesFileName*]
{ *ServerName* ...
  | *ServerName***:***InitialValueDocumentName* ...
  | *ServerName***:***InitialValueJobName* ...
  | *ServerName***:***LogName* ...
  | [*ServerName***:**]*PrinterName* ...
  | [*ServerName***:**]*QueueName* ...
  | *LocalJobId* ...
  | *GlobalJobId* ... }

DESCRIPTION

Use the **pdset** command to define or modify the values of the print-object attributes. Printers must be disabled before their attributes can be modified.

Table 1-1 lists the four modification operators:

Table 1-1. Four Modification Operators:

Operator	Syntax	Description
replace	*attribute=value*	Replaces the entire value of the attribute *attribute* with value or, if not already present, adds the attribute-value pair to the job.
add-values	*attribute+=value*	Adds the value *value* to the attribute *attribute*. You cannot add values to single-valued attributes. An add request that duplicates values on a multi-valued attribute has no effect on the job.
remove-values	*attribute-=value*	Removes the values *value* from the attribute *attribute*. A remove request for a nonexistent value has no effect on the object. A remove request for the last or only value of an attribute is equivalent to a set-to-default request.
reset-to-default	*attribute==*	Set the attribute *attribute* to the default values according to the job defaulting-hierarchy. If values are supplied with a reset request, they are ignored.

If you do not specify a value with a replace, add, or remove request, an error is issued and the request to change the object is rejected.

Only values for settable attributes can be changed using the **pdset** command. Using this command for non-settable attributes results in an error. This is determined on an object-by-object basis.

Changes made to objects are permanent and remain changed even if the print system is shutdown and then restarted.

Options

You can use the following options with the **pdset** command:

-h Display a command-specific help message containing information about command syntax and options. This option cannot be used with another option or with an attribute.

-c ObjectClass
Specify the object class that the command is to operate upon. The *ObjectClass* can be one of the following:

printer (default)
job
server
queue

p

document
initial-value-job
initial-value-document
log

Within the valid classes, **printer** is for a logical printer or a physical printer and **server** is for a spooler or a supervisor.

This option is equivalent to specifying the command-attribute **class**.

-g Turn off headings. This option is equivalent to specifying the command-attribute **headings=false**.

-m "*MessageText***"**
Specify the message you want associated with the **printer**, **job**, **server**, **queue**, **document**, **initial-value-job**, **initial-value-document**, or **log** that is being modified.

You can use the message to give the reason for the modification or to provide other comments. If you do not specify the **-m** option, the message already stored with the object remains unchanged.

If the command is for a **job**, you can list this message by specifying **requested-attributes=job-message-from-administrator** with the **pdls** command.

If the command is for the other objects, you can list this message by specifying **requested-attributes=message** with the **pdls** command.

This option is equivalent to specifying the command-attribute **message**.

-r *RequestedAttributes*
Specify the attribute values that you want to display for the specified objects.

RequestedAttributes can be **none** (default), **brief**, or **verbose**.

This option is equivalent to the command-attribute **requested-attributes**.

-s *StyleName*
Specify the style in which you want the attributes displayed.

StyleName may be **column** (default), or **line**.

This option is equivalent to the command-attribute **style**.

-x "*AttributeValuePairs***"**
A single attribute string, consisting of one or more attribute-value pairs. Prefix the attribute value with the replace operator, **=** , to replace a value, the add-value operator, **+=** , to add a value, and the remove-value operator, **-=** , to remove a value. Use the reset-to-defaul operator, **==** , with no attribute value to set the attribute to its default value.

-X *AttributesFileName*
The name of a file containing attribute-value pairs to be inserted at the current point in the command line. This option is equivalent to the command-attribute **attributes**.

Command Attributes

You can specify these attributes in a **-x "***AttributeValuePairs***"** string or in an attributes file designated with the **-X** *AttributesFileName* option.

attributes=*AttributesFileName*
Cause the designated attributes file to be read.

class=*ObjectClass*
Specify the *ObjectClass* that you want for this command. Valid object class names are: **printer** (default), **job**, **server**, **queue**, **document**, **initial-value-job**, **initial-value-document**, and **log**.

headings=Boolean
Boolean can be **true** (default) or **false.** Specifies if you want headings displayed.

message="*MessageText***"**
Specify the message that you want associated with the **printer**, **job**, **server**, **queue**, **document**, **initial-value-job**, **initial-value-document**, or **log** that is being modified.

You can use the message to give the reason for the modification or to provide other comments. If you do not specify the **message** attribute, the message already stored with the object remains

p

unchanged.

If the command is for a **job**, you can list this message by specifying **requested-attributes=job-message-from-administrator** with the **pdls** command.

If the command is for the other objects, you can list this message by specifying **requested-attributes=message** with the **pdls** command.

requested-attributes=*AttributeType*
Specify which attributes you want displayed. *AttributeType* can be **none** (default), **brief**, or **verbose**.

style=*StyleName*
Specify the presentation format that you want for the displayed output. *StyleName* can be **column** (default) or **line**.

Object Attributes

There are no object attributes for this command.

Arguments

Use the arguments to identify the specific objects that you want to set or modify. If you specify multiple objects, each must be separated by spaces. You can use the following argument values with the **pdset** command:

ServerName

*ServerName***:***InitialValueDocumentName*

*ServerName***:***InitialValueJobName*

*ServerName***:***LogName*

[*ServerName***:**] *PrinterName*

[*ServerName***:**] *QueueName*

LocalJobId

GlobalJobId

EXAMPLES

Specifying a Descriptor for a Server

To set a description for the supervisor SUPER1 and identify the processor name where it is located and the TCP/IP address for the processor, enter the command:

```
pdset -c server -x "descriptor='Supervisor SUPER1 is installed on
cowboy.  The TCP/IP address for cowboy is 9.99.9.143.'" SUPER1
```

Specifying a List of Managers for a Server

To identify the people who are responsible for the server and how to contact them, enter the command:

```
pdset -c server -x "list-of-managers='John Smith, extension 4488,
office 3-10' 'Mary Jones, extension 5562, office 110'" SUPER1
```

Change the Operator to Receive Messages

To identify the new person that is to receive start and stop messages for physical printer PhysPrt1, enter the command:

```
pdset -c printer -x "notify-operator=electronic-mail:ro@cowboy"
SUPER1:PhysPrt1
```

Set the Job Size Range

To set the size of jobs that can submitted to a physical printer PhysPrt2, enter the command:

```
pdset -x "job-size-range-supported=0:10000 job-size-range-
ready=0:10000" PhysPrt2
```

To set a Message

To set the message for server SUPER2, enter:

```
    pdset -c server -m "printing system now ok" SUPER2
```

SEE ALSO

pdclean(1), pdcreate(1), pddelete(1), pddisable(1), pdenable(1), pdls(1), pdmod(1), pdpause(1), pdpr(1), pdpromote(1), pdq(1), pdresubmit(1), pdresume(1), pdrm(1), pdset(1), pdshutdown(1)

For information about:

- Headings, see the **`pdls`** command-attribute **`style`**.
- Style, see the **`pdls`** command-attribute, **`style`**.

To view information about all supported attributes, see manpage *pd_att*(5). It contains a list of files by object from which you can select the attribute listing that you want.

STANDARDS CONFORMANCE

`pdset`: POSIX 1387.4

NAME

pdshutdown - stops servers

SYNOPSIS

pdshutdown -h

pdshutdown [**-c** *Server*] [**-m** "*MessageText*"] [**-w** *WhenTime*] [**-x** "*AttributeValuePairs*"] [**-X** *AttributesFileName*] *ServerName*

DESCRIPTION

Use the **pdshutdown** command to terminate a server process.

A server can be shut down either immediately or after it has finished processing some or all of its current jobs.

While a server is shutting down and after it has shut down, the server cannot accept new jobs. Printers that were enabled at shutdown time will be enabled after restart. Printers that were disabled at shutdown time will be disabled after restart.

A confirmation message may be issued, depending on the value of the **server-notification-profile** attribute for the specified server when the shutdown is complete.

Note: To restart a server once shutdown, you need to use the **pdstartspl** or **pdstartsuv** utility. See *pdstartspl*(1M) and *pdstartsuv*(1M).

Options

You can use the following options with the **pdshutdown** command:

-h Display a command-specific help message containing information about command syntax and options. This option cannot be used with another option or with an attribute.

-c *server*
The only supported class for this command is **server**.

-m "*MessageText*"
Specify the message to be associated with the specified server being shut down. You can use this message to indicate the reason for shutting down the server or to provide other comments.

You can list this message by specifying **requested-attributes=message** with the **pdls** command.

If the **-m** option is not specified, the message already stored with the object remains unchanged.

This option is equivalent to specifying the command-attribute **message**.

-w *WhenTime*
Specify when the you want the shutdown to occur. *WhenTime* can be **after-current** (default), **now**, or **after-all**.

now If the server is a **Supervisor**, the result of the command is to abort any currently printing jobs, then shut down the supervisor. These jobs will be rescheduled to the next available printer if the spooler is still active. If the spooler has been shutdown, the job is aborted. When the spooler and supervisor become active again, the job will have to be submitted again.

If the server is a **Spooler**, the spooler shuts down immediately.

after-current (default)
If the server is a **Supervisor**, The supervisor continues to accept requests other than print requests until the currently-printing jobs finish printing. Then the supervisor shuts down.

If the server is a **Spooler**, the spooler shuts down immediately.

after-all
The server (spooler or supervisor) continues to accept all requests except print requests until all scheduled jobs finish printing. Then the server shuts down.

This option is equivalent to specifying the command-attribute **when**.

-x "*AttributeValuePairs*"
A single attribute string, consisting of one or more attribute-value pairs.

-X *AttributesFileName*
The name of a file containing attribute-value pairs you want inserted at the current point in the

command line. This option is equivalent to specifying the command-attribute **attributes**.

Command Attributes

You may specify these attributes in a **-x "***AttributeValuePairs***"** string or in an attributes file designated with the I -X AttributesFileName option.

attributes=*AttributesFileName*
Cause the designated attributes file to be read.

class=*server*
The only supported class for this command is **server**.

message="*MessageText***"**
Specify the message that you want associated with the server being shutdown. You can use this message to indicate the reason for shutting down the server or to provide other comments.

If the **message** attribute is not specified, the message already stored with the object remains unchanged.

You can list this message by specifying **requested-attributes=message** with the **pdls** command.

when=*WhenTime*
Specify when the you want the shutdown to occur. *WhenTime* can be **after-current** (default), **now**, or **after-all**.

now If the server is a **Supervisor**, the result of the command is to abort any currently printing jobs, then shut down the supervisor. These job will be rescheduled to the next available printer if the spooler is still active. If the spooler has been shutdown, the job is aborted. When the spooler and supervisor become active again, the job will have to be submitted again.

If the server is a **Spooler**, the spooler shuts down immediately.

after-current (default)
If the server is a **Supervisor**, The supervisor continues to accept requests other than print requests until the currently-printing jobs finish printing. Then the supervisor shuts down.

If the server is a **Spooler**, the spooler shuts down immediately.

after-all
The server (spooler or supervisor) continues to accept all requests except print requests until all scheduled jobs finish printing. Then the server shuts down.

Object Attribute

There are no object attributes for this command.

Arguments

You must use the following argument value with the **pdshutdown** command:

ServerName Specify the server to shut down.

EXAMPLES

Shutdown a Server

To shut down supervisor SUPER1, enter the following command. This command also aborts all print jobs that are currently printing and sets a message, "Down for maintenance".

```
pdshutdown -w now -m "Down for maintenance" SUPER1
```

SEE ALSO

pdstartspl(1M), pdstartsuv(1M)

STANDARDS CONFORMANCE

pdshutdown: POSIX 1387.4

NAME

pg - file perusal filter for soft-copy terminals

SYNOPSIS

pg [**-***number*] [**-p***string*] [**-cefnrs**] [**+***linenumber*] [**+/***pattern*] [*file* ...]

Remarks

pg and **more** are both used in similar situations (see *more*(1)). Text highlighting features supported by **more** are not available from **pg**. However, **pg** has some useful features not provided by **more**.

DESCRIPTION

pg is a *text file* filter that allows the examination of *files* one screenful at a time on a soft-copy terminal. If **-** is used as a *file* argument, or **pg** detects NULL arguments in the comand line, the standard input is used. Each screenful is followed by a prompt. To display a new page, press **Return**. Other possibilities are enumerated below.

This command is different from other paginators such as **more** in that it can back up for reviewing something that has already passed. The method for doing this is explained below.

In order to determine terminal attributes, **pg** scans the **terminfo** data base for the terminal type specified by the environment variable **TERM** (see *terminfo*(4)). If **TERM** is not defined, terminal type **dumb** is assumed.

Options

pg recognizes the following command line options:

-*number*	*number* is an integer specifying the size (in lines) of the window that **pg** is to use instead of the default (on a terminal containing 24 lines, the default window size is 23).
-p *string*	Causes **pg** to use *string* as the prompt. If the prompt string contains a **%d**, the first occurrence of **%d** in the prompt is replaced by the current page number when the prompt is issued. The default prompt string is a colon (**:**).
-c	Home the cursor and clear the screen before displaying each page. This option is ignored if **clear_screen** is not defined in the **terminfo** data base for this terminal type.
-e	Causes **pg** to *not* pause at the end of each file.
-f	Normally, **pg** splits lines longer than the screen width, but some sequences of characters in the text being displayed (such as escape sequences for underlining) generate undesirable results. The **-f** option inhibits **pg** from splitting lines.
-n	Normally, commands must be terminated by a new-line character. This option causes an automatic end-of-command as soon as a command letter is entered.
-r	Restricted mode. The shell escape is disallowed. **pg** will print an error message but does not exit.
-s	Causes **pg** to print all messages and prompts in standout mode (usually inverse video).
+*linenumber*	Start display at *linenumber*.
+/*pattern***/**	Start up at the first line containing text that matches the regular expression *pattern*.

pg looks in the environment variable **PG** to preset any flags desired. For example, if you prefer to view files using the **-c** mode of operation, the Bourne-shell command sequence **PG='-c' ; export PG** or the C-shell command **setenv PG -c** causes all invocations of **pg**, including invocations by programs such as **man** and **msgs**, to use this mode. The command sequence to set up the **PG** environment variable is normally placed in the user **.profile** or **.cshrc** file. No form of quoting is provided, so the string and pattern arguments are limited to single word.

The responses that can be typed when **pg** pauses can be divided into three categories: those causing further perusal, those that search, and those that modify the perusal environment.

Commands that cause further perusal normally take a preceding *address*, an optionally signed number indicating the point from which further text should be displayed. This *address* is interpreted either in pages or lines, depending on the command. A signed *address* specifies a point relative to the current page

p

or line; an unsigned *address* specifies an address relative to the beginning of the file. Each command has a default address that is used if none is provided.

Perusal commands and their defaults are as follows:

(+1)<*newline*> or <*blank*>
: Displays one page. The address is specified in pages.

(+1) **l** — With a relative address, **pg** simulates scrolling the screen, forward or backward, the number of lines specified. With an absolute address **pg** prints a screenful beginning at the specified line.

(+1) **d** or **^D** — Simulates scrolling a half-screen forward or backward.

*i***f** — Skip *i* screens of text.

*i***z** — Same as newline except that *i* , if present, becomes the new default number of lines per screenful.

The following perusal commands take no *address*:

. or **^L** — Typing a single period causes the current page of text to be redisplayed.

$ — Displays the last windowful in the file. Use with caution when the input is a pipe.

The following commands are available for searching for text patterns in the text. The Basic Regular Expression syntax (see *regexp*(5)) is supported. The terminal **/**, **^**, or **?** can be omitted from the pattern search commands. Regular expressions must always be terminated by a new-line character, even if the **-n** option is specified.

*i***/***pattern***/** — Search forward for the *i*th (default *i*=1) occurrence of *pattern*. Searching begins immediately after the current page and continues to the end of the current file, without wrap-around.

*i***^***pattern***^**
*i***?***pattern***?** — Search backwards for the *i*th (default *i*=1) occurrence of *pattern*. Searching begins immediately before the current page and continues to the beginning of the current file, without wrap-around. The **^** notation is useful for Adds 100 terminals which cannot properly handle the **?**.

After searching, **pg** normally displays the line found at the top of the screen. This can be modified by appending **m** or **b** to the search command to leave the line found in the middle or at the bottom of the window from now on. The suffix **t** can be used to restore the original situation.

pg users can modify the perusal environment with the following commands:

*i***n** — Begin perusing the *i*th next file in the command line. The *i* is an unsigned number, default value is 1.

*i***p** — Begin perusing the *i*th previous file in the command line. *i* is an unsigned number, default is 1.

*i***w** — Display another window of text. If *i* is present, set the window size to *i*.

s *filename* — Save the input in the named file. Only the current file being perused is saved. The white space between the **s** and *filename* is optional. This command must always be terminated by a new-line character, even if the **-n** option is specified.

h — Help by displaying an abbreviated summary of available commands.

q or **Q** — Quit *pg*.

!*command* — *command* is passed to the shell, whose name is taken from the **SHELL** environment variable. If this is not available, the default shell is used. This command must always be terminated by a new-line character, even if the **-n** option is specified.

At any time when the output is being sent to the terminal, the user can press the quit key (normally CTRL-\), the interrupt (break) key or the DEL key. This causes **pg** to stop sending output, and display the prompt. The user may then enter one of the commands in the normal manner. Unfortunately, some output is lost when this is done, due to the fact that any characters waiting in the terminal's output queue are flushed when the quit signal occurs.

If the standard output is not a terminal, **pg** is functionally equivalent to **cat** (see *cat*(1)), except that a header is printed before each file if more than one file is specified.

EXTERNAL INFLUENCES

Environment Variables

LC_COLLATE determines the collating sequence used in evaluating regular expressions.

LC_CTYPE determines the interpretation of text as single and/or multi-byte characters, and the characters matched by character class expressions in regular expressions.

LANG determines the language in which messages are displayed.

If **LC_COLLATE** or **LC_CTYPE** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**. If any internationalization variable contains an invalid setting, **pg** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported.

EXAMPLES

To use **pg** when reading system news:

```
news | pg -p "(Page %d):"
```

WARNINGS

If terminal tabs are not set every eight positions, undesirable results may occur.

When using **pg** as a filter with another command that changes the terminal I/O options (such as *crypt*(1)), terminal settings may not be restored correctly.

While waiting for terminal input, **pg** responds to **BREAK**, **DEL**, and **^** by terminating execution. Between prompts, however, these signals interrupt *pg*'s current task and place the user in prompt mode. These should be used with caution when input is being read from a pipe, because an interrupt is likely to terminate the other commands in the pipeline.

Users of **more** will find that the **z** and **f** commands are available, and that the terminal **/**, **^**, or **?** can be omitted from the pattern search commands.

FILES

/usr/share/lib/terminfo/?/*	terminal information data base
/tmp/pg*	temporary file when input is from a pipe

p

SEE ALSO

crypt(1), grep(1), more(1), terminfo(4), environ(5), lang(5), regexp(5).

STANDARDS CONFORMANCE

pg: SVID2, SVID3, XPG2, XPG3

NAME

pppd - PPP daemon

SYNOPSIS

pppd [**options...**]

DESCRIPTION

pppd is a daemon process used in UNIX systems to manage connections to other hosts using **PPP**, the **Point to Point Protocol**, or **SLIP**, the **Serial Line Internet Protocol**. It uses the UNIX host's native serial ports. It communicates with the UNIX kernel's own TCP/IP implementation via the HP IP tunnel driver (see *tun*(4)).

Daemon Management Options

auto Start in 'autocall' mode and detach from the controlling terminal to run as a daemon. Initiate a connection in response to a packet specified in the 'bringup' category in filter-file. Requires the remote address.

up When used with **auto**, bring the link up immediately rather than waiting for traffic. If the link goes down, attempt to restart it (after the call retry delay timer expires) without waiting for an outbound packet.

dedicated Treat the connection as a dedicated line rather than a demand-dial connection. This option tells **pppd** to never give up on the connection; that is, if the peer tries to shut down the link, go ahead and do so, but then immediately try to reestablish the connection. Similarly, when first trying to connect, **pppd** will not give up after sending a fixed number of Configure-Request messages. Hangup events (LQM failures, loss of Carrier Detect) will still cause the device to be closed, just as with dial-up connections, and the **Systems** file will then be checked for alternate entries. If none are available, the connection will be reestablished after the call retry delay timer expires. Use a short call retry delay timer on dedicated circuits; something like **Any;5-30** should work well. Implies **up**.

nodetach Don't detach from the controlling terminal in 'autocall' mode. When used with **log -**, this can be useful for watching the progress of the PPP session.

log *log-file* Append logging messages to log-file (default: **/var/adm/pppd.log**).

acct *acct-file* Append session accounting messages to acct-file. If acct-file is the same as log-file, the session accounting messages will be interleaved with other logging information.

filter *filter-file* Look in filter-file for packet filtering and link management information (default: **/etc/ppp/Filter**).

debug *debug-level* Set the log file verbosity to the following debug-level and each debugging verbosity level also provides the information of all the lower-numbered levels.

0	Daemon start messages
1	Link status messages, calling attempts (the default)
2	Chat script processing, input framing errors
3	LCP, IPCP, PAP and CHAP negotiation
4	LQM status summaries
5	IP interface changes
6	IP message summaries
7	Full LQM reports
8	All PPP messages (without framing)
9	Characters read or written
10	Procedure call messages
11	Internal timers

exec *exec-cmd*
Run 'exec-cmd **up** addr args' when the link comes up, and 'exec-cmd **down** addr args' when it goes down. *Addr* is the IP address of the peer, and *args* is the list of arguments given to **pppd**.

nonice
Run at a normal user process priority, rather than using the nice() library routine to elevate **pppd** scheduling priority to -10.

Communications Options

asyncmap *async-map*
Set the desired Async Control Character Map to async-map, expressed in C-style hexadecimal notation (default 0xA0000).

noasyncmap
Disable LCP Async Control Character Map negotiation.

escape *odd-character*
In addition to those characters specified in the PPP Async Control Character Map (which can include only 0x00 through 0x1F), also apply the escaping algorithm when transmitting odd-character. The value of odd-character must be between 0x00 and 0xFF, and cannot be any of 0x5E, 0x7D or 0x7E.

Odd-character can be specified as a decimal number, in C-style hexadecimal notation, or as an ASCII character with optional '^' control-character notation. For example, the XON character could be specified as 17, 0x11, or ^Q.

If a character specified with the **escape** argument, when transformed into its escaped form, would be the same as a character contained in the peer's negotiated Async Control Character Map, a warning will be printed in the log file and the character specified on the command line will not be escaped.

If a character specified with the **escape** argument, when transformed into its escaped form, would be the same as a character specified in another **escape** argument on the daemon's command line, **pppd** will print an error message and exit.

device
Communicate over the named device (default **/dev/tty**).

comm-speed
Set communications rate to comm-speed bits per second.

ignore-cd
Ignore the state of the CD (Carrier Detect, also called DCD, Data Carrier Detect) signal. This is useful for systems that don't support CD but want to run PPP over a dedicated line.

xonxoff
Set the line to use in-band ('software') flow control, using the characters DC3 (^S, XOFF, ASCII 0x13) to stop the flow and DC1 (^Q, XON, ASCII 0x11) to resume. (The default is to use no flow control.) For an outbound connection, this may be specified either in **Devices** or on the **pppd** command line.

telnet
When used on an answering **pppd** command line, negotiate the telnet binary option and understand telnet escape processing. Not for use with **device** or **auto**.

Link Management Options

nooptions
Disable all LCP and IPCP options.

noaccomp
Disable HDLC Address and Control Field compression.

noprotcomp
Disable LCP Protocol Field Compression.

slip
Use RFC 1055 SLIP packet framing rather than PPP packet framing. Disables all option negotiation, and implies **noasyncmap**, **noipaddress**, **vjslots 16**, **novjcid**, **nomagic**, **nomru**, and **mru 1006**. Implies **vjcomp** if peer sends a header-compressed TCP packet.

extra-slip-end
When running in SLIP mode, prepend a SLIP packet framing character (0xC0) to each frame before transmission, even if this frame immediately follows the previous frame. By default, **pppd** transmits only one framing character between adjacent SLIP frames.

nomagic
Disable LCP Magic Number negotiation.

mru *mru-size*
Set LCP Maximum Receive Unit value to mru-size for negotiation. The default is 1500 for PPP and 1006 for SLIP.

nomru	Disable LCP Maximum Receive Unit negotiation, and use 1500 for our interface.
active	Begin LCP parameter negotiation immediately (the default).
passive	Do not send our first LCP packet until we receive an LCP packet from the peer.
timeout *restart-time*	Set the LCP, IPCP, CCP, PAP, and CHAP option negotiation restart timers to restart-time (default 3 seconds).
lqrinterval *time*	Send Link-Quality-Reports or Echo-Requests every *time* seconds (default 10 seconds). If the peer responds with a Protocol-Reject, send LCP Echo-Requests every *time* seconds instead, and use the received LCP Echo-Replies for link status policy decisions.
lqthreshold *min/per*	Set a minimum standard for link quality by considering the connection to have failed if fewer than *min* out of the last *per* LQRs we sent have been responded to by the peer (default 1/5).
echolqm	Use LCP Echo-Requests rather than standard Link-Quality-Report messages for link quality assessment and policy decisions. The peer can override this if it actively tries to configure Link Quality Monitoring unless the **nolqm** parameter is also specified.
nolqm	Don't send or recognize Link-Quality-Report messages. If **echolqm** is also specified, Echo-Request messages will be used to detect link failures.

idle *idle-time*[**/***session-idle-time*]

Shut down the link when idle-time seconds pass without receiving or transmitting a packet specified in the 'keepup' category in the filter file (default is to never consider the link idle).

If session-idle-time is specified and any TCP sessions are open, shut down the link when session-idle-time seconds pass without receiving or transmitting a packet.

max-configure *tries*	Set the PPP Max-Configure counter (the maximum number of Configure-Requests sent without a response) to *tries* .
max-terminate *tries*	Set the PPP Max-Terminate counter (the maximum number of Terminate-Requests sent without a response) to *tries* .
max-failure *tries*	Set the PPP Max-Failure counter (the maximum number of Configure-Naks sent without a positive response) to *tries* .

IP Options

*local***:***remote*

The address of this machine, followed by the expected address for the remote machine. Can be specified either as symbolic names or as literal IP addresses, if their addresses cannot be discovered locally without using the PPP link.

Both addresses are optional, but a colon by itself is not valid, and the remote address is required when running as a daemon in 'autocall' mode. If only *local*: is specified when receiving an incoming call, the remote address will be discovered during IPCP IP-Address negotiations.

If either address is followed by a tilde character ('˜'), or if the tilde appears alone, **pppd** accepts the IP address given by the peer during IPCP negotiations, whether for the local end or the peer's end of the link. (not available in SLIP mode)

Because SLIP cannot perform option negotiations, including IPCP, both addresses should normally be specified, and the tilde option is unavailable. To obtain a similar "feature", the peer must provide the IP address textually during the login process, and a new value must be obtained using the Systems file '\A' chat script feature (see *ppp.Systems*(4)).

netmask *subnet-mask*

Set the subnet mask of the interface to subnet-mask, expressed either in C-style hexadecimal (e.g. 0xffffff00) or in decimal dotted-quad notation (e.g. 255.255.255.0). The default subnet mask will be appropriate for the network (class A, B, or C), assuming no subnetting.

noipaddress	Disable IPCP IP-Address negotiation.
vjcomp	Enable RFC 1144 'VJ' Van Jacobson TCP header compression negotiation with 16 slots and slot ID compression (this is the default with PPP framing). 'VJ' compression is enabled by default for async connections, and disabled by default for sync connections.
novjcomp	Disable RFC 1144 'VJ' Van Jacobson TCP header compression (this is the default with SLIP framing, until the peer sends a header-compressed TCP packet).
vjslots *vj-slots*	Set the number of VJ compression slots (min 3, max 256, default 16).
novjcid	Disable VJ compression slot ID compression (enabled by default).
rfc1172-vj	Backwards compatibility with older PPP implementations (4-byte VJ configuration option), but with the correct option negotiation value of 0x002d.
rfc1172-typo-vj	Backwards compatibility with older PPP implementations (4-byte VJ configuration option) that conform to the typographical error in RFC 1172 section 5.2 (Compression-Type value 0x0037).
rfc1172-addresses	Backwards compatibility with older PPP implementations that conform to RFC 1172 section 5.1 (IP-Addresses, IPCP configuration option 1) and not with the newer RFC 1332 (IP-Address, IPCP configuration option 3), but that respond with something besides a Configure-Reject when they receive an IPCP Configure-Request containing an option 3.

Authentication Options

requireauth	Require either PAP or CHAP authentication.
requirechap	Require CHAP authentication as described in RFC 1334.
requiremschap	Require MS-CHAP authentication.
requirepap	Require PAP authentication.
rechap *interval*	Demand that the peer re-authenticate itself (using CHAP) every interval seconds. If the peer fails the new challenge, the link is terminated.
name *identifier*	Provide the identifier used during PAP or CHAP negotiation. This option is necessary if the PPP peer requires authentication. The default value is the value returned by the *gethostname*(2) system call or the *hostname*(1) command.

MicroSoft Compatibility Options

ms-dns *address*	Set the MS DNS address to provide to the peer. First occurrence of this option on the command line sets the primary address; the second occurrence sets the secondary address.
ms-nbns *address*	Set the MS NBNS address to provide to the peer. First occurrence of this option on the command line sets the primary address; the second occurrence sets the secondary address.

Encryption Options

Encryption is not currently available in software exported from the USA. However, customer may contact sales@progressive-systems.com to obtain encryption functionality.

Link Compression Options

compress	Offer all supported link compression types (currently only Predictor-1) when negotiating. The default is to propose and accept no link compression type.
compress-pred1	Accept any supported compression type, but prefer Predictor type 1 compression.
nopred1	Never use Predictor-1 compression.

LOG FILE

Status information is recorded in the log file (**/var/adm/pppd.log** by default) by each copy of **pppd** running on a single machine. Each line in the file consists of a message preceded by the date, the time, and the process ID number of the daemon writing the message. The quantity and verbosity of messages are controlled with the **debug** option and with the **log** filter (see *ppp.Filter*(4)).

Each packet that brings up the link (at debug level 1 or more), each packet that matches the **log** filter (at any debug level), or any packet when the debug level is 7 or more writes a one-line description of the packet to the log file. The first item of the message is the protocol (**tcp**, **udp**, **icmp**, or a numeric protocol value). For ICMP packets, the keyword **icmp** is followed by the ICMP message type and sub code, separated by slashes. After the protocol comes an IP address and optionally a TCP or UDP port number, followed by an arrow indicating whether the packet was sent (->) or received (<-), followed by another address and port number, followed by the length of the packet in bytes before VJ TCP header compression, followed by zero or more keywords. For transmitted packets, the first IP address is the source address, while for received packets, the first IP address is the destination address. Well known TCP and UDP port numbers will be replaced by the name returned by the **getservbyport()** library function. The keywords and their meanings are:

frag	The packet is a middle or later part of a fragmented IP frame.
syn	The packet has the TCP SYN bit set.
fin	The packet has the TCP FIN bit set.
bringup	The transmitted packet matches the **bringup** filter and is bringing up the link.
!keepup	the packet has been rejected by the **keepup** filter.
!pass	The packet has been rejected by the **pass** filter.
dial failed	The packet was dropped because **pppd** is waiting for the call retry timer to expire.
(c)	The received packet is VJ TCP header compressed.
(u)	The received packet is VJ TCP header uncompressed.

For example, the following log file line

```
9/6-14:06:26-83 tcp 63.1.6.3/1050 -> 8.1.1.9/smtp 44 syn
```

indicates that at 2:06:26 PM on September 6, process ID 83 sent a 44-byte TCP packet with the SYN bit set from port 1050 on 63.1.6.3 to the SMTP port on 8.1.1.9.

SIGNALS

Upon reception of the following signals, **pppd** closes and reopens the log file, re-reads the filter and key files, then takes the indicated actions:

SIGKILL	Don't use this. Never, never use this. Since **pppd** won't be able to shut down gracefully, it will leave your serial interfaces (whether **/dev/tty**) and your IP tunnel driver in some unknown state. Use SIGTERM instead, so **pppd** will shut down cleanly, and leave the system in a well-defined state.
SIGINT	Disconnect gracefully from an active session. If in 'autocall' mode, reset the call retry delay timer and call retry backoff interval. If **up** was specified, attempt to re-establish the link. Exit if not in 'autocall' mode.
SIGHUP	Disconnect abruptly from an active session. If **up** was specified, attempt to re-establish the link. Exit if not in 'autocall' mode.
SIGTERM	Disconnect gracefully from an active session, clean up the state of any serial and IP interfaces that are open, then exit.
SIGUSR1	Increment the verbosity level for debugging information written to the log file.
SIGUSR2	Reset the debugging verbosity level to the base value (1 unless **debug 0** was supplied on the command line).
SIGALRM	Take no action except to re-read the filter and key files.

EXAMPLE

To run a pair of daemons on 'oursystem', one maintaining a constant link with 'backbonesystem' and the other prepared to initiate outbound calls to a neighboring machine named 'theirsystem', add the following to **/sbin/rc2.d/S522ppp**:

```
if [ -f /etc/ppp/Autostart ]; then
     /etc/ppp/Autostart
fi
```

p

Then make **/etc/ppp/Autostart** look like this:

```
#!/bin/sh

PATH=/usr/etc:/bin:/usr/bin

if [ -f /var/adm/pppd.log ]; then
        mv /var/adm/pppd.log /var/adm/OLDpppd.log
fi

echo -n "Starting PPP daemons:"        >/dev/console

pppd oursystem:backbonesystem auto up
        (echo -n ' backbonesystem')    >/dev/console
pppd oursystem:theirsystem auto idle 120
        (echo -n ' theirsystem')       >/dev/console

echo '.'                               >/dev/console
```

To allow a PPP implementation running on 'theirsystem' to dial into 'oursystem', insert the following into **/etc/passwd** on 'oursystem':

```
Pthem:?:105:20:Their PPP:/etc/ppp:/etc/ppp/Login
```

where group 20 is the gid of the ppp group which owns /usr/etc/pppd, and **/etc/ppp/Login** is an executable shell script that looks something like

```
#!/bin/sh
PATH=/usr/bin:/usr/etc:/bin
mesg n
stty -tostop
exec pppd `hostname`:
```

RECOMMENDATIONS

Use host names when running **/etc/ppp/Autostart** from **/sbin/rc2.d/S522ppp** only if they are known locally. If a PPP connection to a DNS server would be required to resolve a host name, use its literal IP address instead.

p

EXTERNAL INFLUENCES

Environment Variables

The environment variable **PPPHOME**, if present, specifies the directory in which **pppd** looks for its configuration files (**Filter** and **Auth** for all connections, along with **Systems**, **Devices**, and **Dialers** if the connection is 'outbound'). You can specify **PPPHOME** either in the **Autostart** script or in an incoming connection's **Login** script. If **PPPHOME** is not present, **pppd** will expect to find its configuration files in **/etc/ppp/***.

SECURITY CONCERNS

pppd should be mode 4750, owned by root, and executable only by the members of the group containing all the incoming PPP login 'users'.

AUTHOR

pppd was developed by the Progressive Systems.

SEE ALSO

tun(4), ppp.Auth(4), ppp.Devices(4), ppp.Dialers(4), ppp.Filter(4), ppp.Keys(4), ppp.Systems(4), RFC 1548, RFC 1549, RFC 1332, RFC 1333, RFC 1334, RFC 1172, RFC 1144, RFC 1055, ds.internic.net:/internet-drafts/draft-ietf-pppext-compression-04.txt.

STANDARDS CONFORMANCE

HP PPP implements the IETF Proposed Standard Point-to-Point Protocol and many of its options and extensions, in conformance with RFCs 1548, 1549, 1332, 1333, 1334, and 1144. It can be configured to be conformant with earlier specifications of the PPP protocol, as described in RFCs 1134, 1171, and 1172. It implements the nonstandard SLIP protocol as described in RFCs 1055 and 1144.

NAME

pr - print files

SYNOPSIS

pr [*options*] [*files*]

DESCRIPTION

The **pr** command prints the named files on the standard output. If *file* is **-**, or if no files are specified, the standard input is assumed. By default, the listing is separated into pages, each headed by the page number, a date and time, and the name of the file.

By default, columns are of equal width, separated by at least one space; lines that do not fit are truncated. If the **-s** option is used, lines are not truncated and columns are separated by the separation character.

If the standard output is associated with a terminal, error messages are withheld until **pr** has completed printing.

Options

The following *options* can be used singly or combined in any order:

+*k*	Begin printing with page *k* (default is 1).
-*k*	Produce *k*-column output (default is 1). This option should not be used with **-m**. The options **-e** and **-i** are assumed for multi-column output.
-c *k*	Produce *k*-column output, same as **-***k*.
-a	Print multi-column output across the page. This option is appropriate only with the **-***k* option.
-m	Merge and print all files simultaneously, one per column (overrides the **-***k* and **-a** options).
-d	Double space the output.
-e*ck*	Expand *input* tabs to character positions *k*+1, 2×*k*+1, 3×*k*+1, etc. If *k* is 0 or is omitted, default tab settings at every eighth position are assumed. Tab characters in the input are expanded into the appropriate number of spaces. If *c* (any nondigit character) is given, it is treated as the input tab character (default for *c* is the tab character).
-i*ck*	In *output*, replace white space wherever possible by inserting tabs to character positions *k*+1, 2×*k*+1, 3×*k*+1, etc. If *k* is 0 or is omitted, default tab settings at every eighth position are assumed. If *c* (any nondigit character) is given, it is treated as the output tab character (default for *c* is the tab character).
-n*ck*	Provide *k*-digit line numbering (default for *k* is 5). The number occupies the first *k*+1 character positions of each column of normal output or each line of **-m** output. If *c* (any nondigit character) is given, it is appended to the line number to separate it from whatever follows (default for *c* is a tab).
-w*k*	Set the width of a line to *k* character positions (default is 72 for equal-width, multi-column output; no limit otherwise). Width specifications are only effective for multi-columnar output.
-o*k*	Offset each line by *k* character positions (default is 0). The number of character positions per line is the sum of the width and offset.
-l*k*	Set the length of a page to *k* lines (default is 66). If *k* is less than what is needed for the page header and trailer, the **-t** option is in effect; that is, header and trailer lines are suppressed in order to make room for text.
-h	Use the next argument as the header to be printed instead of the file name.
-p	Pause before beginning each page if the output is directed to a terminal (**pr** rings the bell at the terminal and waits for a **Return**).
-F	Use form-feed character for new pages (default is to use a sequence of line-feeds). Pause before beginning the first page if the standard output is associated with a terminal.
-f	Same as **-F**. Provided for backwards compatibility.
-r	Print no diagnostic reports on failure to open files.

-t Print neither the five-line identifying header nor the five-line trailer normally supplied for each page. Quit printing after the last line of each file without spacing to the end of the page.

-s*c* Separate columns by the single character *c* instead of by the appropriate number of spaces (default for *c* is a tab).

EXTERNAL INFLUENCES

Environment Variables

LC_CTYPE determines the interpretation of text and the arguments associated with the **-e**, **-i**, **-n**, and **-s** options as single-byte and/or multi-byte characters.

LC_TIME determines the format and contents of date and time strings.

LC_MESSAGES determines the language in which messages are displayed.

If **LC_CTYPE**, **LC_TIME**, or **LC_MESSAGES** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of **C** (see *lang*(5)) is used instead of **LANG**.

If any internationalization variable contains an invalid setting, **pr** behaves as if all internationalization variables are set to **C**. See *environ*(5).

International Code Set Support

Single-byte and multi-byte character code sets are supported.

RETURN VALUE

The **pr** returns the following values upon completion:

0	Successful completion.
>0	One or more of the input *files* do not exist or cannot be opened.

EXAMPLES

Print **file1** and **file2** as a double spaced, three column listing headed by “file list”:

```
pr -3dh "file list" file1 file2
```

Write **file1** on **file2**, expanding tabs to columns 10, 19, 28, 37, . . . :

```
pr -e9 -t <file1 >file2
```

Print **file1** in default format with nonblank lines numbered down the left side:

```
nl file1 | pr
```

FILES

/dev/tty

SEE ALSO

cat(1), lp(1), nl(1), ul(1).

STANDARDS CONFORMANCE

pr: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

NAME

praliases - print system-wide sendmail aliases

SYNOPSIS

praliases [**-f** *file*] [*key* ...]

DESCRIPTION

praliases prints out the contents of the alias data base used by **sendmail** to resolve system-wide mail aliases. The alias data base is built with the command **newaliases** or **/usr/sbin/sendmail -bi**. See *sendmail*(1M).

If the **-f** option is specified, **praliases** accesses the alias database built from *file* with the command

newaliases -oA *file*

Otherwise, **praliases** accesses the database built from the default alias file, **/etc/mail/aliases**.

Note that **praliases** accesses the database, not the alias file itself. If the alias file has changed since the alias database was last built, naturally the output of **praliases** cannot match the contents of the alias file.

Each *key* argument, if any, is looked up in the alias database. **praliases** prints out the aliases to which each key expands in the form:

*key***:** *mailing list*

where mailing list can be a comma-separated list of addresses to which the key resolves.

DIAGNOSTICS

*key***: No such key**

key was not found in the alias database.

EXAMPLES

```
$ praliases root postmaster no_user
root: jan_user
postmaster: joe_user
no_user: No such key
```

The output reveals that **root** is aliased to **jan_user**, **postmaster** is aliased to **joe_user**, and that there is no alias for the key **no_user**.

p

WARNINGS

Because **sendmail** supports NIS aliases, some NIS key-words may appear in the praliases output. These key-words, which include YP_LAST_MODIFIED and YP_MASTER_NAME, may be safely ignored; they merely indicate that **sendmail** is properly updating the alias database.

AUTHOR

praliases was developed by the University of California, Berkeley.

FILES

/etc/mail/aliases	default alias file
/etc/mail/aliases.db	default alias database

SEE ALSO

sendmail(1M).

NAME

prealloc - preallocate disk storage

SYNOPSIS

prealloc *name size*

DESCRIPTION

prealloc preallocates at least *size* bytes of disk space for an ordinary file *name*, creating the file if *name* does not already exist. The space is allocated in an implementation-dependent fashion for fast sequential reads and writes of the file.

prealloc fails and no disk space is allocated if *name* already exists and is not an ordinary file of zero length, if insufficient space is left on disk, or if *size* exceeds the maximum file size or the file size limit of the process (see *ulimit*(2)). The file is zero-filled.

DIAGNOSTICS

prealloc returns one of the following values upon completion:

0	Successful completion.
1	*name* already exists and is not an ordinary file of zero length.
2	There is insufficient room on the disk.
3	*size* exceeds file size limits.

EXAMPLES

The following example preallocates 50 000 bytes for the file **myfile**:

```
prealloc myfile 50000
```

WARNINGS

Allocation of file space is highly dependent on current disk usage. A successful return does not indicate how fragmented the file actually might be if the disk is approaching its capacity.

AUTHOR

prealloc was developed by HP.

SEE ALSO

prealloc(2), ulimit(2).

NAME

printenv - print out the environment

SYNOPSIS

printenv [*name*]

DESCRIPTION

printenv prints out the values of the variables in the environment. If a *name* is specified, only its value is printed.

RETURN VALUE

If a *name* is specified and it is not defined in the environment, **printenv** returns 1; otherwise it returns zero.

SEE ALSO

sh(1), environ(5), csh(1).

NAME

printf - format and print arguments

SYNOPSIS

printf *format* [*arg* ...]

DESCRIPTION

printf writes formatted arguments to the standard output. The *arg* arguments are formatted under control of the *format* operand.

format is a character string patterned after the formatting conventions of *printf*(3C), and contains the following types of objects:

characters — Characters that are not *escape sequences* or *conversion specifications* (as described below) are copied to standard output.

escape sequences — These are interpreted as non-graphic characters:

\a	alert
\b	backspace
\c	print line without appending a new-line
\f	form-feed
\n	new-line
\r	carriage return
\t	tab
\v	vertical tab
\'	single quote character
\\	backslash
\*n*	the 8-bit character whose ASCII code is the 1-, 2-, 3-, or 4-digit octal number *n*, whose first character must be a zero.

conversion specification
Specifies the output format of each argument (see below).

Arguments following *format* are interpreted as strings if the corresponding format is either **c** or **s**; otherwise they are treated as constants.

p

Conversion Specifications

Each conversion specification is introduced by the percent character **%**. After the **%** character, the following can appear in the sequence indicated:

flags — Zero or more *flags*, in any order, which modify the meaning of the conversion specification. The flag characters and their meanings are:

-	The result of the conversion is left-justified within the field.
+	The result of a signed conversion always begins with a sign, **+** or **-**.
<space>	If the first character of a signed conversion is not a sign, a space character is prefixed to the result. This means that if the space flag and **+** flag both appear, the space flag is ignored.
#	The value is to be converted to an "alternate form". For **c**, **d**, **i**, **u**, and **s** conversions, this flag has no effect. For **o** conversion, it increases the precision to force the first digit of the result to be a zero. For **x** or **X** conversion, a non-zero result has **0x** or **0X** prefixed to it. For **e**, **E**, **f**, **g**, and **G** conversions, the result always contains a radix character, even if no digits follow the radix character. For **g** and **G** conversions, trailing zeros are not removed from the result, contrary to usual behavior.

field width — An optional string of decimal digits to specify a minimum *field width*. For an output field, if the converted value has fewer characters than the field width, it is padded on the left (or right, if the left-adjustment flag, **-** has been given) to the field width.

precision — The *precision* specifies the minimum number of digits to appear for the **d**, **o**, **i**, **u**, **x**, or **X** conversions (the field is padded with leading zeros), the number of digits to appear after the radix character for the **e** and **f** conversions, the maximum number of significant digits for the **g** conversion, or the maximum number of characters to be printed from a string in s conversion. The precision takes the form of a period **.** followed by a decimal digit string. A null digit string is treated as a zero.

conversion characters
A *conversion character* indicates the type of conversion to be applied:

d,**i**, **o**,**u**, **x**,**X** The integer argument is printed a signed decimal (**d** or **i**), unsigned octal (**o**), unsigned decimal (**u**), or unsigned hexadecimal notation (**x** and **X**). The **x** conversion uses the numbers and letters **0123456789abcdef**, and the **X** conversion uses the numbers and letters **0123456789ABCDEF**. The *precision* component of the argument specifies the minimum number of digits to appear. If the value being converted can be represented in fewer digits than the specified minimum, it is expanded with leading zeroes. The default precision is 1. The result of converting a zero value with a precision of 0 is no characters.

f The floating-point number argument is printed in decimal notation in the style [**-**]*ddd***r***ddd*, where the number of digits after the radix character, **r**, is equal to the *precision* specification. If the *precision* is omitted from the argument, six digits are output; if the *precision* is explicitly 0, no radix appears.

e,**E** The floating-point-number argument is printed in the style [**-**]*d***r***ddd***e**±*dd*, where there is one digit before the radix character, and the number of digits after it is equal to the precision. When the precision is missing, six digits are produced; if the precision is 0, no radix character appears. The **E** conversion character produces a number with **E** introducing the exponent instead of **e**. The exponent always contains at least two digits. However, if the value to be printed requires an exponent greater than two digits, additional exponent digits are printed as necessary.

g,**G** The floating-point-number argument is printed in style **f** or **e** (or int style **E** in the case of a **G** conversion character), with the precision specifying the number of significant digits. The style used depends on the value converted; style **e** is used only if the exponent resulting from the conversion is less than **-h** or greater than or equal to the precision. Trailing zeros are remove from the result. A radix character appears only if it is followed by a digit.

c The first character of the argument is printed.

s The argument is taken to be a string, and characters from the string are printed until the end of the string or the number of characters indicated by the *precision* specification of the argument is reached. If the precision is omitted from the argument, it is interpreted as infinite and all characters up to the end of the string are printed.

% Print a **%** character; no argument is converted.

b Similar to the **s** conversion specifier, except that the string can contain backslash-escape sequences which are then converted to the characters they represent.

In no case does a nonexistent or insufficient field width cause truncation of a field; if the result of a conversion is wider than the field width, the field is simply expanded to contain the conversion result.

EXTERNAL INFLUENCES

Environment Variables

LC_CTYPE determines the interpretation of *arg* as single and/or multi-byte characters.

LC_MESSAGES determines the language in which messages are displayed.

If **LC_CTYPE** or **LC_MESSAGES** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**.

If any internationalization variable contains an invalid setting, *printf* behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported.

RETURN VALUE

printf exits with one of the following values:

- **0** Successful completion;
- **>0** Errors occurred. The exit value is increased by one for each error that occurred up to a maximum of 255.

DIAGNOSTICS

If an argument cannot be converted into a form suitable for the corresponding conversion specification, or for any other reason cannot be correctly printed, a diagnostic message is printed to standard error, the argument is output as a string form as it was given on the command line, and the exit value is incremented.

EXAMPLES

The following command prints the number 123 in octal, hexadecimal and floating point formats in their alternate form

```
printf "%#o, %#x, %#X, %#f, %#g, %#e\n" 123 123 123 123 123 123
```

resulting in the following output

```
0173, 0x7b, 0X7B, 123.000000, 123.000, 1.230000e+02
```

Print the outputs with their corresponding field widths and precision:

```
printf "%.6d, %10.6d, %.6f, %.6e, %.6s\n" 123 123 1.23 123.4 MoreThanSix
```

resulting in the following output

```
000123,     000123, 1.230000, 1.234000e+02, MoreTh
```

SEE ALSO

echo(1), printf(3S).

STANDARDS CONFORMANCE

printf: XPG4, POSIX.2

NAME

prmail - print out mail in the incoming mailbox file

SYNOPSIS

prmail [*user* ...]

DESCRIPTION

prmail prints the mail which waits for you or the specified user in the incoming mailbox file. The mailbox file is not disturbed.

prmail is functionally similar to the command:

```
cat /var/mail/mailfile | more
```

or

```
cat /var/mail/mailfile | pg
```

depending upon the setting of the user's **PAGER** environment variable

FILES

/var/mail/* incoming mailbox files

AUTHOR

prmail was developed by the University of California, Berkeley.

SEE ALSO

from(1), mail(1).

NAME

prof - display profile data

SYNOPSIS

prof [**-tcan**] [**-ox**] [**-g**] [**-z**] [**-h**] [**-s**] [**-m** *mdata*] [*prog*]

DESCRIPTION

prof interprets a profile file produced by **monitor()** (see *monitor*(3C)). The symbol table in the object file *prog* (**a.out** by default) is read and correlated with a profile file (**mon.out** by default). For each external text symbol, the percentage of time spent executing between the address of that symbol and the address of the next is printed, together with the number of times that function was called and the average number of milliseconds per call.

The mutually exclusive options **t**, **c**, **a**, and **n** determine the type of sorting of the output lines:

-t	Sort by decreasing percentage of total time (default).
-c	Sort by decreasing number of calls.
-a	Sort by increasing symbol address.
-n	Sort by symbol name in ascending collation order (see Environment Variables below).

The mutually exclusive options **o** and **x** specify the printing of the address of each symbol monitored:

-o	Print each symbol address (in octal) along with the symbol name.
-x	Print each symbol address (in hexadecimal) along with the symbol name.

The following options can be used in any combination:

-g	Include non-global symbols (static functions).
-z	Include all symbols in the profile range (see *monitor*(3C)), even if associated with zero number of calls and zero time.
-h	Suppress the heading normally printed on the report. (This is useful if the report is to be processed further.)
-s	Print a summary of several of the monitoring parameters and statistics on the standard error output.
-m *mdata*	Use file *mdata* instead of **mon.out** as the input profile file.

p

A program creates a profile file if it has been loaded using the **cc -p** option (see *cc*(1)). This option to the **cc** command arranges for calls to **monitor()** at the beginning and end of execution (see *monitor*(3C)). It is the call to the **monitor** command at the end of execution that causes a profile file to be written. The number of calls to a function is tallied if the **-p** option was used when the file containing the function was compiled.

The name of the file created by a profiled program is controlled by the environment variable **PROFDIR**. If **PROFDIR** is not set, **mon.out** is produced in the directory current when the program terminates. If **PROFDIR=string**, **string/pid.progname** is produced, where *progname* consists of argv[0] with any path prefix removed, and *pid* is the program's process ID. If **PROFDIR** is set to a null string, no profiling output is produced.

EXTERNAL INFLUENCES

Environment Variables

LC_COLLATE determines the collating order output by the **-n** option.

If **LC_COLLATE** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**. If any internationalization variable contains an invalid setting, **prof** behaves as if all internationalization variables are set to "C" (see *environ*(5)).

WARNINGS

The times reported in successive identical runs may show variances of 20% or more, because of varying cache-hit ratios due to sharing of the cache with other processes. Even if a program seems to be the only one using the machine, hidden background or asynchronous processes may blur the data. In rare cases, the clock ticks initiating recording of the program counter may "beat" with loops in a program, grossly distorting measurements.

Call counts are always recorded precisely, however.

Only programs that call **`exit()`** (see *exit*(2)) or return from **`main`** cause a profile file to be produced, unless a final call to **`monitor()`** is explicitly coded.

The use of the **`cc -p`** option to invoke profiling imposes a limit of 600 functions that can have call counters established during program execution. For more counters, call **`monitor()`** directly. If this limit is exceeded, other data is overwritten and the **`mon.out`** file is corrupted. The number of call counters used is reported automatically by the **`prof`** command whenever the number exceeds 5/6 of the maximum.

FILES

`mon.out`	for profile
`a.out`	for namelist

SEE ALSO

cc(1), exit(2), profil(2), crt0(3), end(3C), monitor(3C).

STANDARDS CONFORMANCE

`prof`: SVID2, SVID3, XPG2

NAME

prs - print and summarize an SCCS file

SYNOPSIS

prs [**-d** *dataspec*] [**-r**[*SID*]] [**-e**] [**-l**] [**-c** *cutoff*] [**-a**] *file* ...

DESCRIPTION

The **prs** command prints, on the standard output, parts or all of an SCCS file (see *sccsfile*(4)) in a user-supplied format. If a directory is named, **prs** behaves as though each file in the directory were specified as a named file, except that non-SCCS files (last component of the path name does not begin with **s.**), and unreadable files are silently ignored. If a name of **-** is given, the standard input is read; each line of the standard input is taken to be the name of an SCCS file or directory to be processed; non-SCCS files and unreadable files are silently ignored. A **--** on the command line implies that all following arguments are file/directory names.

Arguments to **prs**, which can appear in any order, consist of options and file names.

Options

All of the described options apply independently to each named file:

-d *dataspec* Used to specify the output data specification. *dataspec* is a string consisting of SCCS file *data keywords* (see Data Keywords below) interspersed with optional user-supplied text.

-r[*SID*] Used to specify the *S*CCS *ID*entification (*SID*) string of a delta for which information is desired. If no *SID* is specified, the *SID* of the most recently created delta is assumed. If an *SID* is specified, it must agree exactly with an *SID* in the file (that is, the *SID* structure used by **get** does not work here — see *get*(1)).

-e Requests information for all deltas created *earlier* than and including the delta designated via the **-r** option or the date given by the **-c** option.

-l Requests information for all deltas created *later* than and including the delta designated via the **-r** option or the date given by the **-c** option.

-c *cutoff* Cutoff date-time, in the form

YY[*MM*[*DD*[*HH*[*MM*[*SS*]]]]]

Units omitted from the date-time default to their maximum possible values. Thus, **-c7502** is equivalent to **-c750228235959**. One or more non-numeric characters can be used to separate the various 2-digit segments of the cutoff date (for example **-c77/2/2 9:22:25**).

For 2-digit year input (*YY*), the following interpretation is taken: [**70**-**99**, **00**-**69** (1970-1999, 2000-2069)].

-a Requests printing of information for both removed, i.e., delta type = *R*, (see *rmdel*(1)) and existing, that is, delta type = *D*, deltas. If the **-a** option is not specified, information is provided for existing deltas only.

If no option letters (or only **-a**) are given, **prs** prints the file name using the default *dataspec* and the **-e** option. This produces information on all deltas.

Data Keywords

Data keywords specify which parts of an SCCS file are to be retrieved and output. All parts of an SCCS file (see *sccsfile*(4)) have an associated data keyword. There is no limit on the number of times a data keyword can appear in a *dataspec*.

The information printed by **prs** consists of: (1) the user-supplied text; and (2) appropriate values (extracted from the SCCS file) substituted for the recognized data keywords in the order of appearance in the *dataspec*. The format of a data keyword value is either **Simple** (S), in which keyword substitution is direct, or **Multi-line** (M), in which keyword substitution is followed by a carriage return.

User-supplied text is any text other than recognized data keywords. Escapes can be used as follows:

backslash	\	form feed	**\f**
backspace	**\b**	new-line	**\n**

p

carriage-return	**\r**	single quote	**\´**
colon	**\:**	tab	**\t**

The default *dataspec* is:

```
":Dt:\t:DL:\nMRs:\n:MR:COMMENTS:\n:C:"
```

SCCS File Data Keywords

Keyword	Data Item	File Section	Value	Fmt
:Dt:	Delta information	Delta Table	See below*	S
:DL:	Delta line statistics	"	**:Li:/:Ld:/:Lu:**	S
:Li:	Lines inserted by Delta	"	*nnnnn*	S
:Ld:	Lines deleted by Delta	"	*nnnnn*	S
:Lu:	Lines unchanged by Delta	"	*nnnnn*	S
:DT:	Delta type	"	*D* or *R*	S
:I:	SCCS ID string (SID)	"	**:R:.:L:.:B:.:S:**	S
:R:	Release number	"	*nnnn*	S
:L:	Level number	"	*nnnn*	S
:B:	Branch number	"	*nnnn*	S
:S:	Sequence number	"	*nnnn*	S
:D:	Date Delta created	"	**:Dy:/:Dm:/:Dd:**	S
:Dy:	Year Delta created	"	*nn*	S
:Dm:	Month Delta created	"	*nn*	S
:Dd:	Day Delta created	"	*nn*	S
:T:	Time Delta created	"	**:Th:::Tm:::Ts:**	S
:Th:	Hour Delta created	"	*nn*	S
:Tm:	Minutes Delta created	"	*nn*	S
:Ts:	Seconds Delta created	"	*nn*	S
:P:	Programmer who created Delta	"	*logname*	S
:DS:	Delta sequence number	"	*nnnn*	S
:DP:	Predecessor Delta seq-no.	"	*nnnn*	S
:DI:	Seq # of deltas incl, excl, ign	"	**:Dn:/:Dx:/:Dg:**	S
:Dn:	Deltas included (seq #)	"	**:DS: :DS: . . .**	S
:Dx:	Deltas excluded (seq #)	"	**:DS: :DS: . . .**	S
:Dg:	Deltas ignored (seq #)	"	**:DS: :DS: . . .**	S
:MR:	MR numbers for delta	"	*text*	M
:C:	Comments for delta	"	*text*	M
:UN:	User names	User name	*text*	M
:FL:	Flag list	Flags	*text*	M
:Y:	Module type flag	"	*text*	S
:MF:	MR validation flag	"	**yes** or **no**	S
:MP:	MR validation pgm name	"	*text*	S
:KF:	Keyword error/warning flag	"	**yes** or **no**	S
:KV:	Keyword validation string	"	*text*	S
:BF:	Branch flag	"	**yes** or **no**	S
:J:	Joint edit flag	"	**yes** or **no**	S
:LK:	Locked releases	"	**:R: . . .**	S
:Q:	User defined keyword	"	*text*	S
:M:	Module name	"	*text*	S
:FB:	Floor boundary	"	**:R:**	S
:CB:	Ceiling boundary	"	**:R:**	S
:Ds:	Default SID	"	**:I:**	S
:ND:	Null delta flag	"	**yes** or **no**	S
:FD:	File descriptive text	Comments	*text*	M
:BD:	Body	Body	*text*	M
:GB:	Gotten body	"	*text*	M
:W:	A form of *what*(1) string	N/A	**:Z::M:\t:I:**	S
:A:	A form of *what*(1) string	N/A	**:Z::Y: :M: :I::Z:**	S
:Z:	*what*(1) string delimiter	N/A	**@(#)**	S
:F:	SCCS file name	N/A	*text*	S
:PN:	SCCS file path name	N/A	*text*	S

p

```
* :Dt: = :DT: :I: :D: :T: :P: :DS: :DP:
```

If no option letters (or only **-a**) are given, **prs** prints the file name, using the default *dataspec*, and the **-e** option; thus, information on all deltas is produced.

EXTERNAL INFLUENCES

Environment Variables

LC_CTYPE determines the interpretation of dataspec as single- and/or multi-byte characters.

LC_MESSAGES determines the language in which messages are displayed.

If **LC_CTYPE** or **LC_MESSAGES** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**. If any internationalization variable contains an invalid setting, **prs** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single-byte and multi-byte character code sets are supported.

DIAGNOSTICS

Use *sccshelp*(1) for explanations.

EXAMPLES

The entry

```
prs -d"Users and/or user IDs for :F: are :\n:UN:" s.file
```

may produce on the standard output:

```
Users and/or user IDs for s.file are:
xyz
131
abc
```

The entry

```
prs -d"Newest delta for pgm :M:: :I: Created :D: By :P:" -r s.file
```

may produce on the standard output:

```
Newest delta for pgm main.c: 3.7 Created 77/12/1 By cas
```

As a *special case* (when no specifications for selecting or printing are given)

```
prs s.file
```

may produce on the standard output:

```
D 1.1 77/12/1 00:00:00 cas 1 000000/00000/00000
MRs:
bl78-12345
bl79-54321
COMMENTS:
this is the comment line for s.file initial delta
```

for each delta table entry of the “D” type. The only option argument allowed to be used with the *special case* is the **-a** option.

FILES

/tmp/pr?????

SEE ALSO

admin(1), delta(1), get(1), sccshelp(1), sccsfile(4).

STANDARDS CONFORMANCE

prs: SVID2, SVID3, XPG2, XPG3, XPG4

NAME

ps - report process status

SYNOPSIS

ps [**-adeflP**] [**-g** *grplist*] [**-p** *proclist*] [**-R** *prmgrplist*] [**-t** *termlist*] [**-u** *uidlist*]

XPG4 Synopsis

ps [**-aAcdefHjlP**] [**-C** *cmdlist*] [**-g** *grplist*] [**-G** *gidlist*] [**-n** *namelist*] [**-o** *format*] [**-p** *proclist*] [**-R** *prmgrplist*] [**-s** *sidlist*] [**-t** *termlist*] [**-u** *uidlist*] [**-U** *uidlist*]

DESCRIPTION

ps prints information about selected processes. Use options to specify which processes to select and what information to print about them.

Process Selection Options

Use the following options to choose which processes should be selected.

NOTE: If an option is used in both the default (standard HP-UX) and XPG4 environments, the description provided here documents the default behavior. Refer to the **UNIX95** variable under EXTERNAL INFLUENCES for additional information on XPG4 behavior.

(none)	Select those processes associated with the current terminal.
-A	(XPG4 Only.) Select all processes. (Synonym for **-e**.)
-a	Select all processes except process group leaders and processes not associated with a terminal.
-C *cmdlist*	(XPG4 Only.) Select processes executing a command with a basename given in *cmdlist*.
-d	Select all processes except process group leaders.
-e	Select all processes.
-g *grplist*	Select processes whose process group leaders are given in *grplist*.
-G *gidlist*	(XPG4 Only.) Select processes whose real group ID numbers or group names are given in *gidlist*.
-n *namelist*	(XPG4 Only.) This option is ignored; its presence is allowed for standards compliance.
-p *proclist*	Select processes whose process ID numbers are given in *proclist*.
-R *prmgrplist*	Select processes belonging to PRM process resource groups whose names or ID numbers are given in *prmgrplist*. See DEPENDENCIES.
-s *sidlist*	(XPG4 Only.) Select processes whose session leaders are given in *sidlist*. (Synonym for **-g**).
-t *termlist*	Select processes associated with the terminals given in *termlist*. Terminal identifiers can be specified in one of two forms: the device's file name (such as **tty04**) or if the device's file name starts with **tty**, just the rest of it (such as **04**). If the device's file is in a directory other than **/dev** or **/dev/pty**, the terminal identifier must include the name of the directory under **/dev** that contains the device file (such as **pts/5**).
-u *uidlist*	Select processes whose effective user ID numbers or login names are given in *uidlist*.
-U *uidlist*	(XPG4 Only.) Select processes whose real user ID numbers or login names are given in *uidlist*.

If any of the **-a**, **-A**, **-d**, or **-e** options is specified, the **-C**, **-g**, **-G**, **-p**, **-R**, **-t**, **-u**, and **-U** options are ignored.

If more than one of **-a**, **-A**, **-d**, and **-e** are specified, the least restrictive option takes effect.

If more than one of the **-C**, **-g**, **-G**, **-p**, **-R**, **-t**, **-u**, and **-U** options are specified, processes will be selected if they match any of the options specified.

The lists used as arguments to the **-C**, **-g**, **-G**, **-p**, **-R**, **-t**, **-u**, and **-U** options can be specified in one of two forms:

- A list of identifiers separated from one another by a comma.
- A list of identifiers enclosed in quotation marks (**"**) and separated from one another by a comma and/or one or more spaces.

Output Format Options

Use the following options to control which columns of data are included in the output listing. The options are cumulative.

(none)	The default columns are: **pid**, **tty**, **time**, and **comm**, in that order.
-f	Show columns **user**, **pid**, **ppid**, **cpu**, **stime**, **tty**, **time**, and **args**, in that order.
-l	Show columns **flags**, **state**, **uid**, **pid**, **ppid**, **cpu**, **intpri**, **nice**, **addr**, **sz**, **wchan**, **tty**, **time**, and **comm**, in that order.
-fl	Show columns **flags**, **state**, **user**, **pid**, **ppid**, **cpu**, **intpri**, **nice**, **addr**, **sz**, **wchan**, **stime**, **tty**, **time**, and **args**, in that order.
-c	(XPG4 Only.) Remove columns **cpu** and **nice**; replace column **intpri** with columns **cls** and **pri**.
-j	(XPG4 Only.) Add columns **pgid** and **sid** after column **ppid** (or **pid**, if **ppid** is not being displayed).
-P	Add column **prmid** (for **-l**) or **prmgrp** (for **-f** or **-fl**) immediately before column **pid**. See DEPENDENCIES.
-o *format*	(XPG4 Only.) *format* is a comma- or space-separated list of the columns to display, in the order they should be displayed. (Valid column names are listed below.) A column name can optionally be followed by an equals sign (**=**) and a string to use as the heading for that column. (Any commas or spaces after the equals sign will be taken as a part of the column heading; if more columns are desired, they must be specified with additional **-o** options.) The width of the column will be the greater of the width of the data to be displayed and the width of the column heading. If an empty column heading is specified for every heading, no heading line will be printed. This option overrides options **-c**, **-f**, **-j**, **-l**, and **-P**; if they are specified, they are ignored.
-H	(XPG4 Only.) Shows the process hierarchy. Each process is displayed under its parent, and the contents of the **args** or **comm** column for that process is indented from that of its parent. Note that this option is expensive in both memory and speed.

p

The column names and their meanings are given below. Except where noted, the default heading for each column is the uppercase form of the column name.

addr	The memory address of the process, if resident; otherwise, the disk address.
args	The command line given when the process was created. This column should be the last one specified, if it is desired. Only a subset of the command line is saved by the kernel; as much of the command line will be displayed as is available. The output in this column may contain spaces. The default heading for this column is **COMMAND** if **-o** is specified and **CMD** otherwise.
cls	Process scheduling class, see *rtsched*(1).
comm	The command name. The output in this column may contain spaces. The default heading for this column is **COMMAND** if **-o** is specified and **CMD** otherwise.
cpu	Processor utilization for scheduling. The default heading for this column is **C**.
etime	Elapsed time of the process. The default heading for this column is **ELAPSED**.
flags	Flags (octal and additive) associated with the process:

0	Swapped
1	In core
2	System process
4	Locked in core (e.g., for physical I/O)
10	Being traced by another process
20	Another tracing flag

The default heading for this column is **F**.

intpri	The priority of the process as it is stored internally by the kernel. This column is provided for backward compatibility and its use is not encouraged.
gid	The group ID number of the effective process owner.
group	The group name of the effective process owner.
nice	Nice value; used in priority computation (see *nice*(1)). The default heading for this column is **NI**.
pcpu	The percentage of CPU time used by this process during the last scheduling interval. The default heading for this column is **%CPU**.
pgid	The process group ID number of the process group to which this process belongs.
pid	The process ID number of the process.
ppid	The process ID number of the parent process.
pri	The priority of the process. The meaning of the value depends on the process scheduling class; see **cls**, above, and *rtsched*(1).
prmid	The PRM process resource group ID number.
prmgrp	The PRM process resource group name.
rgid	The group ID number of the real process owner.
rgroup	The group name of the real process owner.
ruid	The user ID number of the real process owner.
ruser	The login name of the real process owner.
sid	The session ID number of the session to which this process belongs.
state	The state of the process:

0	Nonexistent
S	Sleeping
W	Waiting
R	Running
I	Intermediate
Z	Terminated
T	Stopped
X	Growing

The default heading for this column is **S**.

stime	Starting time of the process. If the elapsed time is greater than 24 hours, the starting date is displayed instead.
sz	The size in physical pages of the core image of the process, including text, data, and stack space. Physical page size is defined by **_SC_PAGE_SIZE** in the header file **<unistd.h>** (see *sysconf*(2) and *unistd*(5)).
time	The cumulative execution time for the process.
tty	The controlling terminal for the process. The default heading for this column is **TT** if **-o** is specified and **TTY** otherwise.
uid	The user ID number of the effective process owner.
user	The login name of the effective process owner.
vsz	The size in kilobytes (1024 byte units) of the core image of the process. See column **sz**, above.
wchan	The event for which the process is waiting or sleeping; if there is none, a hyphen (-) is displayed.

Notes

ps prints the command name and arguments given at the time of the process was created. If the process changes its arguments while running (by writing to its *argv* array), these changes are not displayed by **ps**.

A process that has exited and has a parent, but has not yet been waited for by the parent, is marked **<defunct>** (see **zombie process** in *exit*(2)).

The time printed in the **stime** column, and used in computing the value for the **etime** column, is the time when the process was forked, *not* the time when it was modified by **exec()**.

To make the **ps** output safer to display and easier to read, all control characters in the **comm** and **args** columns are displayed as "visible" equivalents in the customary control character format, **^***x*.

EXTERNAL INFLUENCES

Environment Variables

UNIX95 specifies to use the XPG4 behavior for this command. The changes for XPG4 include support for the entire option set specified above and include the following behavioral changes:

- The **TIME** column format changes from *mmmm*:*ss* to [*dd*-]*hh*:*mm*:*ss*.
- When the **comm**, **args**, **user**, and **prmgrp** fields are included by default or the **-f** or **-l** flags are used, the column headings of those fields change to **CMD**, **CMD**, **USER**, and **PRMGRP**, respectively.
- **-a**, **-d**, and **-g** will select processes based on session rather than on process group.
- The uid or user column displayed by **-f** or **-l** will display effective user rather than real user.
- The **-u** option will select users based on effective UID rather than real UID.
- The **-C** and **-H** options, while they are not part of the XPG4 standard, are enabled.

LC_TIME determines the format and contents of date and time strings. If it is not specified or is null, it defaults to the value of **LANG**.

If **LANG** is not specified or is null, it defaults to **C** (see *lang*(5)).

If any internationalization variable contains an invalid setting, all internationalization variables default to **C** (see *environ*(5)).

International Code Set Support

Single-byte character code sets are supported.

EXAMPLES

Generate a full listing of all processes currently running on your machine:

```
ps -ef
```

To see if a certain process exists on the machine, such as the **cron** clock daemon, check the far right column for the command name, **cron**, or try

```
ps -f -C cron
```

WARNINGS

Things can change while **ps** is running; the picture it gives is only a snapshot in time. Some data printed for defunct processes is irrelevant.

If two special files for terminals are located at the same select code, that terminal may be reported with either name. The user can select processes with that terminal using either name.

Users of **ps** must not rely on the exact field widths and spacing of its output, as these will vary depending on the system, the release of HP-UX, and the data to be displayed.

DEPENDENCIES

HP Process Resource Manager

The **-P** and **-R** options require the optional HP Process Resource Manager (PRM) software to be installed and configured. See *prmconfig*(1) for a description of how to configure HP PRM, and *prmconf*(4) for the definition of "process resource group."

If HP PRM is not installed and configured and **-P** or **-R** is specified, a warning message is displayed and (for **-P**) hyphens (-) are displayed in the **prmid** and **prmgrp** columns.

FILES

`/dev`	Directory of terminal device files
`/etc/passwd`	User ID information
`/var/adm/ps_data`	Internal data structure

SEE ALSO

kill(1), nice(1), acctcom(1M), exec(2), exit(2), fork(2), sysconf(2), unistd(5).

HP Process Resource Manager: prmconfig(1), prmconf(4) in *HP Process Resource Manager User's Guide*.

STANDARDS COMPLIANCE

`ps`: SVID2, XPG2, XPG3, XPG4

NAME

ptx - permuted index

SYNOPSIS

ptx [*options*] [*input* [*output*]]

DESCRIPTION

ptx generates the file *output* that can be processed with a text formatter to produce a permuted index of file *input* (standard input and output default). It has three phases: the first does the permutation, generating one line for each keyword in an input line. The keyword is rotated to the front. The permuted file is then sorted (see *sort*(1) and Environment Variables below). Finally, the sorted lines are rotated so the keyword comes at the middle of each line. **ptx** output is in the form:

.xx "*tail***" "***before keyword***" "***keyword and after***" "***head***"**

where **.xx** is assumed to be an **nroff** or **troff** macro provided by the user, or provided by the **mptx** macro package (see NOTES below). The *before keyword* and *keyword and after* fields incorporate as much of the line as will fit around the keyword when it is printed. *tail* and *head*, at least one of which is always the empty string, are wrapped-around pieces small enough to fit in the unused space at the opposite end of the line.

The following *options* can be applied:

-f	Fold uppercase and lowercase letters for sorting.
-t	Prepare the output for the phototypesetter by using a line length of 100.
-w *n*	Use the next argument, *n*, as the length of the output line. The default line length is 72 characters for **nroff** and 100 for **troff**.
-g *n*	Use the next argument, *n*, as the number of characters that **ptx** will reserve in its calculations for each gap among the four parts of the line as finally printed. The default gap is 3.
-o *only*	Use as keywords only the words given in the *only* file.
-i *ignore*	Do not use as keywords any words given in the *ignore* file. If the **-i** and **-o** options are missing, use **/usr/lib/eign** as the *ignore* file.
-b *break*	Use the characters in the *break* file to separate words. Tab, new-line, and space characters are *always* used as break characters. Punctuation characters are treated as part of the word in the absence of this option.
-r	Take any leading non-blank characters of each input line to be a reference identifier (as to a page or chapter), separate from the text of the line. Attach that identifier as a 5th field on each output line.

p

EXTERNAL INFLUENCES

Environment Variables

LC_COLLATE determines the order in which the output is sorted.

LC_CTYPE determines the default break characters.

If **LC_COLLATE** or **LC_CTYPE** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**. If any internationalization variable contains an invalid setting, **ptx** behaves as if all internationalization variables are set to "C" (see *environ*(5)).

International Code Set Support

Single-byte character code sets are supported.

WARNINGS

Line length counts do not account for overstriking or proportional spacing.

Lines containing tildes (~) are botched because **ptx** uses that character internally.

FILES

`/usr/lib/eign`
`/usr/bin/sort`
`/usr/share/lib/tmac/tmac.ptx`

NOTES

The **`mptx`** macro package is not provided as part of the HP-UX operating system. It is part of the Documenters Work Bench (DWB) software package originally developed by AT&T which has been ported to HP 9000 systems by various third-party software suppliers including Elan Computer Group, Inc. of Mountain View California and others.

Permuted indexes produced by using **`ptx`** usually have a 4-column format that some users prefer and others dislike greatly. The two-column format index provided in this manual is created by processing index entries that are hidden as comments at the end of each manual entry file.

SEE ALSO

nroff(1), mm(5).

NAME

pwd - working directory name

SYNOPSIS

pwd

DESCRIPTION

pwd prints the path name of the working (current) directory.

EXTERNAL INFLUENCES

Environment Variables

LC_MESSAGES determines the language in which messages are displayed.

If **LC_MESSAGES** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**.

If any internationalization variable contains an invalid setting, **pwd** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported.

DIAGNOSTICS

Cannot open ..
Read error in ..

Possible file system trouble; contact system administrator.

pwd: cannot access parent directories

Current directory has been removed (usually by a different process). Use **cd** command to move to a valid directory (see *cd*(1)).

EXAMPLES

This command lists the path of the current working directory. If your home directory were **/mnt/staff** and the command **cd camp/nevada** were executed from the home directory, typing **pwd** would produce the following:

```
/mnt/staff/camp/nevada
```

AUTHOR

pwd was developed by AT&T and HP.

SEE ALSO

cd(1).

STANDARDS CONFORMANCE

pwd: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

NAME

pwget, grget - get password and group information

SYNOPSIS

pwget [**-n** *name* | **-u** *uid*]

grget [**-n** *name* | **-g** *gid*]

DESCRIPTION

pwget and **grget** locate and display information from **/etc/passwd** and **/etc/group**.

The standard output of **pwget** contains lines of colon-separated password information whose format is the same as that used in the **/etc/passwd** file (see *passwd*(4)).

The standard output of **grget** contains lines of colon-separated group information whose format is the same as that used in the **/etc/group** file (see *group*(4)).

With no options, **pwget** and **grget** get all entries with **getpwent()** or **getgrent()** respectively, (see *getpwent*(3C) and *getgrent*(3C)), and output a line for each entry found.

Options

When an option is given, only a single entry is printed.

The options for **pwget** are:

-n *name*	Output the first entry that matches *name* using **getpwnam()** (see *getpwent*(3C)).
-u *uid*	Output the first entry that matches *uid* using **getpwuid()** (see *getpwent*(3C)).

The options for **grget** are:

-n *name*	Output the first entry that matches *name* using **getgrnam()** (see *getgrent*(3C)).
-g *gid*	Output the first entry that matches *gid* using **getgrgid()** (see *getgrent*(3C)).

NETWORKING FEATURES

NFS

If Network Information System (NIS) is in use, these commands provide password and group information based on the NIS version of the password and group databases in addition to the local password and group files.

p

RETURN VALUE

These commands return **0** upon success, **1** when a specific search fails, and **2** upon error.

DEPENDENCIES

NFS:

WARNING: If the Network Information System network database is in use and the NIS client daemon (**ypbind**) is not bound to a NIS server daemon (see *ypserv*(1M)), these utilities will wait until such a binding is established. These commands can be terminated in this condition by sending a **SIGINT** signal to the process (see *kill*(1)).

See *ypmatch*(1), and *ypserv*(1M).

AUTHOR

pwget and **grget** were developed by HP.

FILES

/etc/group	local group data file
/etc/passwd	local password data file

SEE ALSO

getgrent(3C), getpwent(3C), group(4), passwd(4).

NAME

quota - display disk usage and limits

SYNOPSIS

quota [**-v**] [*user* ...]

DESCRIPTION

The **quota** command displays the disk usage and limits for one or more *user*s. Without the **-v** option, it displays information only when the usage exceeds the limits.

user is a user name or a numeric UID. The default is the login user name.

Only users with appropriate privileges can view the limits of other users.

Options

The **quota** command recognizes the following option:

-v Display the statistics whether they exceed limits or not. Note that no usage statistics exist if no quota is set.

EXTERNAL INFLUENCES

Environment Variables

LC_MESSAGES determines the language in which messages are displayed.

If **LC_MESSAGES** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**.

If any internationalization variable contains an invalid setting, **quota** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported.

AUTHOR

Disk Quotas were developed by the University of California, Berkeley, Sun Microsystems, Inc., and HP.

FILES

directory/**quotas**	Quota statistics static storage for a file system, where *directory* is the root of the file system as specified to the **mount** command (see *mount*(1M)).
/etc/mnttab	List of currently mounted file systems

SEE ALSO

quota(5).

NAME

ranlib - regenerate archive symbol table

SYNOPSIS

ranlib *archive* ...

DESCRIPTION

ranlib regenerates the symbol tables of the specified archives. It is equivalent to executing **ar qs archive** on each of the archives. After using the **z** modifier of **ar**, the symbol table of an archive must be regenerated before it can be used.

EXTERNAL INFLUENCES

Environment Variables

The following internationalization variables affect the execution of **ranlib**:

LANG
: Determines the locale category for native language, local customs and coded character set in the absence of **LC_ALL** and other **LC_*** environment variables. If **LANG** is not specified or is set to the empty string, a default of **C** (see *lang*(5)) is used instead of **LANG**.

LC_ALL
: Determines the values for all locale categories and has precedence over **LANG** and other **LC_*** environment variables.

LC_CTYPE
: Determines the locale category for character handling functions.

LC_MESSAGES
: Determines the locale that should be used to affect the format and contents of diagnostic messages written to standard error.

LC_NUMERIC
: Determines the locale category for numeric formatting.

LC_TIME
: Determines the format and contents of date and time formatting.

NLSPATH
: Determines the location of message catalogues for the processing of **LC_MESSAGES**.

If any internationalization variable contains an invalid setting, **ranlib** behaves as if all internationalization variables are set to **C**. See *environ*(5).

In addition, the following environment variable affects **ranlib**:

TMPDIR
: Specifies a directory for temporary files (see *tmpnam*(3S)).

SEE ALSO

System Tools:

ar(1)	create archived libraries

Miscellaneous:

ar(4)	archive format
strip(1)	strip symbol and line number information from an object file

NAME

rcp - remote file copy

SYNOPSIS

Copy Single File

rcp [**-p**] [**-S** *size*] [**-R** *size*] *source_file1 dest_file*

Copy Multiple Files

rcp [**-p**] [**-S** *size*] [**-R** *size*] *source_file1* [*source_file2*]... *dest_dir*

Copy One or More Directory Subtrees

rcp [**-p**] [**-S** *size*] [**-R** *size*] **-r** *source_dir1* [*source_dir2*]... *dest_dir*

Copy Files and Directory Subtrees

rcp [**-p**] [**-S** *size*] [**-R** *size*] **-r** *file_or_dir1* [*file_or_dir2*]... *dest_dir*

DESCRIPTION

The **rcp** command copies files, directory subtrees, or a combination of files and directory subtrees from one or more systems to another. In many respects, it is similar to the **cp** command (see *cp*(1)).

To use **rcp**, you must have read access to files being copied, and read and search (execute) permission on all directories in the directory path.

Options and Arguments

rcp recognizes the following options and arguments:

source_file
source_dir The name of an existing file or directory on a local or remote machine that you want copied to the specified destination. Source file and directory names are constructed as follows:

user_name@*hostname*:*pathname*/*filename*

or

user_name@*hostname*:*pathname*/*dirname*

Component parts of file and directory names are described below. If multiple existing files and/or directory subtrees are specified (*source_file1*, *source_file2*, ..., etc.), the destination must be a directory. Shell file name expansion is allowed on both local and remote systems. Multiple files and directory subtrees can be copied from one or more systems to a single destination directory with a single command.

dest_file The name of the destination file. If host name and path name are not specified, the existing file is copied into a file named *dest_file* in the current directory on the local system. If *dest_file* already exists and is writable, the existing file is overwritten. Destination file names are constructed the same way as source files except that file name expansion characters cannot be used.

dest_dir The name of the destination directory. If host name and path name are not specified, the existing file is copied into a directory named *dest_dir* in the current directory on the local system. If *dest_dir* already exists in the specified directory path (or current directory if not specified), a new directory named *dest_dir* is created underneath the existing directory named *dest_dir*. Destination directory names are constructed the same way as source directory tree names except that file name expansion characters cannot be used.

file_or_dir If a combination of files and directories are specified for copying (either explicitly or by file name expansion), only files are copied unless the **-r** option is specified. If the **-r** option is present, all files and directory subtrees whose names match the specified *file_or_dir* name are copied.

-p Preserve (duplicate) modification times and modes (permissions) of source files, ignoring the current setting of the **umask** file creation mode mask. If this option is specified, **rcp** preserves the sticky bit only if the target user is superuser.

If the **-p** option is not specified, **rcp** preserves the mode and owner of *dest_file* if it already exists; otherwise **rcp** uses the mode of the source file modified by the **umask** on the destination host. Modification and access times of the destination file are set to the time when the copy was made.

-S *size* This option sets the size of the socket send buffer.

-R *size* This option sets the size of the socket receive buffer.

-r Recursively copy directory subtrees rooted at the source directory name. If any directory subtrees are to be copied, **rcp** recursively copies each subtree rooted at the specified source directory name to directory *dest_dir*. If *source_dir* is being copied to an existing directory of the same name, **rcp** creates a new directory *source_dir* within *dest_dir* and copies the subtree rooted at *source_dir* to *dest_dir***/***source_dir*. If *dest_dir* does not exist, **rcp** creates it and copies the subtree rooted at *source_dir* to *dest_dir*.

Constructing File and Directory Names

As indicated above, file and directory names contain one, two, or four component parts:

user_name	Login name to be used for accessing directories and files on remote system.
hostname	Hostname of remote system where directories and files are located.
pathname	Absolute directory path name or directory path name relative to the login directory of user *user_name*.
filename	Actual name of source or destination file. File name expansion is allowed on source file names.
dirname	Actual name of source or destination directory subtree. File name expansion is allowed on source directory names.

Each *file* or *directory* argument is either a remote file name of the form *hostname***:***path*, or a local file name (with a slash (**/**) before any colon (**:**)). *hostname* can be either an official host name or an alias (see *hosts*(4)). If *hostname* is of the form *ruser*@*rhost*, *ruser* is used on the remote host instead of the current user name. An unspecified *path* (that is, *hostname***:**) refers to the remote user's login directory. If *path* does not begin with **/**, it is interpreted relative to the remote user's login directory on *hostname*. Shell metacharacters in remote *paths* can be quoted with backslash (****), single quotes (**''**), or double quotes (**""**), so that they will be interpreted remotely.

The **rcp** routine does not prompt for passwords. The current local user name or any user name specified via *ruser* must exist on *rhost* and allow remote command execution via *remsh*(1) and *rcmd*(3N). *remshd*(1M) must be executable on the remote host.

Third-party transfers in the form:

```
rcp ruser1@rhost1:path1 ruser2@rhost2:path2
```

are performed as:

```
remsh rhost1 -l ruser1 rcp path1 ruser2@rhost2:path2
```

Therefore, for a such a transfer to succeed, *ruser2* on *rhost2* must allow access by *ruser1* from *rhost1* (see *hosts.equiv*(4)).

WARNINGS

The **rcp** routine is confused by any output generated by commands in a **.cshrc** file on the remote host (see *csh*(1)).

Copying a file onto itself, for example:

```
rcp path ‘hostname‘:path
```

may produce inconsistent results. The current HP-UX version of **rcp** simply copies the file over itself. However, some implementations of **rcp**, including some earlier HP-UX implementations, corrupt the file. In addition, the same file may be referred to in multiple ways, for example, via hard links, symbolic links, or NFS. It is not guaranteed that **rcp** will correctly copy a file over itself in all cases.

Implementations of **rcp** based on the 4.2BSD version (including the implementations of **rcp** prior to HP-UX 7.0) require that remote users be specified as *rhost*.*ruser*. If the first remote host specified in a third party transfer (*rhost1* in the example below) uses this older syntax, the command must have the form:

```
rcp ruser1@rhost1:path1 rhost2.ruser2:path2
```

since the target is interpreted by *rhost1*. A common problem that is encountered is when two remote files are to be copied to a remote target that specifies a remote user. If the two remote source systems, *rhost1* and *rhost2*, each expect a different form for the remote target, the command:

```
rcp rhost1:path1 rhost2:path2 rhost3.ruser3:path3
```

will certainly fail on one of the source systems. Perform such a transfer using two separate commands.

DIAGNOSTICS

Diagnostics can occur from both the local and remote hosts. Those that occur on the local host before the connection is completely established are written to standard error. Once the connection is established, any error messages from the remote host are written to standard output, like any other data.

`Error! could not retrieve authentication type.`

`Please notify sys admin.`

There are two authentication mechanisms used by **`rcp`**. One authentication mechanism is based on Kerberos and the other is not. The type of authentication mechanism is obtained from a system file which is updated by inetsvcs_sec(1M). If the system file does not contain known authentication types, the above error is displayed.

AUTHOR

`rcp` was developed by the University of California, Berkeley.

SEE ALSO

cp(1), ftp(1), remsh(1), remshd(1M), inetsvcs_sec(1M), rcmd(3N), hosts(4), hosts.equiv(4).

`ftp` chapter in *Using Internet Services*.

NAME

rcp - remote file copy

SYNOPSIS

Copy Single File

rcp [**-k** *realm*] [**-P**] [**-p**] [**-S** *size*] [**-R** *size*] *source_file1 dest_file*

Copy Multiple Files

rcp [**-k** *realm*] [**-P**] [**-p**] [**-S** *size*] [**-R** *size*] *source_file1* [*source_file2*]... *dest_dir*

Copy One or More Directory Subtrees

rcp [**-k** *realm*] [**-P**] [**-p**] [**-S** *size*] [**-R** *size*] **-r** *source_dir1* [*source_dir2*]... *dest_dir*

Copy Files and Directory Subtrees

rcp [**-k** *realm*] [**-P**] [**-p**] [**-S** *size*] [**-R** *size*] **-r** *file_or_dir1* [*file_or_dir2*]... *dest_dir*

DESCRIPTION

The **rcp** command copies files, directory subtrees, or a combination of files and directory subtrees from one or more systems to another. In many respects, it is similar to the **cp** command (see *cp*(1)).

To use **rcp**, you must have read access to files being copied, and read and search (execute) permission on all directories in the directory path.

In a Kerberos V5 Network Authentication environment, **rcp** uses the Kerberos V5 protocol while initiating the connection to a remote host. The authorization mechanism is dependent on the command line options used to invoke **remshd** on the remote host (i.e., **-K**, **-R**, **-r**, or **-k**). Kerberos authentication and authorization rules are described in the Secure Internet Services man page, *sis*(5).

Although Kerberos authentication and authorization may apply, the Kerberos mechanism is **not** applied when copying files. The files are still transferred in cleartext over the network.

Options and Arguments

rcp recognizes the following options and arguments:

source_file
source_dir The name of an existing file or directory on a local or remote machine that you want copied to the specified destination. Source file and directory names are constructed as follows:

user_name@*hostname*:*pathname*/*filename*

or

user_name@*hostname*:*pathname*/*dirname*

Component parts of file and directory names are described below. If multiple existing files and/or directory subtrees are specified (*source_file1*, *source_file2*, ..., etc.), the destination must be a directory. Shell file name expansion is allowed on both local and remote systems. Multiple files and directory subtrees can be copied from one or more systems to a single destination directory with a single command.

dest_file The name of the destination file. If host name and path name are not specified, the existing file is copied into a file named *dest_file* in the current directory on the local system. If *dest_file* already exists and is writable, the existing file is overwritten. Destination file names are constructed the same way as source files except that file name expansion characters cannot be used.

dest_dir The name of the destination directory. If host name and path name are not specified, the existing file is copied into a directory named *dest_dir* in the current directory on the local system. If *dest_dir* already exists in the specified directory path (or current directory if not specified), a new directory named *dest_dir* is created underneath the existing directory named *dest_dir*. Destination directory names are constructed the same way as source directory tree names except that file name expansion characters cannot be used.

file_or_dir If a combination of files and directories are specified for copying (either explicitly or by file name expansion), only files are copied unless the **-r** option is specified. If the **-r** option is present, all files and directory subtrees whose names match the specified *file_or_dir* name are copied.

-k *realm* Obtain tickets from the remote host in the specified *realm* instead of the remote host's default realm as specified in the configuration file *krb.realms*.

-P Disable Kerberos authentication. Only applicable in a secure environment based on Kerberos V5. If the remote host has been configured to prevent non-secure access, using this option would result in the generic error,

```
rcmd: connect: <hostname>: Connection refused
```

See DIAGNOSTICS in *remshd*(1M) for more details.

-p Preserve (duplicate) modification times and modes (permissions) of source files, ignoring the current setting of the **umask** file creation mode mask. If this option is specified, **rcp** preserves the sticky bit only if the target user is superuser.

If the **-p** option is not specified, **rcp** preserves the mode and owner of *dest_file* if it already exists; otherwise **rcp** uses the mode of the source file modified by the **umask** on the destination host. Modification and access times of the destination file are set to the time when the copy was made.

-S *size* This option sets the size of the socket send buffer.

-R *size* This option sets the size of the socket receive buffer.

-r Recursively copy directory subtrees rooted at the source directory name. If any directory subtrees are to be copied, **rcp** recursively copies each subtree rooted at the specified source directory name to directory *dest_dir*. If *source_dir* is being copied to an existing directory of the same name, **rcp** creates a new directory *source_dir* within *dest_dir* and copies the subtree rooted at *source_dir* to *dest_dir***/***source_dir*. If *dest_dir* does not exist, **rcp** creates it and copies the subtree rooted at *source_dir* to *dest_dir*.

Constructing File and Directory Names

As indicated above, file and directory names contain one, two, or four component parts:

user_name	Login name to be used for accessing directories and files on remote system.
hostname	Hostname of remote system where directories and files are located.
pathname	Absolute directory path name or directory path name relative to the login directory of user *user_name*.
filename	Actual name of source or destination file. File name expansion is allowed on source file names.
dirname	Actual name of source or destination directory subtree. File name expansion is allowed on source directory names.

r

Each *file* or *directory* argument is either a remote file name of the form *hostname***:***path*, or a local file name (with a slash (**/**) before any colon (**:**)). *hostname* can be either an official host name or an alias (see *hosts*(4)). If *hostname* is of the form *ruser@rhost*, *ruser* is used on the remote host instead of the current user name. An unspecified *path* (that is, *hostname***:**) refers to the remote user's login directory. If *path* does not begin with **/**, it is interpreted relative to the remote user's login directory on *hostname*. Shell metacharacters in remote *paths* can be quoted with backslash (\), single quotes (**''**), or double quotes (**""**), so that they will be interpreted remotely.

rcp does not prompt for passwords. In a non-secure or traditional environment, user authorization is checked by determining if the current local user name or any user name specified via *ruser* exists on *rhost*. In a Kerberos V5 Network Authentication or secure environment, the authorization method is dependent upon the command line options for **remshd** (see *remshd*(1M) for details). In either case, remote command execution via *remsh*(1) and *rcmd*(3N) must be allowed and *remshd*(1M) must be executable on the remote host.

Third-party transfers in the form:

```
rcp ruser1@rhost1:path1 ruser2@rhost2:path2
```

are performed as:

```
remsh rhost1 -l ruser1 rcp path1 ruser2@rhost2:path2
```

Therefore, for a such a transfer to succeed, *ruser2* on *rhost2* must allow access by *ruser1* from *rhost1* (see *hosts.equiv*(4)).

WARNINGS

The **rcp** routine is confused by any output generated by commands in a **.cshrc** file on the remote host (see *csh*(1)).

Copying a file onto itself, for example:

```
rcp path 'hostname':path
```

may produce inconsistent results. The current HP-UX version of **rcp** simply copies the file over itself. However, some implementations of **rcp**, including some earlier HP-UX implementations, corrupt the file. In addition, the same file may be referred to in multiple ways, for example, via hard links, symbolic links, or NFS. It is not guaranteed that **rcp** will correctly copy a file over itself in all cases.

Implementations of **rcp** based on the 4.2BSD version (including the implementations of **rcp** prior to HP-UX 7.0) require that remote users be specified as *rhost*.*ruser*. If the first remote host specified in a third party transfer (*rhost1* in the example below) uses this older syntax, the command must have the form:

```
rcp ruser1@rhost1:path1 rhost2.ruser2:path2
```

since the target is interpreted by *rhost1*. A common problem that is encountered is when two remote files are to be copied to a remote target that specifies a remote user. If the two remote source systems, *rhost1* and *rhost2*, each expect a different form for the remote target, the command:

```
rcp rhost1:path1 rhost2:path2 rhost3.ruser3:path3
```

will certainly fail on one of the source systems. Perform such a transfer using two separate commands.

DIAGNOSTICS

Diagnostics can occur from both the local and remote hosts. Those that occur on the local host before the connection is completely established are written to standard error. Once the connection is established, any error messages from the remote host are written to standard output, like any other data.

Error! could not retrieve authentication type.

Please notify sys admin.

There are two authentication mechanisms used by **rcp**. One authentication mechanism is based on Kerberos and the other is not. The type of authentication mechanism is obtained from a system file which is updated by inetsvcs_sec(1M). If the system file does not contain known authentication types, the above error is displayed.

AUTHOR

rcp was developed by the University of California, Berkeley.

SEE ALSO

cp(1), ftp(1), remsh(1), remshd(1M), inetsvcs_sec(1M), rcmd(3N), hosts(4), hosts.equiv(4), sis(5).

ftp chapter in *Using Internet Services*.

NAME

rcs - change RCS file attributes

SYNOPSIS

rcs [*options*] *file* ...

DESCRIPTION

rcs creates new RCS files or changes attributes of existing ones. An RCS file contains multiple revisions of text, an access list, a change log, descriptive text, and some control attributes. For **rcs** to work, the user's login name must be on the access list, except if the access list is empty, if the user is the owner of the file or the superuser, or if the **-i** option is present.

The user of the command must have read/write permission for the directory containing the RCS file and read permission for the RCS file itself. **rcs** creates a semaphore file in the same directory as the RCS file to prevent simultaneous update. For changes, **rcs** always creates a new file. On successful completion, **rcs** deletes the old one and renames the new one. This strategy makes links to RCS files useless.

Files ending in **,v** are RCS files; all others are working files. If a working file is given, **rcs** tries to find the corresponding RCS file first in directory **./RCS**, then in the current directory, as explained in *rcsintro*(5).

Options

rcs recognizes the following options:

-a*logins* Appends the login names appearing in the comma-separated list *logins* to the access list of the RCS file.

-A*oldfile* Appends the access list of *oldfile* to the access list of the RCS file.

-c "*string***"** Sets the comment leader to *string*. The comment leader is printed before every log message line generated by the keyword **Log** during check out (see *co*(1)). This is useful for programming languages without multi-line comments. During **rcs -i** or initial **ci**, the comment leader is guessed from the suffix of the working file. Note, a comment leader is inserted at the beginning of each line of log information. The comment leader is determined by the suffix used with the file name, as in foo.c, or foo.sh, or foo.p. Note you can specify a different comment leader through the "rcs" command. The following table shows the comment leader associated with each file name suffix:

SUFFIX	FILES	Comment Character
c	c	'*'
C	C Header	'*'
sh	shell	'#'
s	Assembly	'#'
p	pascal	'*'
r	ratfor	'#'
e	efl	'#'
l	lex	'*'
y	yacc	'*'
yr	yacc-rarfor	'*'
ye	yacc-efl	'*'
ml	mocklisp	';'
mac	macro	';'
f	fortran	'c'
ms	ms-macros	'\'
me	me-macros	'\'
""	empty suffix	'#'
nil	unknown suffix	'""'

-e[*logins*]
Erases the login names appearing in the comma-separated list *logins* from the access list of the RCS file. If *logins* is omitted, the entire access list is erased.

-i Creates and initializes a new RCS file, but does not deposit any revision. If the RCS file has no path prefix, **rcs** tries to place it first into the subdirectory **./RCS**, then into the current

r

directory. If the RCS file already exists, an error message is printed.

-l[*rev*]
: Locks the revision with number *rev*. If a branch is given, the latest revision on that branch is locked. If *rev* is omitted, the latest revision on the trunk is locked. Locking prevents overlapping changes. A lock is removed with **ci** or **rcs -u** (see below).

-L Sets locking to **strict**. Strict locking means that the owner of an RCS file is not exempt from locking for check in. This option should be used for files that are shared.

-n*name*[**:**[*rev*]]
: Associates the symbolic name *name* with the branch or revision *rev*. **rcs** prints an error message if *name* is already associated with another number. If *rev* is omitted, the symbolic name is associated with the latest revision on the trunk. If **:***rev* is omitted, the symbolic name is deleted.

-N*name*[**:**[*rev*]]
: Same as **-n**, except that it overrides a previous assignment of *name*.

-o*range*
: Deletes ("obsoletes") the revisions given by *range*. A range consisting of a single revision number means that revision. A range consisting of a branch number means the latest revision on that branch. A range of the form *rev1*–*rev2* means revisions *rev1* to *rev2* on the same branch, **-***rev* means from the beginning of the branch containing *rev* up to and including *rev*, and rev**-** means from revision *rev* to the head of the branch containing *rev*. None of the outdated revisions can have branches or locks.

-q Quiet mode; diagnostics are not printed.

-s*state*[**:***rev*]
: Sets the state attribute of the revision *rev* to *state*. If *rev* is omitted, the latest revision on the trunk is assumed. If *rev* is a branch number, the latest revision on that branch is assumed. Any identifier is acceptable for *state*. A useful set of states is **Exp** (for experimental), **Stab** (for stable), and **Rel** (for released). By default, **ci** sets the state of a revision to **Exp**.

-t[*txtfile*]
: Writes descriptive text into the RCS file (deletes the existing text). If *txtfile* is omitted, **rcs** prompts the user for text supplied from the standard input, terminated with a line containing a single **.** or Ctrl-D. Otherwise, the descriptive text is copied from the file *txtfile*. If the **-i** option is present, descriptive text is requested even if **-t** is not given. The prompt is suppressed if the standard input is not a terminal.

-u[*rev*]
: Unlocks the revision with number *rev*. If a branch is given, the latest revision on that branch is unlocked. If *rev* is omitted, the latest lock held by the user is removed. Normally, only the locker of a revision may unlock it. Somebody else unlocking a revision breaks the lock. This causes a mail message to be sent to the original locker. The message contains a commentary solicited from the breaker. The commentary is terminated with a line containing a single **.** or Control-D.

-U Sets locking to non-strict. Non-strict locking means that the owner of a file need not lock a revision for check in. This option should *not* be used for files that are shared. The default (**-L** or **-U**) is determined by the system administrator.

Access Control Lists (ACLs)

Do not add optional ACL entries to an RCS file, because they are deleted when the file is updated. The resulting access modes for the new file might not be as desired.

DIAGNOSTICS

The RCS filename and the revisions outdated are written to the diagnostic output. The exit status always refers to the last RCS file operated upon, and is 0 if the operation was successful; 1 if unsuccessful.

EXAMPLES

Add the names **jane**, **mary**, **dave**, and **jeff** to the access list of RCS file **vision,v**:

```
rcs -ajane,mary,dave,jeff vision
```

Set the comment leader to *tab*`*` for file **`vision`**:

```
rcs -c'tab*' vision
```

Associate the symbolic name **`sso/6_0`** with revision **`38.1`** of file **`vision`**:

```
rcs -Nsso/6_0:38.1 vision
```

Lock revision **`38.1`** of file **`vision,v`** so that only the locker is permitted to check in (see *ci*(1)) the next revision of the file. This command prevents two or more people from simultaneously revising the same file and inadvertently overwriting each other's work.

```
rcs -l38.1 vision,v
```

WARNINGS

All **`rcs`** command options are available to anyone whose name appears in the file access list, including those to add and delete names in the access list, change strict locking, etc. If these options must be restricted, other security methods should be employed. Also see previous note regarding Access Control Lists.

AUTHOR

`rcs` was developed by Walter F. Tichy.

SEE ALSO

co(1), ci(1), rcsdiff(1), rcsmerge(1), rlog(1), rcsfile(4), acl(5), rcsintro(5).

NAME

rcsdiff - compareRCS revisions

SYNOPSIS

rcsdiff [**-bcefhn**] [**-r***rev1*] [**-r***rev2*] *file* ...

DESCRIPTION

rcsdiff compares two revisions of each given RCS file and creates output very similar to **diff** (see *diff*(1)). A file name ending in **,v** is an RCS file name, otherwise it is a working file name. **rcsdiff** derives the working file name from the RCS file name and vice versa, as explained in *rcsintro*(5). Pairs consisting of both an RCS and a working file name can also be specified.

rcsdiff recognizes the following options:

-b Same as described in *diff*(1);

-e Same as described in *diff*(1);

-f Same as described in *diff*(1);

-h Same as described in *diff*(1);

-n Generate an edit script of the format used by RCS.

-c[*n*]

Generate a diff with lines of context. The default is to present 3 lines of context. To change, specify *n*; for example, **-c10** gives 10 lines of context.

-c modifies the output format slightly from the normal *diff*(1) output. The "context" output begins with identification of the files involved and their creation dates, then each change is separated by a line with a dozen ***** (asterisks). Lines removed from *file1* are marked with **-** (dashes); those added to *file2* with **+** (pluses). Lines that are changed from one file to the other are marked in both files with **!** (exclamation marks).

If both *rev1* and *rev2* are omitted, **rcsdiff** compares the latest revision on the trunk with the contents of the corresponding working file. This is useful for determining what was changed since the last check-in.

If *rev1* is given, but *rev2* is omitted, **rcsdiff** compares revision *rev1* of the RCS file with the contents of the corresponding working file.

If both *rev1* and *rev2* are given, **rcsdiff** compares revisions *rev1* and *rev2* of the RCS file.

Both *rev1* and *rev2* can be given numerically or symbolically.

EXAMPLES

Compare the latest trunk revision of RCS file **f.c,v** and the contents of working file **f.c**:

```
rcsdiff f.c
```

Compare the revisions 1.1 and 1.2 in the RCS file **foo.c,v**:

```
rcsdiff -r1.1 -r1.2 foo.c
```

AUTHOR

rcsdiff was developed by Walter F. Tichy.

SEE ALSO

ci(1), co(1), diff(1), ident(1), rcs(1), rcsmerge(1), rlog(1), rcsfile(4), rcsintro(5).

NAME

rcsmerge - merge RCS revisions

SYNOPSIS

rcsmerge -r*rev1* [**-r** *rev2*] [**-p**] *file*

DESCRIPTION

rcsmerge incorporates the changes between *rev1* and *rev2* of an RCS file into the corresponding working file. If **-p** is given, the result is printed on the standard output; otherwise the result overwrites the working file.

A file name ending in **,v** is an RCS file name; otherwise it is a working file name. **rcsmerge** derives the working file name from the RCS file name and vice versa, as explained in *rcsintro*(5). A pair consisting of both an RCS and a working file name can also be specified.

rev1 cannot be omitted. If *rev2* is omitted, the latest revision on the trunk is assumed. Both *rev1* and *rev2* can be given numerically or symbolically.

rcsmerge prints a warning if there are overlaps, and delimits the overlapping regions as explained for the **-j** option of *co*(1). The command is useful for incorporating changes into a checked-out revision.

EXAMPLES

Suppose you have released revision 2.8 of **f.c**. Assume furthermore that you just completed revision 3.4 when you receive updates to release 2.8 from someone else. To combine the updates to 2.8 and your changes between 2.8 and 3.4, put the updates to 2.8 into file **f.c** and execute:

```
rcsmerge -p -r2.8 -r3.4 f.c >f.merged.c
```

Then examine **f.merged.c**. Alternatively, if you want to save the updates to 2.8 in the RCS file, check them in as revision 2.8.1.1 and execute **co -j**:

```
ci -r2.8.1.1 f.c
co -r3.4 -j2.8:2.8.1.1 f.c
```

As another example, the following command undoes the changes between revision 2.4 and 2.8 in your currently checked out revision in **f.c**:

```
rcsmerge -r2.8 -r2.4 f.c
```

Note the order of the arguments, and that **f.c** is overwritten.

WARNINGS

rcsmerge does not work for files that contain lines with a single **.**.

AUTHOR

rcsmerge was developed by Walter F. Tichy.

SEE ALSO

ci(1), co(1), merge(1), ident(1), rcs(1), rcsdiff(1), rlog(1), rcsfile(4).

NAME

rdist - remote file distribution program

SYNOPSIS

rdist [**-bhinqvwyMR**] [**-f** *distfile*] [**-d** *var***=***value*] [**-m** *host*] [*label...*]

rdist [**-bhinqvwyMR**] **-c** *name...* [*login@*]*host*[:*dest*]

DESCRIPTION

rdist facilitates the maintaining of identical copies of files over multiple hosts. It preserves the owner, group, mode, and modification time of files if possible and can update programs that are executing.

-f *distfile* Specify a *distfile* for **rdist** to execute. *distfile* contains a sequence of entries that specify the files to be copied, the destination hosts, and what operations to perform to do the updating. The format of *distfile* is described in detail later. If *distfile* is **-**, the standard input is used. If no **-f** option is present, the program looks first for a file called **distfile**, then **Distfile** in the local host's working directory to use as the input.

-d *var=value*
Define *var* to have *value*. The **-d** option is used to define variable definitions in the *distfile*. *value* can be an empty string, one name, or a list of name separated by tabs and/or spaces and enclosed by a pair of parentheses. However, if the variable specified is already defined in the *distfile*, the **-d** option has no effect (because the *distfile* overrides the **-d** option).

-m *host* Limit which machines are to be updated. Multiple **-m** arguments can be given to limit updates to a subset of hosts that are listed in the *distfile*.

label Label of a command to execute. The label must be defined in *distfile*.

-c *name...* The **-c** option forces **rdist** to interpret the remaining arguments as a small *distfile*. The equivalent distfile is as follows.

(*name* ...) -> [*login@*]*host*
install [*dest*] ;

-n Print the commands without executing them. This option is useful for debugging *distfile*.

-q Quiet mode. Files that are being modified are normally printed on standard output. The **-q** option suppresses this.

-R Remove extraneous files. If a directory is being updated, any files that exist on the remote host that do not exist in the master directory are removed. This is useful for maintaining truly identical copies of directories.

-h Follow symbolic links. Copy the file that the link points to rather than the link itself.

-i Ignore unresolved links. **rdist** will normally try to maintain the link structure of files being transferred and warn the user if it cannot find all the links.

-v Verify that the files are up to date on all the hosts. Any files that are out of date will be displayed but no files will be changed nor any mail sent.

-w Whole mode. The whole file name is appended to the destination directory name. Normally, only the last component of a name is used when renaming files. This will preserve the directory structure of the files being copied instead of flattening the directory structure. For example, renaming a list of files such as (**dir1/f1 dir2/f2**) to **dir3** would create files **dir3/dir1/f1** and **dir3/dir2/f2** instead of **dir3/f1** and **dir3/f2**.

-y Younger mode. Files are normally updated if their *mtime* and *size* (see *stat*(2)) disagree. The **-y** option causes **rdist** not to update files that are younger than the master copy. This can be used to prevent newer copies on other hosts from being replaced. A warning message is printed for files which are newer than the master copy.

-b Binary comparison. Perform a binary comparison and update files if they differ rather than comparing dates and sizes.

-M Check that mode, ownership, and group are the same in addition to any other form of comparison that is in effect. This option will cause files to be replaced but will only correct the problem with a directory and print a warning message.

The *distfile* used by **rdist** contains a sequence of entries that specify the files to be copied, the destination hosts, and what operations to perform to do the updating. Each entry has one of the following formats.

> *variable_name* = *name_list*
> [*label*:] *source_list* -> *destination_list command_list*
> [*label*:] *source_list* :: *time_stamp_file command_list*

The first format is used for defining variables. The second format is used for distributing files to other hosts. The third format is used for making lists of files on the local host that have been changed since some given date. (See *EXAMPLES*.)

variable_name
: Specify the name of a variable.

name_list List of names (such as list of hosts or lists of files) separated by tabs and/or spaces and enclosed by parentheses.

source_list
: Specify a list of files and/or directories on the local host to be used as the master copy for distribution. Each file in the *source_list* is added to a list for changes, *if* the file is out of date on the host that is being updated (second format), or *if* the file is newer than the time stamp file (third format). *source_list* may contain a single name, or multiple names separated by tabs and/or spaces and enclosed by parentheses.

destination_list
: List of hosts to which these files are to be copied. *destination_list* may contain a single name, or multiple names separated by tabs and/or spaces and enclosed by parentheses.

time_stamp_file
: Specify a given date to generate a list of files on the local host that were modified since that date.

label: Labels are optional. They are used to identify a command for partial updates.

command_list
: Specifies a list of commands to be performed.

: The command list consists of zero or more commands of the following format.

> **install** [*options*] *opt_dest_name*;
> **notify** *name_list*;
> **except** *name_list*;
> **except_pat** *pattern_list*;
> **special** *name_list string*;

The **install** command is used to copy out-of-date files and/or directories. Each source file is copied to each host in the destination list. Directories are recursively copied in the same way. *opt_dest_name* is an optional parameter to rename files. If no **install** command appears in the command list or the destination name is not specified, *source_list* is used. Directories in the path name will be created if they do not exist on the remote host. To help prevent disasters, a non-empty directory on a target will never be replaced with a regular file or a symbolic link. However, under the **-R** option a non-empty directory will be removed if the corresponding filename is completely absent on the master host. The *options* are **-b**,**-h**,**-i**, **-v**,**-w**,**-y**, **-M**, and **-R**, and have the same semantics as options on the command line, except that they only apply to the files in the specified *source_list*. The login name used on the destination host is the same as on the local host, unless the destination name is of the form "login@host".

The **notify** command is used to mail the list of files updated (and any errors that may have occurred) to the listed names, in *name_list*. If no **@** appears in the name, the destination host is appended to the name (e.g., name1@host, name2@host, ...).

The **except** command is used to update all of the files in the source list, *except* for the files listed in *name_list*. This is usually used to copy everything in a directory except certain files.

The **except_pat** command is like the **except** command except that *pattern_list* is a list of regular expressions (see *ed*(1) for details). If one of the patterns matches some string within a file name, that file will be ignored. Note that since the backslash (\) is a quote character, it must be doubled to become part of the regular expression. Variables are expanded in *pattern_list* but not shell file pattern matching characters. To include a **$**, it must be escaped with the backslash.

The **special** command is used to specify *sh*(1) commands that are to be executed on the remote host after the file in *name_list* is updated or installed. If the *name_list* is omitted then the shell commands will be executed for every file updated or installed. The shell variable 'FILE' is set to the current filename before executing the commands in *string*. *string* starts and ends with double quotes (") and can cross multiple lines in *distfile*. Multiple commands to the shell should be separated by semi-colons (**;**). Commands are executed in the user's home directory on the host being updated. The **special** command can be used, for example, to rebuild private databases after a program has been updated. Shell variables cannot be used in the command because there is no escape mechanism for the **$** character.

Newlines, tabs, and blanks are only used as separators and are otherwise ignored. Comments begin with **#** and end with a newline.

A generalized way of dynamically building variable lists is provided by using a backquote syntax much like the shell. In this way, arbitrary commands that generate stdout with space-separated words may be used to build the list (see the use of **cat** command in the examples).

Variables to be expanded begin with **$** followed by the variable name enclosed in curly braces.

The shell meta-characters **[**, **]**, **{**, **}**, *****, and **?** are recognized and expanded (on the local host only) in the same way as *csh*(1). They can be escaped with a backslash. The **~** character is also expanded in the same way as *csh* but is expanded separately on the local and destination hosts. When the **-w** option is used with a file name that begins with **~**, everything except the home directory is appended to the destination name. File names which do not begin with **/** or **~** use the destination user's home directory as the root directory for the rest of the file name.

EXAMPLES

The following is a small example.

```
HOSTS = ( matisse root@arpa )

FILES = ( /usr/lib /usr/bin /usr/local/games
     /usr/include/{*.h,{sys,rpc*,arpa}/*.h}
     /usr/man/man? `cat ./std-files` )

EXLIB = ( Mail.rc aliases aliases.dir aliases.pag crontab dshrc
     sendmail.cf sendmail.fc sendmail.hf sendmail.st uucp vfont )

${FILES} -> ${HOSTS}
     install -R ;
     except /usr/lib/${EXLIB} ;
     except /usr/local/games/lib ;
     special /usr/sbin/sendmail " /usr/sbin/sendmail -bz" ;

srcs:
/usr/local/src -> arpa
     except_pat ( \\.o$ /SCCS\$ ) ;

IMAGEN = (ips dviimp catdvi)

imagen:
/usr/local/${IMAGEN} -> arpa
     install /usr/local/lib ;
     notify ralph ;

${FILES} :: stamp.cory
     notify root@cory ;
```

AUTHOR

rdist was developed by the University of California, Berkeley.

FILES

distfile	Input command file.
/tmp/rdist*	Temporary file for update lists.

HISTORY

rdist appeared in the 4.3 Berkeley Software Distribution.

SEE ALSO

sh(1), csh(1), stat(2)

DIAGNOSTICS

A complaint about mismatch of rdist version numbers may mean that an executable **rdist** is not in the shell's path on the remote system.

BUGS

Source files must reside on the local host where **rdist** is executed.

There is no easy way to have a special command executed after all files in a directory have been updated.

Variable expansion only works for name lists and in the **special** command string; there should be a general macro facility.

rdist aborts on files that have a negative mtime (before Jan 1, 1970).

rdist does carry the atime when installing a file but will preserve it on an updated file.

There should be a 'force' option to allow replacement of non-empty directories by regular files or symlinks.

NAME

read - read a line from standard input

SYNOPSIS

read [**-r**] *var* ...

DESCRIPTION

read reads a single line from standard input. The line is split into fields as when processed by the shell (refer to shells in SEE ALSO); the first field is assigned to the first variable *var*, the second field to the second variable *var*, and so forth. If there are more fields than there are specified *var* operands, the remaining fields and their intervening separators are assigned to the last *var*. If there are more *var*s than fields, the remaining *var*s are set to empty strings.

The setting of variables specified by the *var* operands affect the current shell execution environment.

Standard input to **read** can be redirected from a text file.

Since **read** affects the current shell execution environment, it is usually provided as a normal shell special (built-in) command. Thus, if it is called in a subshell or separate utility execution environment similar to the following, it does not affect the shell variables in the caller's environment:

```
(read foo)
nohup read ...
find . -exec read ... ;
```

Options and Arguments

read recognizes the following options and command-line arguments:

-r	Do not treat a backslash character in any special way. Consider each backslash to be part of the input line.
var	The name of an existing or non-existing shell variable.

EXTERNAL INFLUENCES

Environment Variables

IFS determines the internal field separators used to delimit fields.

RETURN VALUE

read exits with one of the following values:

0	Successful completion.
>**0**	End-of-file was detected or an error occurred.

EXAMPLES

Print a file with the first field of each line moved to the end of the line.

```
while read -r xx yy
do
        printf "%s %s \n" "$yy" "$xx"
done < input_file
```

SEE ALSO

csh(1), ksh(1), sh-posix(1), sh(1).

STANDARDS CONFORMANCE

read: SVID2, XPG2, XPG3, XPG4, POSIX.2 FIPS

NAME

readmail - read mail from a mail folder or incoming mailbox

SYNOPSIS

readmail [**-ahnp**] [**-f** *folder*] [*number-list*|*pattern*]

DESCRIPTION

The **readmail** program displays messages from your incoming mailbox or a specified mail folder.

Within the **elm** mail system (see *elm*(1) with no operands and optionally the **-h** or **-n** option, **readmail** displays the appropriate headers and the body of the current message.

With the *number-list* operand and no options, **readmail** displays the corresponding messages and a summary of the headers from your incoming mailbox.

With the *pattern* operand and no options, **readmail** displays the first message that matches the pattern and a summary of the headers from your incoming mailbox.

Options

readmail supports the following options.

-a	Print all messages that match *pattern*. If no pattern was specified, this option is ignored.
-f *folder*	Use file *folder* for the operations instead of the incoming mailbox.
-h	Include the entire header of the matched message or messages when displaying their text. The default is to display the **From:**, **Date:**, and **Subject:** lines only.
-n	Exclude all headers.
-p	Put form feeds (**Ctrl-L**) between message headers. This is useful when printing sets of messages.

Operands

readmail supports the following operands.

number-list	A blank-separated list of the ordinal locations of messages in the mail file (i.e., their "message numbers"), up to 25 at a time. The character **$** means the last message in the mail file. Similarly, ***** represents every message in the file (i.e., **1 2 3** ... **$**) The message numbers are sorted into ascending order. Thus, **1 3 2** produces the same output as **1 2 3**.
pattern	A string that is present in one of the messages. This pattern can be typed in directly (no quotes) if the words are separated by a single space in the actual message. The pattern matching is case sensitive, so **Hello** and **hello** are not equivalent. Leading digits (on the first word) are not permitted; however, you can precede them with a space and quote the entire string, if the space occurs in the message, as in **" 1st item of business"** .

r

EXAMPLES

If you are using **vi** to reply to a message from within the **elm** mail system, you can insert the text of the current message with the command:

```
:r !readmail
```

If you define an alias similar to:

```
alias rd='readmail $ | page'     (Bourne, Korn, or POSIX shell)
alias rd 'readmail $ | page'     (C shell)
```

you can use it with a program such **newmail** to peruse mail as it arrives, without needing to start a mail system (see *newmail*(1)).

AUTHOR

readmail was developed by HP.

FILES

`/var/mail/`*loginname*	Incoming mailbox
`$HOME/.elm/readmail`	Temporary file for **`elm`**

SEE ALSO

elm(1), newmail(1), vi(1).

NAME

remsh, rexec - execute from a remote shell

SYNOPSIS

remsh *host* [**-l** *username*] [**-n**] *command*
host [**-l** *username*] [**-n**] *command*

rexec *host* [**-l** *username*] [**-n**] *command*

DESCRIPTION

remsh connects to the specified *host* and executes the specified *command*. The host name can be either the official name or an alias as understood by **gethostbyname()** (see *gethostent*(3N) and *hosts*(4)). **remsh** copies its standard input (**stdin**) to the remote command, and the standard output of the remote command to its standard output (**stdout**), and the standard error of the remote command to its standard error (**stderr**). Hangup, interrupt, quit, terminate, and broken pipe signals are propagated to the remote command. **remsh** exits when the sockets associated with **stdout** and **stderr** of the remote command are closed. This means that **remsh** normally terminates when the remote command does (see *remshd*(1M)).

By default, **remsh** uses the following path when executing the specified *command*:

```
/usr/bin:/usr/ccs/bin:/usr/bin/X11:
```

remsh uses the default remote login shell with the **-c** option to execute the remote command. If the default remote shell is *csh*, csh sources the remote **.cshrc** file before the command. **remsh** cannot be used to run commands that require a terminal interface (such as **vi**) or commands that read their standard error (such as **more**). In such cases, use **rlogin** or **telnet** instead (see *rlogin*(1) and *telnet*(1)).

The remote account name used is the same as your local account name, unless you specify a different remote name with the **-l** option. This remote account name must be equivalent to the originating account; no provision is made for specifying a password with a command. For more details about **equivalent** hosts and how to specify them, see *hosts.equiv*(4). The files inspected by **remshd** on the remote host are **/etc/hosts.equiv** and **$HOME/.rhosts** (see *remshd*(1M)).

If *command*, is not specified, instead of executing a single command, you will be logged in on the remote host using **rlogin** (see *rlogin*(1)). Any **rlogin** options typed in on the command line are transmitted to **rlogin**. If *command* is specified, options specific to **rlogin** are ignored by **remsh**.

By default, **remsh** reads its standard input and sends it to the remote command because **remsh** has no way to determine whether the remote command requires input. The **-n** option redirects standard input to **remsh** from **/dev/null**. This is useful when running a shell script containing a **remsh** command, since otherwise remsh may use input not intended for it. The **-n** option is also useful when running **remsh** in the background from a job control shell, **/usr/bin/csh** or **/usr/bin/ksh**. Otherwise, **remsh** stops and waits for input from the terminal keyboard for the remote command. **/usr/bin/sh** automatically redirects its input from **/dev/null** when jobs are run in the background.

Host names for remote hosts can also be commands (linked to **remsh**) in the directory **/usr/hosts**. If this directory is specified in the **$PATH** environment variable, you can omit **remsh**. For example, if **remotehost** is the name of a remote host, **/usr/hosts/remotehost** is linked to **remsh**, and if **/usr/hosts** is in your search path, the command

```
remotehost command
```

executes **command** on **remotehost**, and the command

```
remotehost
```

is equivalent to

```
rlogin remotehost
```

The **rexec** command works the same as **remsh** except that it uses the **rexec()** library routine and **rexecd** for command execution (see *rexec*(3N) and *rexecd*(1M)) and does not support Kerberos authentication. **rexec** prompts for a password before executing the command instead of using **hosts.equiv** for authentication. It should be used in instances where a password to a remote account is known but there are insufficient permissions for **remsh**.

EXAMPLES

Shell metacharacters that are not quoted are interpreted on the local host; quoted metacharacters are interpreted on the remote host. Thus the command line:

```
remsh otherhost cat remotefile >> localfile
```

appends the remote file **remotefile** to the local file **localfile**, while the command line

```
remsh otherhost cat remotefile ">>" otherremotefile
```

appends **remotefile** to the remote file **otherremotefile**.

If the remote shell is **/usr/bin/sh**, the following command line sets up the environment for the remote command before executing the remote command:

```
remsh otherhost . .profile 2>&- \; command
```

The **2>&-** throws away error messages generated by executing **.profile** when *stdin* and *stdout* are not a terminal.

The following command line runs **remsh** in the background on the local system, and the output of the remote command comes to your terminal asynchronously:

```
remsh otherhost -n command &
```

The background **remsh** completes when the remote command does.

The following command line causes **remsh** to return immediately without waiting for the remote command to complete:

```
remsh otherhost -n "command 1>&- 2>&- &"
```

(See *remshd*(1M) and *sh*(1)). If your login shell on the remote system is *csh*, use the following form instead:

```
remsh otherhost -n "sh -c \"command 1>&- 2>&- &\""
```

RETURN VALUE

If **remsh** fails to set up the secondary socket connection, it returns 2. If it fails in some other way, it returns 1. If it fully succeeds in setting up a connection with **remshd**, it returns 0 once the remote command has completed. Note that the return value of **remsh** bears no relation to the return value of the remote command.

DIAGNOSTICS

Besides the errors listed below, errors can also be generated by the library functions **rcmd()** and **rresvport()** which are used by **remsh** (see *rcmd*(3N)). Those errors are preceded by the name of the library function that generated them. **remsh** can produce the following diagnostic messages:

Error! could not retrieve authentication type.
Please notify sys admin.

There are two authentication mechanisms used by **remsh**. One authentication mechanism is based on Kerberos and the other is not. The type of authentication mechanism is obtained from a system file which is updated by **inetsvcs_sec** (see *inetsvcs_sec*(1M)). If the system file does not contain known authentication types, the above error is displayed.

rlogin: ...

Error in executing **rlogin** (**rlogin** is executed when the user does not specify any commands to be executed). This is followed by the error message specifying why the execution failed.

shell/tcp: Unknown service

The "shell" service specification is not present in the **/etc/services** file.

Can't establish stderr

remsh cannot establish secondary socket connection for **stderr**.

*<system call>***:** ...

Error in executing system call. Appended to this error is a message specifying the cause of the failure.

r

There is no entry for you (user ID *uid*) in /etc/passwd
Check with the system administrator to see if your entry in the password file has been deleted by mistake.

WARNINGS

For security reasons, the **/etc/hosts.equiv** and **.rhosts** files should exist, even if empty, and should be readable and writable only by the owner. Note also that all information, including any passwords asked for, is passed unencrypted between the two hosts.

If **remsh** is run with an interactive command it hangs.

DEPENDENCIES

remsh is the same service as **rsh** on BSD systems. The name was changed due to a conflict with the existing System V command **rsh** (restricted shell).

AUTHOR

remsh was developed by the University of California, Berkeley.

FILES

/usr/hosts/* for version of the command invoked only with hostname

SEE ALSO

rlogin(1), remshd(1M), rexecd(1M), inetsvcs_sec(1M), gethostent(3N), rcmd(3N), rexec(3N), hosts.equiv(4), hosts(4).

NAME

remsh, rexec - execute from a remote shell

SYNOPSIS

remsh *host* [**-l** *username*] [**-f/F**] [**-k** *realm*] [**-P**] [**-n**] *command*
host [**-l** *username*] [**-f/F**] [**-k** *realm*] [**-P**] [**-n**] *command*

rexec *host* [**-l** *username*] [**-n**] *command*

DESCRIPTION

remsh connects to the specified *host* and executes the specified *command*. The host name can be either the official name or an alias as understood by **gethostbyname()** (see *gethostent*(3N) and *hosts*(4)). **remsh** copies its standard input (**stdin**) to the remote command, and the standard output of the remote command to its standard output (**stdout**), and the standard error of the remote command to its standard error (**stderr**). Hangup, interrupt, quit, terminate, and broken pipe signals are propagated to the remote command. **remsh** exits when the sockets associated with **stdout** and **stderr** of the remote command are closed. This means that **remsh** normally terminates when the remote command does (see *remshd*(1M)).

By default, **remsh** uses the following path when executing the specified *command*:

```
/usr/bin:/usr/ccs/bin:/usr/bin/X11:
```

remsh uses the default remote login shell with the **-c** option to execute the remote command. If the default remote shell is *csh*, csh sources the remote **.cshrc** file before the command. **remsh** cannot be used to run commands that require a terminal interface (such as **vi**) or commands that read their standard error (such as **more**). In such cases, use **rlogin** or **telnet** instead (see *rlogin*(1) and *telnet*(1)).

The remote account name used is the same as your local account name, unless you specify a different remote name with the **-l** option. In addition, the remote host account name must also conform to other rules which differ depending upon whether the remote host is operating in a Kerberos V5 Network Authentication, i.e., secure environment or not. In a non-secure, or traditional environment, the remote account name must be equivalent to the originating account; no provision is made for specifying a password with a command. For more details about **equivalent** hosts and how to specify them, see *hosts.equiv*(4). The files inspected by **remshd** on the remote host are **/etc/hosts.equiv** and **$HOME/.rhosts** (see *remshd*(1M)).

In a Kerberos V5 Network Authentication environment, the local host must be successfully authenticated before the remote account name is checked for proper authorization. The authorization mechanism is dependent on the command line options used to invoke **remshd** on the remote host (i.e., **-K**, **-R**, **-r**, or **-k**). For further information on Kerberos authentication and authorization see the Secure Internet Services man page, *sis*(5) and *remshd*(1M).

Although Kerberos authentication and authorization may apply, the Kerberos mechanism is *not* applied to the *command* or to its response. All information transferred between the local and remote host is still sent in cleartext over the network.

In a secure or Kerberos V5-based environment, the following command line options are available:

-f Forward the ticket granting ticket (TGT) to the remote system. The TGT is not forwardable from there.

-F Forward the TGT to the remote system and have it forwardable from there to another remote system. **-f** and **-F** are mutually exclusive.

-k *realm*
Obtain tickets from the remote host in the specified *realm* instead of the remote host's default realm as specified in the configuration file *krb.realms*.

-P Disable Kerberos authentication.

If a *command* is not specified, instead of executing a single command, you will be logged in on the remote host using **rlogin** (see *rlogin*(1)). Any **rlogin** options typed in on the command line are transmitted to **rlogin**. If no *command* and the option **-P** is specified, **rlogin** will be invoked with **-P** to indicate that Kerberos authentication (or secure access) is not required. This will mean that if a password is requested, the password will be sent in cleartext. If a *command* is specified, options specific to **rlogin** are ignored by **remsh**.

If a *command* and the option *-n* are specified, then standard input is redirected to **remsh** by **/dev/null**. If *-n* is not specified (the default case), **remsh** reads its standard input and sends the input to the remote command. This is because **remsh** has no way to determine whether the remote command requires input. This option is useful when running a shell script containing a **remsh** command, since otherwise remsh may use input not intended for it. The **-n** option is also useful when running **remsh** in the background from a job control shell, **/usr/bin/csh** or **/usr/bin/ksh**. Otherwise, **remsh** stops and waits for input from the terminal keyboard for the remote command. **/usr/bin/sh** automatically redirects its input from **/dev/null** when jobs are run in the background.

Host names for remote hosts can also be commands (linked to **remsh**) in the directory **/usr/hosts**. If this directory is specified in the **$PATH** environment variable, you can omit **remsh**. For example, if **remotehost** is the name of a remote host, **/usr/hosts/remotehost** is linked to **remsh**, and if **/usr/hosts** is in your search path, the command

remotehost command

executes **command** on **remotehost**, and the command

remotehost

is equivalent to

rlogin remotehost

The **rexec** command works the same as **remsh** except that it uses the **rexec()** library routine and **rexecd** for command execution (see *rexec*(3N) and *rexecd*(1M)) and does not support Kerberos authentication. **rexec** prompts for a password before executing the command instead of using **hosts.equiv** for authentication. It should be used in instances where a password to a remote account is known but there are insufficient permissions for **remsh**.

EXAMPLES

Shell metacharacters that are not quoted are interpreted on the local host; quoted metacharacters are interpreted on the remote host. Thus the command line:

```
remsh otherhost cat remotefile >> localfile
```

appends the remote file **remotefile** to the local file **localfile**, while the command line

```
remsh otherhost cat remotefile ">>" otherremotefile
```

appends **remotefile** to the remote file **otherremotefile**.

If the remote shell is **/usr/bin/sh**, the following command line sets up the environment for the remote command before executing the remote command:

```
remsh otherhost . .profile 2>&- \; command
```

r

The **2>&-** throws away error messages generated by executing **.profile** when *stdin* and *stdout* are not a terminal.

The following command line runs **remsh** in the background on the local system, and the output of the remote command comes to your terminal asynchronously:

```
remsh otherhost -n command &
```

The background **remsh** completes when the remote command does.

The following command line causes **remsh** to return immediately without waiting for the remote command to complete:

```
remsh otherhost -n "command 1>&- 2>&- &"
```

(See *remshd*(1M) and *sh*(1)). If your login shell on the remote system is *csh*, use the following form instead:

```
remsh otherhost -n "sh -c \"command 1>&- 2>&- &\""
```

RETURN VALUE

If **remsh** fails to set up the secondary socket connection, it returns 2. If it fails in some other way, it returns 1. If it fully succeeds in setting up a connection with **remshd**, it returns 0 once the remote command has completed. Note that the return value of **remsh** bears no relation to the return value of the remote command.

DIAGNOSTICS

Besides the errors listed below, errors can also be generated by the library functions **rcmd()** and **rresvport()** which are used by **remsh** (see *rcmd*(3N)). Those errors are preceded by the name of the library function that generated them. **remsh** can produce the following diagnostic messages:

Error! could not retrieve authentication type.

Please notify sys admin.
There are two authentication mechanisms used by **remsh**. One authentication mechanism is based on Kerberos and the other is not. The type of authentication mechanism is obtained from a system file which is updated by **inetsvcs_sec** (see *inetsvcs_sec*(1M)). If the system file does not contain known authentication types, the above error is displayed.

rlogin: ...
Error in executing **rlogin** (**rlogin** is executed when the user does not specify any commands to be executed). This is followed by the error message specifying why the execution failed.

shell/tcp: Unknown service
The "shell" service specification is not present in the **/etc/services** file.

Can't establish stderr
remsh cannot establish secondary socket connection for **stderr**.

<*system call*>**:** ...
Error in executing system call. Appended to this error is a message specifying the cause of the failure.

There is no entry for you (user ID *uid*) **in /etc/passwd**
Check with the system administrator to see if your entry in the password file has been deleted by mistake.

rcmd: connect: <hostname>: Connection refused
One cause for display of this generic error message could be due to the absence of an entry for *shell* in **/etc/inetd.conf** on the remote system. This entry may have been removed or commented out to prevent non-secure access.

Kerberos-specific errors are listed in *sis*(5).

WARNINGS

For security reasons, the **/etc/hosts.equiv** and **.rhosts** files should exist, even if empty, and should be readable and writable only by the owner.

If **remsh** is run with an interactive command it hangs.

DEPENDENCIES

remsh is the same service as **rsh** on BSD systems. The name was changed due to a conflict with the existing System V command **rsh** (restricted shell).

AUTHOR

remsh was developed by the University of California, Berkeley.

FILES

/usr/hosts/*	for version of the command invoked only with hostname

SEE ALSO

rlogin(1), remshd(1M), rexecd(1M), inetsvcs_sec(1M), gethostent(3N), rcmd(3N), rexec(3N), hosts.equiv(4), hosts(4), sis(5).

NAME

rev - reverse lines of a file

SYNOPSIS

rev [*file*] ...

DESCRIPTION

rev copies the named files to the standard output, reversing the order of characters in every line. If no file is specified, the standard input is copied.

EXTERNAL INFLUENCES

Environment Variables

LC_CTYPE determines the interpretation of text as single- and/or multi-byte characters.

If **LC_CTYPE** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**. If any internationalization variable contains an invalid setting, *rev* behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported.

NAME

rlog - print log messages and other information on RCS files

SYNOPSIS

rlog [*options*] *file* ...

DESCRIPTION

rlog prints information about RCS files. Files ending in **,v** are RCS files; all others are working files. If a working file is given, **rlog** tries to find the corresponding RCS file first in directory **./RCS**, then in the current directory, as explained in *rcsintro*(5).

rlog prints the following information for each RCS file: RCS file name, working file name, head (i.e., the number of the latest revision on the trunk), access list, locks, symbolic names, suffix, total number of revisions, number of revisions selected for printing, and descriptive text. This is followed by entries for the selected revisions in reverse chronological order for each branch. For each revision, **rlog** prints revision number, author, date/time, state, number of lines added/deleted (with respect to the previous revision), locker of the revision (if any), and log message. Without options, **rlog** prints complete information. The options below restrict this output.

Options

rlog recognizes the following options:

-d*dates* Print information about revisions whose check-in date and time fall within the ranges given by the semicolon-separated list of *dates*. A range of the form *d1***<***d2* or *d2***>***d1* selects the revisions that were deposited between *d1* and *d2* (inclusive). A range of the form **<***d* or *d***>** selects all revisions dated *d* or earlier. A range of the form *d***<** or **>***d* selects all revisions dated *d* or later. A range of the form *d* selects the single, latest revision dated *d* or earlier. The date/time strings *d*, *d1*, and *d2* are in the format explained in *co*(1). Quoting is normally necessary, especially for **<** and **>**. Note that the separator is a semicolon.

-h Print only RCS file name, working file name, head, access list, locks, symbolic names, and suffix.

-l[*lockers*] Print information about locked revisions. If the comma-separated list *lockers* of login names is given, only the revisions locked by the given login names are printed. If the list is omitted, all locked revisions are printed.

-L Ignore RCS files that have no locks set; convenient in combination with **-R**, **-h**, or **-l**.

-r*revisions* Print information about revisions given in the comma-separated list *revisions* of revisions and ranges. A range *rev1*-*rev2* means revisions *rev1* to *rev2* on the same branch, **-***rev* means revisions from the beginning of the branch up to and including *rev*, and *rev***-** means revisions starting with *rev* to the head of the branch containing *rev*. An argument that is a branch means all revisions on that branch. A range of branches means all revisions on the branches in that range.

-R Print only the name of the RCS file; convenient for translating a working file name into an RCS file name.

-s*states* Print information about revisions whose state attributes match one of the states given in the comma-separated list *states*.

-t Print the same as **-h**, plus the descriptive text.

-w[*logins*] Prints information about revisions checked in by users whose login names appearing in the comma-separated list *logins*. If *logins* is omitted, the user's login is assumed.

rlog prints the intersection of the revisions selected with the options **-d**, **-l**, **-s**, **-w**, and **-r**.

EXAMPLES

Print the names of all RCS files in the subdirectory named **RCS** that have locks:

```
rlog -L -R RCS/*,v
```

Print the headers of those files:

```
rlog -L -h RCS/*,v
```

Print the headers plus the log messages of the locked revisions:

```
rlog -L -l RCS/*,v
```

Print complete log information:

```
rlog RCS/*,v
```

Print the header and log messages of all revisions checked in after 1:00am on December 25th, 1991:

```
rlog -d">12/25/92, 1:00" RCS/*,v
```

Print the header and log messages of those revisions that were created between 10:00am and 2:00pm on July 4th, 1992:

```
rlog -d"07/04/92, 10:00 > 92/07/04, 14:00" RCS/*,v
```

DIAGNOSTICS

The exit status always refers to the last RCS file operated upon, and is 0 if the operation was successful, 1 if unsuccessful.

AUTHOR

rlog was developed by Walter F. Tichy.

SEE ALSO

ci(1), co(1), ident(1), rcs(1), rcsdiff(1), rcsmerge(1), rcsfile(4), rcsintro(5).

NAME

rlogin - remote login

SYNOPSIS

rlogin *rhost* [**-7**] [**-8**] [**-e***e*] [**-l** *username*]

rhost [**-7**] [**-8**] [**-e***e*] [**-l** *username*]

DESCRIPTION

The **rlogin** command connects your terminal on the local host to the remote host (*rhost*). **rlogin** acts as a virtual terminal to the remote system. The host name *rhost* can be either the official name or an alias as listed in the file **/etc/hosts** (see *hosts*(4)).

In a manner similar to the **remsh** command (see *remsh*(1)), **rlogin** allows a user to log in on an equivalent remote host, *rhost*, bypassing the normal login/password sequence. For more information about equivalent hosts and how to specify them in the files **/etc/hosts.equiv** and **.rhosts**, see *hosts.equiv*(4). The searching of the files **/etc/hosts.equiv** and **.rhosts** occurs on the remote host, and the **.rhosts** file must be owned by the remote user account.

If the originating user account is not equivalent to the remote user account, the originating user is prompted for the password of the remote account. If this fails, a login name and password are prompted for, as when **login** is used (see *login*(1)).

The terminal type specified by the current **TERM** environment variable is propagated across the network and used to set the initial value of your **TERM** environment variable on the remote host. Your terminal baud rate is also propagated to the remote host, and is required by some systems to set up the pseudo-terminal used by **rlogind** (see *rlogind*(1M)).

All echoing takes place at the remote site, so that (except for delays) the remote login is transparent.

If at any time **rlogin** is unable to read from or write to the socket connection on the remote host, the message **Connection closed** is printed on standard error and **rlogin** exits.

Options

rlogin recognizes the following options. Note that the options follow the *rhost* argument.

-7 Set the character size to seven bits. The eighth bit of each byte sent is set to zero (space parity).

-8 Use an eight-bit data path. This is the default HP-UX behavior.

To use eight-bit characters, the terminal must be configured to generate either eight-bit characters with no parity, or seven bit characters with space parity. The HP-UX implementation of **rlogind** (see *rlogind*(1M)) interprets seven bit characters with even, odd, or mark parity as eight-bit non-USASCII characters. You may also need to reconfigure the remote host appropriately (see *stty*(1) and *tty*(7)). Some remote hosts may not provide the necessary support for eight-bit characters. In this case, or if it is not possible to disable parity generation by the local terminal, use the **-7** option.

-e*e* Set the escape character to *e*. There is no space separating the option letter and the argument character. To start a line with the escape character, two of the escape characters must be entered. The default escape character is tilde (**~**). Some characters may conflict with your terminal configuration, such as **^S**, **^Q**, or backspace. Using one of these as the escape character may not be possible or may cause problems communicating with the remote host (see *stty*(1) and *tty*(7)).

-l *username* Set the user login name on the remote host to *username*. The default name is the current account name of the user invoking **rlogin**.

Escape Sequences

rlogin can be controlled with two-character escape sequences, in the form *ex*, where *e* is the escape character and *x* is a code character described below. Escape sequences are recognized only at the beginning of a line of input. The default escape character is tilde (**~**). It can be changed with the **-e** option.

The following escape sequences are recognized:

ey If *y* is NOT a code character described below, pass the escape character and *y* as characters to the remote host.

r

ee Pass the escape character as a character to the remote host.

e`.` Disconnect from the remote host.

e`!` Escape to a subshell on the local host. Use **`exit`** to return to the remote host.

If **`rlogin`** is run from a shell that supports job control (see *csh*(1), *ksh*(1), and *sh-posix*(1)), escape sequences can be used to suspend **`rlogin`**. The following escape sequences assume that **`^Z`** and **`^Y`** are set as the user's **`susp`** and **`dsusp`** characters, respectively (see *stty*(1) and *termio*(7)).

e`^Z` Suspend the **`rlogin`** session and return the user to the shell that invoked **`rlogin`**. The **`rlogin`** job can be resumed with the **`fg`** command (see *csh*(1), *ksh*(1), and *sh-posix*(1)). *e*`^Z` suspends both **`rlogin`** processes: the one transmitting user input to the remote login, and the one displaying output from the remote login.

e`^Y` Suspend the **`rlogin`** session and return the user to the shell that invoked **`rlogin`**. The **`rlogin`** job can be resumed with the **`fg`** command (see *csh*(1), *ksh*(1), and *sh-posix*(1)). *e*`^Y` suspends only the input process; output from the remote login continues to be displayed.

If you "daisy-chain" remote logins (for example, you **`rlogin`** from host A to host B and then **`rlogin`** from host B to host C) without setting unique escape characters, you can repeat the escape character until it reaches your chosen destination. For example, the first escape character, *e*, is seen as an escape character on host A; the second *e* is passed as a normal character by host A and seen as an escape character on host B; a third *e* is passed as a normal character by hosts A and B and accepted as a normal character by host C.

Remote Host Name As Command

The system administrator can arrange for more convenient access to a remote host (*rhost*) by linking **`remsh`** to **`/usr/hosts/`***rhost*, allowing use of the remote host name (*rhost*) as a command (see *remsh*(1)). For example, if **`remotehost`** is the name of a remote host and **`/usr/hosts/remotehost`** is linked to **`remsh`**, and if **`/usr/hosts`** is in your search path, the command:

```
remotehost
```

is equivalent to:

```
rlogin remotehost
```

RETURN VALUES

`rlogin` sends an error message to standard error and returns a nonzero value if an error occurs before the connection to the remote host is completed. Otherwise, it returns a zero.

DIAGNOSTICS

Diagnostics can occur from both the local and remote hosts. Those that occur on the local host before the connection is completely established are written to standard error. Once the connection is established, any error messages from the remote host are written to standard output, like any other data.

`Error! could not retrieve authentication type.`

`Please notify sys admin.`

There are two authentication mechanisms used by **`rlogin`**. One authentication mechanism is based on Kerberos and the other is not. The type of authentication mechanism is obtained from a system file which is updated by **`inetsvcs_sec`** (see *inetsvcs_sec*(1M)). If the system file does not contain known authentication types, the above error is displayed.

`login/tcp: Unknown service`

`rlogin` was unable to find the login service listed in the **`/etc/services`** database file.

`There is no entry for you (user ID` *username***`) in /etc/passwd`**

`rlogin` was unable to find your user ID in the password file.

Next Step: Contact your system administrator.

system call`:`*...*

An error occurred when **`rlogin`** attempted the indicated system call. See the appropriate manual entry for information about the error.

EXAMPLES

Log in as the same user on the remote host **`remote`**:

r

```
rlogin remote
```

Set the escape character to a **!**, use a seven-bit data connection, and attempt a login as user **guest** on host **remhost**:

```
rlogin remhost -e! -7 -l guest
```

Assuming that your system administrator has set up the links in **/usr/hosts**, the following is equivalent to the previous command:

```
remhost -e! -7 -l guest
```

WARNINGS

For security purposes, the **/etc/hosts.equiv** and **.rhosts** files should exist, even if they are empty. These files should be readable and writable only by the owner. See *host.equiv*(4) for more information.

Note also that all information, including any passwords asked for, is passed unencrypted between the two hosts.

rlogin is unable to transmit the Break key as an interrupt signal to the remote system, regardless of whether the user has set **stty brkint** on the local system. The key assigned to **SIGINT** with the command **stty intr** *c* should be used instead (see *stty*(1)).

AUTHOR

rlogin was developed by the University of California, Berkeley.

FILES

$HOME/.rhosts	User's private equivalence list
/etc/hosts.equiv	List of equivalent hosts
/usr/hosts/*	For *rhost* version of the command

SEE ALSO

csh(1), ksh(1), login(1), remsh(1), sh(1), sh-bourne(1), sh-posix(1), stty(1), telnet(1), rlogind(1M), inetsvcs_sec(1M), hosts(4), hosts.equiv(4), inetd.conf(4), services(4), termio(7), tty(7).

NAME

rlogin - remote login

SYNOPSIS

rlogin *rhost* [**-7**] [**-8**] [**-e***e*] [**-f/F**] [**-k** *realm*] [**-l** *username*] [**-P**]

rhost [**-7**] [**-8**] [**-e***e*] [**-f/F**] [**-k** *realm*] [**-l** *username*] [**-P**]

DESCRIPTION

The **rlogin** command connects your terminal on the local host to the remote host (*rhost*). **rlogin** acts as a virtual terminal to the remote system. The host name *rhost* can be either the official name or an alias as listed in the file **/etc/hosts** (see *hosts*(4)).

The terminal type specified by the current **TERM** environment variable is propagated across the network and used to set the initial value of your **TERM** environment variable on the remote host. Your terminal baud rate is also propagated to the remote host, and is required by some systems to set up the pseudo-terminal used by **rlogind** (see *rlogind*(1M)).

All echoing takes place at the remote site, so that (except for delays) the remote login is transparent.

If at any time **rlogin** is unable to read from or write to the socket connection on the remote host, the message **Connection closed** is printed on standard error and **rlogin** exits.

In a Kerberos V5 Network Authentication environment, **rlogin** uses the Kerberos V5 protocol to authenticate the connection to a remote host. If the authentication is successful, user authorization will be performed according to the command line options selected for **rlogind** (i.e., **-K**, **-R**, **-r**, or **-k**). A password will not be required, so a password prompt will not be seen and a password will not be sent over the network where it can be observed. For further information on Kerberos authentication and authorization see the Secure Internet Services man page, *sis*(5) and *rlogind*(1M).

Although Kerberos authentication and authorization may apply, the Kerberos mechanism is **not** applied to the login session. All information transferred between your host and the remote host is sent in cleartext over the network.

Options

rlogin recognizes the following options. Note that the options follow the *rhost* argument.

-7 Set the character size to seven bits. The eighth bit of each byte sent is set to zero (space parity).

-8 Use an eight-bit data path. This is the default HP-UX behavior.

To use eight-bit characters, the terminal must be configured to generate either eight-bit characters with no parity, or seven bit characters with space parity. The HP-UX implementation of **rlogind** (see *rlogind*(1M)) interprets seven bit characters with even, odd, or mark parity as eight-bit non-USASCII characters. You may also need to reconfigure the remote host appropriately (see *stty*(1) and *tty*(7)). Some remote hosts may not provide the necessary support for eight-bit characters. In this case, or if it is not possible to disable parity generation by the local terminal, use the **-7** option.

-e*e* Set the escape character to *e*. There is no space separating the option letter and the argument character. To start a line with the escape character, two of the escape characters must be entered. The default escape character is tilde (˜). Some characters may conflict with your terminal configuration, such as **^S**, **^Q**, or backspace. Using one of these as the escape character may not be possible or may cause problems communicating with the remote host (see *stty*(1) and *tty*(7)).

-f Forward the ticket granting ticket (TGT) to the remote system. The TGT is not forwardable from there.

-F Forward the TGT to the remote system and have it forwardable from there to another remote system. **-f** and **-F** are mutually exclusive.

-k *realm* Obtain tickets from the remote host in the specified *realm* instead of the remote host's default realm as specified in the configuration file *krb.realms*.

-l *username* Set the user login name on the remote host to *username*. The default name is the current account name of the user invoking **rlogin**.

-P Disable Kerberos authentication. Only applicable in a secure environment based on Kerberos V5. When this option is specified, a password is required and the password is sent across the network in cleartext. To bypass the normal login/password sequence, you can login to a remote host using an equivalent account in a manner similar to **remsh**. See *hosts.equiv*(4) for details.

Escape Sequences

rlogin can be controlled with two-character escape sequences, in the form *ex*, where *e* is the escape character and *x* is a code character described below. Escape sequences are recognized only at the beginning of a line of input. The default escape character is tilde (**~**). It can be changed with the **-e** option.

The following escape sequences are recognized:

ey If *y* is NOT a code character described below, pass the escape character and *y* as characters to the remote host.

ee Pass the escape character as a character to the remote host.

*e***.** Disconnect from the remote host.

*e***!** Escape to a subshell on the local host. Use **exit** to return to the remote host.

If **rlogin** is run from a shell that supports job control (see *csh*(1), *ksh*(1), and *sh-posix*(1)), escape sequences can be used to suspend **rlogin**. The following escape sequences assume that **^Z** and **^Y** are set as the user's **susp** and **dsusp** characters, respectively (see *stty*(1) and *termio*(7)).

*e***^Z** Suspend the **rlogin** session and return the user to the shell that invoked **rlogin**. The **rlogin** job can be resumed with the **fg** command (see *csh*(1), *ksh*(1), and *sh-posix*(1)). *e***^Z** suspends both **rlogin** processes: the one transmitting user input to the remote login, and the one displaying output from the remote login.

*e***^Y** Suspend the **rlogin** session and return the user to the shell that invoked **rlogin**. The **rlogin** job can be resumed with the **fg** command (see *csh*(1), *ksh*(1), and *sh-posix*(1)). *e***^Y** suspends only the input process; output from the remote login continues to be displayed.

If you "daisy-chain" remote logins (for example, you **rlogin** from host A to host B and then **rlogin** from host B to host C) without setting unique escape characters, you can repeat the escape character until it reaches your chosen destination. For example, the first escape character, *e*, is seen as an escape character on host A; the second *e* is passed as a normal character by host A and seen as an escape character on host B; a third *e* is passed as a normal character by hosts A and B and accepted as a normal character by host C.

Remote Host Name As Command

The system administrator can arrange for more convenient access to a remote host (*rhost*) by linking **remsh** to **/usr/hosts/***rhost*, allowing use of the remote host name (*rhost*) as a command (see *remsh*(1)). For example, if **remotehost** is the name of a remote host and **/usr/hosts/remotehost** is linked to **remsh**, and if **/usr/hosts** is in your search path, the command:

```
remotehost
```

is equivalent to:

```
rlogin remotehost
```

RETURN VALUES

rlogin sends an error message to standard error and returns a nonzero value if an error occurs before the connection to the remote host is completed. Otherwise, it returns a zero.

DIAGNOSTICS

Diagnostics can occur from both the local and remote hosts. Those that occur on the local host before the connection is completely established are written to standard error. Once the connection is established, any error messages from the remote host are written to standard output, like any other data.

Error! could not retrieve authentication type.

Please notify sys admin.

There are two authentication mechanisms used by **rlogin**. One authentication mechanism is based on Kerberos and the other is not. The type of authentication mechanism is obtained from a system file which is updated by **inetsvcs_sec** (see *inetsvcs_sec*(1M)). If the system file does not contain known authentication types, the above error is displayed.

`login/tcp: Unknown service`

`rlogin` was unable to find the login service listed in the **`/etc/services`** database file.

`There is no entry for you (user ID` *username***`) in /etc/passwd`**

`rlogin` was unable to find your user ID in the password file.

Next Step: Contact your system administrator.

system call:...

An error occurred when **`rlogin`** attempted the indicated system call. See the appropriate manual entry for information about the error.

`rcmd: connect <hostname>: Connection refused.`

One cause for display of this generic error message could be due to the absence of an entry for *login* in */etc/inetd.conf* on the remote system. This entry may have been removed or commented out to prevent non-secure access.

Kerberos-specific errors are listed in *sis*(5).

EXAMPLES

Log in as the same user on the remote host **`remote`**:

```
rlogin remote
```

Set the escape character to a **`!`**, use a seven-bit data connection, and attempt a login as user **`guest`** on host **`remhost`**:

```
rlogin remhost -e! -7 -l guest
```

Assuming that your system administrator has set up the links in **`/usr/hosts`**, the following is equivalent to the previous command:

```
remhost -e! -7 -l guest
```

WARNINGS

For security purposes, the **`/etc/hosts.equiv`** and **`.rhosts`** files should exist, even if they are empty. These files should be readable and writable only by the owner. See *host.equiv*(4) for more information.

Note also that all information, including passwords, is passed unencrypted between the two hosts. In a Kerberos V5 Network Authentication environment, a password is not transmitted across the network, so it will be protected.

`rlogin` is unable to transmit the Break key as an interrupt signal to the remote system, regardless of whether the user has set **`stty brkint`** on the local system. The key assigned to **`SIGINT`** with the command **`stty intr`** *c* should be used instead (see *stty*(1)).

AUTHOR

`rlogin` was developed by the University of California, Berkeley.

FILES

`$HOME/.rhosts`	User's private equivalence list
`/etc/hosts.equiv`	List of equivalent hosts
`/usr/hosts/*`	For *rhost* version of the command

SEE ALSO

csh(1), ksh(1), login(1), remsh(1), sh(1), sh-bourne(1), sh-posix(1), stty(1), telnet(1), rlogind(1M), inetsvcs_sec(1M), hosts(4), hosts.equiv(4), inetd.conf(4), services(4), termio(7), tty(7), sis(5).

NAME

rm - remove files or directories

SYNOPSIS

rm [**-f**|**-i**] [**-Rr**] *file* ...

DESCRIPTION

The **rm** command removes the entries for one or more files from a directory. If an entry was the last link to the file, the file is destroyed. Removal of a file requires write and search (execute) permission in its directory, but no permissions on the file itself. However, if the sticky bit is set on the directory containing the file, only the owner of the file, the owner of the directory, or a user having appropriate privileges can remove the file.

If a user does not have write permission for a file to be removed and standard input is a terminal, a prompt containing the file name and its permissions is printed requesting that the removal of the file be confirmed (see Access Control Lists below). A line is then read from standard input. If that line begins with **y** the file is deleted; otherwise, the file remains. No questions are asked when the **-f** option is given or if standard input is not a terminal.

If *file* is of type directory, and the **-f** option is not specified, and either the permissions of *file* do not permit writing and standard input is a terminal or the **-i** option is specified, **rm** writes a prompt to standard error and reads a line from standard input. If the response does not begin with **y**, it does nothing more with the current file and goes on to any remaining files.

If *file* is a symbolic link, then only the symbolic link is removed. The file or directory pointed to by the symbolic link is not affected. If any of the intermediate path components of *file* happens to be a symbolic link, then **rm** follows the symbolic link and removes the *file*.

Options

rm recognizes the following options:

-f Force each file or directory to be removed without prompting for confirmation, regardless of the permissions of the entry. This option also suppresses diagnostic messages regarding nonexistent operands.

This option does not suppress any diagnostic messages other than those regarding nonexistent operands. To suppress all error message and interactive prompts, the **-f** option should be used while redirecting standard error output to **/dev/null**.

This option ignores any previous occurrence of the **-i** option.

-i Write a prompt to standard error requesting confirmation before removing each entry.

This option ignores any previous occurrence of the **-f** option.

-R For each argument that is a directory, this option causes **rm** to recursively delete the entire contents of that directory before removing the directory itself. When used in conjunction with the **-i** option, **rm** asks whether to examine each directory before interactively removing files in that directory and again afterward to confirm removing the directory itself.

The **-R** option will descend to arbitrary depths in a file hierarchy and will not fail due to path length limitations unless the length of file name, **file** specified by the user exceeds system limitations.

-r Equivalent to **-R**.

Access Control Lists

If a file has optional ACL entries, **rm** displays a plus sign (**+**) after the file's permissions. The permissions shown summarize the file's **st_mode** value returned by **stat()** (see *stat*(2)). See also *acl*(5).

EXTERNAL INFLUENCES

Environment Variables

LANG provides a default value for the internationalization variables that are unset or null. If **LANG** is unset or null, the default value of "C" (see *lang*(5)) is used. If any of the internationalization variables contains an invalid setting, **rm** will behave as if all internationalization variables are set to "C". See *environ*(5).

LC_ALL If set to a non-empty string value, overrides the values of all the other internationalization variables.

LC_CTYPE determines the interpretation of file names as single and/or multi-byte characters, the classification of characters as printable, and the characters matched by character class expressions in regular expressions.

LC_MESSAGES determines the locale that should be used to affect the format and contents of diagnostic messages written to standard error and informative messages written to standard output.

NLSPATH determines the location of message catalogues for the processing of **LC_MESSAGES**.

International Code Set Support

Single- and multibyte character code sets are supported.

DIAGNOSTICS

Generally self-explanatory. Note that the **-f** option does not suppress all diagnostic messages.

It is forbidden to remove the file **..**, in order to avoid the consequences of using a command such as:

```
rm -r .*
```

If a designated file is a directory, an error comment is printed unless the **-R** or **-r** option is used.

RETURN VALUE

rm exits with one of the following values:

0 If the **-f** option is not specified, 0 is returned only if all the named directory entries (the arguments specified in the **rm** command) are removed.

If the **-f** option is specified, then all the existing named directory entries are removed. If any of the named directory entries are non-existent, **rm** still returns a zero.

>0 An error occurred.

EXAMPLES

Remove files with a prompt for verification:

```
rm -i file1 file2
```

Remove all the files in a directory:

```
rm -i mydirectory/*
```

Note that the previous command removes files only, and does not remove any directories in **mydirectory**.

r

Remove a file in the current directory whose name starts with **-** or ***** or some other character that is special to the shell:

```
rm ./-filename
rm \*filename
etc.
```

Remove a file in the current directory whose name starts with some strange (usually nonprinting, invisible) character or perhaps has spaces at the beginning or end of the filename, prompting for confirmation:

```
rm -i *filename*
```

If ***filename*** is not unique in the directory, enter **n** when each of the other files is prompted.

A powerful and dangerous command to remove a directory is:

```
rm -fR directoryname
```

or

```
rm -Rf directoryname
```

which removes all files and directories from **directoryname** without any prompting for verification to remove the files or the directories. This command should only be used when you are absolutely certain that all the files and directories in **directoryname** as well as **directoryname** itself are to be removed.

DEPENDENCIES

NFS

`rm` does not display a plus sign (`+`) to indicate the existence of optional access control list entries when asking for confirmation before removing a networked file.

SEE ALSO

rmdir(1), unlink(2), acl(5).

STANDARDS CONFORMANCE

`rm`: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

NAME

rmdel - remove a delta from an SCCS file

SYNOPSIS

rmdel -r *SID file* ...

DESCRIPTION

The **rmdel** command removes the delta specified by the *SID* from each named SCCS file. The delta to be removed must be the newest (most recent) delta in its branch in the delta chain of each named SCCS file. In addition, the SID specified must *not* be that of a version being edited for the purpose of making a delta (i.e., if a *p-file* (see *get*(1)) exists for the named SCCS file, the SID specified must *not* appear in any entry of the *p-file*).

If a directory is named, **rmdel** behaves as though each file in the directory were specified as a named file, except that non-SCCS files (last component of the path name does not begin with **s.**) and unreadable files are silently ignored. If a name of **-** is given, the standard input is read; each line of the standard input is taken to be the name of an SCCS file or directory to be processed; non-SCCS files and unreadable files are silently ignored. When **--** is specified on the command line, all following arguments are treated as file names.

The permissions to remove a delta are either (1) if you make a delta you can remove it; or (2) if you own the file and directory you can remove a delta.

EXTERNAL INFLUENCES

Environment Variables

LC_CTYPE determines the locale for the interpretation of text as single-byte and/or multi-byte characters.

LC_MESSAGES determines the language in which messages are displayed.

LC_MESSAGES also determines the local language equivalent of the affirmative string ("yes").

If **LC_CTYPE** or **LC_MESSAGES** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**.

If any internationalization variable contains an invalid setting, **rmdel** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single-byte and multi-byte character code sets are supported.

DIAGNOSTICS

Use *sccshelp*(1) for explanations.

FILES

x.file	See *delta*(1).
z.file	See *delta*(1).

SEE ALSO

delta(1), get(1), sccshelp(1), prs(1), sccsfile(4).

STANDARDS CONFORMANCE

rmdel: SVID2, SVID3, XPG2, XPG3, XPG4

NAME

rmdir - remove directories

SYNOPSIS

rmdir [**-f**|**-i**] [**-p**] *dir* ...

DESCRIPTION

rmdir removes the directory entry for each *dir* operand that refers to an empty directory.

Directories are removed in the order specified. Consequently, if a directory and a subdirectory of that directory are both specified as arguments, the subdirectory must be specified before the parent directory so that the parent directory will be empty when **rmdir** tries to remove it. Removal of a directory requires write and search (execute) permission in its parent directory, but no permissions on the directory itself; but if the sticky bit is set on the parent directory, only the owner of the directory, the owner of the parent directory, or a user having appropriate privileges can remove the directory.

Options

rmdir recognizes the following options:

-f Force each directory to be removed without prompting for confirmation, regardless of the presence of the **-i** option. This option also suppresses diagnostic messages regarding non-existent operands.

This option does not suppress any diagnostic messages other than those regarding non-existent operands. To suppress all error message and interactive prompts, the **-f** option should be used while redirecting the standard error output to **/dev/null**.

This option ignores any previous occurrence of the **-i** option.

-i Write a prompt to the standard error output requesting confirmation before removing each directory.

This option ignores any previous occurrence of the **-f** option.

-p Path removal. If, after removing a directory with more than one pathname component, the parent directory of that directory is now empty, **rmdir** removes the empty parent directory. This continues until **rmdir** encounters a non-empty parent directory, or until all components of the original pathname have been removed.

When used in conjunction with the **-i** option, **rmdir** asks whether to remove each directory component of a path.

EXTERNAL INFLUENCES

Environment Variables

LANG provides a default value for the internationalization variables that are unset or null. If **LANG** is unset or null, the default value of "C" (see *lang*(5)) is used. If any of the internationalization variables contains an invalid setting, **rmdir** will behave as if all internationalization variables are set to "C". See *environ*(5).

LC_ALL If set to a non-empty string value, overrides the values of all the other internationalization variables.

LC_CTYPE determines the interpretation of *dir* names as single and/or multi-byte characters, the classification of characters as printable, and the characters matched by character class expressions in regular expressions.

LC_MESSAGES determines the locale that should be used to affect the format and contents of diagnostic messages written to standard error and informative messages written to standard output.

NLSPATH determines the location of message catalogues for the processing of **LC_MESSAGES.**

International Code Set Support

Single- and multi-byte character code sets are supported.

DIAGNOSTICS

Generally self-explanatory. Note that the **-f** option does not suppress all diagnostic messages.

EXAMPLES

To remove directories with a prompt for verification:

`rmdir -i` *directories*

To remove as much as possible of a path, type:

`rmdir -p` *component1*/*component2*/*dir*

SEE ALSO

rm(1), rmdir(2), stat(2).

STANDARDS CONFORMANCE

rmdir: SVID2, XPG2, XPG3, XPG4

NAME

rmnl - remove extra new-line characters from file

SYNOPSIS

`rmnl`

DESCRIPTION

rmnl removes all blank lines from a file (except at beginning of file as explained below), and is useful for removing excess white space from files for display on a CRT terminal. Groups of two or more successive \n (new-line) characters are reduced to a single \n character, effectively eliminating all blank lines in the file *except* that one or more blank lines at the beginning of a file remain as a single blank line.

To remove redundant blank lines rather than all blank lines, use *ssp*(1).

To remove all blank lines from a file including beginning of file, use **rmnl** piped to **ssp**, or **ssp** piped to **rmnl**.

EXTERNAL INFLUENCES

International Code Set Support

Single- and multi-byte character code sets are supported.

SEE ALSO

man(1), ssp(1).

NAME

rpcgen - an RPC protocol compiler

SYNOPSIS

rpcgen *infile*

rpcgen [**-a**] [**-b**] [**-C**] [**-D** *name* [= *value*]] [**-i** *size*] [**-I** [**-K** *seconds*]]
[**-L**] [**-M**] [**-N**] [**-T**] [**-u**] [**-Y** *pathname*] *infile*

rpcgen [**-c** | **-h** | **-l** | **-m** | **-t** | **-Sc** | **-Ss** | **-Sm**] [**-o** *outfile*] [*infile*]

rpcgen [**-s** *nettype*] [**-u**] [**-o** *outfile*] [*infile*]

rpcgen [**-n** *netid*] [**-u**] [**-o** *outfile*] [*infile*]

DESCRIPTION

rpcgen is a tool that generates C code to implement an RPC protocol. The input to **rpcgen** is a language similar to C known as RPC Language (Remote Procedure Call Language).

rpcgen is normally used as in the first synopsis where it takes an input file and generates three output files. If the *infile* is named **proto.x**, then **rpcgen** generates a header in **proto.h**, XDR routines in **proto_xdr.c**, server-side stubs in **proto_svc.c**, and client-side stubs in **proto_clnt.c**. With the **-T** option, it also generates the RPC dispatch table in **proto_tbl.i**.

rpcgen can also generate sample client and server files that can be customized to suit a particular application. The **-Sc**, **-Ss** and **-Sm** options generate sample client, server and makefile, respectively. The **-a** option generates all files, including sample files. If the infile is **proto.x**, then the client side sample file is written to **proto_client.c**, the server side sample file to **proto_server.c** and the sample makefile to **makefile.proto**.

The server created can be started both by the port monitors (for example, **inetd** or **listen**) or by itself. When it is started by a port monitor, it creates servers only for the transport for which the file descriptor **0** was passed. The name of the transport must be specified by setting up the environment variable **PM_TRANSPORT**. When the server generated by **rpcgen** is executed, it creates server handles for all the transports specified in **NETPATH** environment variable, or if it is unset, it creates server handles for all the visible transports from **/etc/netconfig** file. Note: the transports are chosen at run time and not at compile time. When the server is self-started, it backgrounds itself by default. A special define symbol **RPC_SVC_FG** can be used to run the server process in foreground.

The second synopsis provides special features which allow for the creation of more sophisticated RPC servers. These features include support for user provided **#defines** and RPC dispatch tables. The entries in the RPC dispatch table contain:

- pointers to the service routine corresponding to that procedure,
- a pointer to the input and output arguments
- the size of these routines

A server can use the dispatch table to check authorization and then to execute the service routine; a client library may use it to deal with the details of storage management and XDR data conversion.

The other three synopses shown above are used when one does not want to generate all the output files, but only a particular one. See the *EXAMPLES* section below for examples of **rpcgen** usage. When **rpcgen** is executed with the **-s** option, it creates servers for that particular class of transports. When executed with the **-n** option, it creates a server for the transport specified by *netid*. If *infile* is not specified, **rpcgen** accepts the standard input.

The C preprocessor, **cc -E** is run on the input file before it is actually interpreted by **rpcgen**. For each type of output file, **rpcgen** defines a special preprocessor symbol for use by the **rpcgen** programmer:

RPC_HDR	defined when compiling into headers
RPC_XDR	defined when compiling into XDR routines
RPC_SVC	defined when compiling into server-side stubs
RPC_CLNT	defined when compiling into client-side stubs
RPC_TBL	defined when compiling into RPC dispatch tables

Any line beginning with “**%**” is passed directly into the output file, uninterpreted by **rpcgen**. To specify the path name of the C preprocessor use **-Y** flag.

For every data type referred to in *infile*, **rpcgen** assumes that there exists a routine with the string **xdr_** prepended to the name of the data type. If this routine does not exist in the RPC/XDR library, it must be provided. Providing an undefined data type allows customization of XDR routines.

Options

-a Generate all files, including sample files.

-b Backward compatibility mode. Generate transport specific RPC code for older versions of the operating system.

-c Compile into XDR routines.

-C Generate header and stub files which can be used with ANSI C compilers. Headers generated with this flag can also be used with C++ programs.

-D*name*[**=***value*]
Define a symbol *name*. Equivalent to the **#define** directive in the source. If no *value* is given, *value* is defined as **1**. This option may be specified more than once.

-h Compile into **C** data-definitions (a header). **-T** option can be used in conjunction to produce a header which supports RPC dispatch tables.

-i *size* Size at which to start generating inline code. This option is useful for optimization. The default size is 5.

-I Compile support for *inetd*(1M) in the server side stubs. Such servers can be self-started or can be started by **inetd**. When the server is self-started, it backgrounds itself by default. A special define symbol **RPC_SVC_FG** can be used to run the server process in foreground, or the user may simply compile without the **-I** option.

If there are no pending client requests, the **inetd** servers exit after 120 seconds (default). The default can be changed with the **-K** option. All of the error messages for **inetd** servers are always logged with *syslog*(3C).

Note: This option is supported for backward compatibility only. It should always be used in conjunction with the **-b** option which generates backward compatibility code. By default (i.e., when **-b** is not specified), **rpcgen** generates servers that can be invoked through portmonitors.

-K *seconds* By default, services created using **rpcgen** and invoked through port monitors wait **120** seconds after servicing a request before exiting. That interval can be changed using the **-K** flag. To create a server that exits immediately upon servicing a request, use **-K 0**. To create a server that never exits, the appropriate argument is **-K -1**.

When monitoring for a server, some portmonitors, like **listen**, *always* spawn a new process in response to a service request. If it is known that a server will be used with such a monitor, the server should exit immediately on completion. For such servers, **rpcgen** should be used with **-K 0**.

-l Compile into client-side stubs.

-L When the servers are started in foreground, use *syslog*(3C) to log the server errors instead of printing them on the standard error.

-m Compile into server-side stubs, but do not generate a main routine. This option is useful for doing callback-routines and for users who need to write their own main routine to do initialization.

-M Generate multithread-safe stubs for passing arguments and results between rpcgen generated code and user written code. This option is useful for users who want to use threads in their code.

-N This option allows procedures to have multiple arguments. It also uses the style of parameter passing that closely resembles C. So, when passing an argument to a remote procedure, you do not have to pass a pointer to the argument, but can pass the argument itself. This behavior is different from the old style of **rpcgen** generated code. To maintain backward compatibility, this option is not the default.

-n *netid* Compile into server-side stubs for the transport specified by *netid*. There should be an entry for *netid* in the **netconfig** database. This option may be specified more than once, so as to compile a server that serves multiple transports.

-o *outfile* Specify the name of the output file. If none is specified, standard output is used (**-c**, **-h**, **-l**, **-m**, **-n**, **-s**, **-Sc**, **-Sm**, **-Ss**, and **-t** modes only).

-s *nettype* Compile into server-side stubs for all the transports belonging to the class *nettype*. The supported classes are **netpath**, **visible**, **circuit_n**, **circuit_v**, **datagram_n**, **datagram_v**, **tcp**, and **udp** (see *rpc*(3N) for the meanings associated with these classes). This option may be specified more than once. Note: the transports are chosen at run time and not at compile time.

-Sc Generate sample client code that uses remote procedure calls.

-Sm Generate a sample Makefile which can be used for compiling the application.

-Ss Generate sample server code that uses remote procedure calls.

-t Compile into RPC dispatch table.

-T Generate the code to support RPC dispatch tables.

The options **-c**, **-h**, **-l**, **-m**, **-s**, **-Sc**, **-Sm**, **-Ss**, and **-t** are used exclusively to generate a particular type of file, while the options **-D** and **-T** are global and can be used with the other options.

-u When the server-side stub is produced, additional code to handle signals is generated. On reception of a signal, this signal handler code unmaps the server program from the port mapper before the server terminates. This code is added only if a **main()** routine is produced in the server-side stub. The **-u** option must not be specified with the **-c**, **-h**, **-l**, **-m**, **-Sc**, **-Sm**, **-Ss** options. The following signals are trapped: **SIGHUP**, **SIGINT**, **SIGQUIT**, and **SIGTERM**.

-Y *pathname* Give the name of the directory where **rpcgen** will start looking for the C-preprocessor.

EXAMPLES

The following example:

```
example% rpcgen -T prot.x
```

generates all the five files: **prot.h**, **prot_clnt.c**, **prot_svc.c**, **prot_xdr.c** and **prot_tbl.i**.

The following example sends the C data-definitions (header) to the standard output.

```
example% rpcgen -h prot.x
```

To send the test version of the **-DTEST**, server side stubs for all the transport belonging to the class **datagram_n** to standard output, use:

```
example% rpcgen -s datagram_n -DTEST prot.x
```

To create the server side stubs for the transport indicated by *netid* **tcp**, use:

```
example% rpcgen -n tcp -o prot_svc.c prot.x
```

AUTHOR

rpcgen was developed by Sun Microsystems, Inc.

SEE ALSO

cc(1), inetd(1M), syslog(3C), rpc(3N), rpc_svc_calls(3N).

NAME

rtprio - execute process with real-time priority

SYNOPSIS

rtprio *priority command* [*arguments*]

rtprio *priority* **-***pid*

rtprio -t *command* [*arguments*]

rtprio -t -*pid*

DESCRIPTION

rtprio executes *command* with a real-time priority, or changes the real-time priority of currently executing process *pid*. Real-time priorities range from zero (highest) to 127 (lowest). Real-time processes are not subject to priority degradation, and are all of greater (scheduling) importance than non-real-time processes. See *rtprio*(2) for more details.

If **-t** is specified instead of a real-time *priority*, **rtprio** executes *command* with a timeshare (non-real-time) priority, or changes the currently executing process *pid* from a possibly real-time priority to a timeshare priority. The former is useful to spawn a timeshare priority command from a real-time priority shell.

If **-t** is not specified, *command* is not scheduled, or *pid*'s real-time priority is not changed, if the user is not a member of a group having **PRIV_RTPRIO** access and is not the user with appropriate privileges. When changing the real-time priority of a currently executing process, the effective user ID of the calling process must be the user with appropriate privileges, or the real or effective user ID must match the real or saved user ID of the process to be modified.

RETURN VALUE

rtprio returns exit status 0 if *command* is successfully scheduled or if *pid*'s real-time priority is successfully changed, 1 if *command* is not executable or *pid* does not exist, and 2 if *command* (*pid*) lacks real-time capability, or the invoker's effective user ID is not a user who has appropriate privileges, or the real or effective user or the real or effective user ID does not match the real or saved user ID of the process being changed.

EXAMPLES

Execute file **a.out** at a real-time priority of 100:

```
rtprio 100 a.out
```

Set the currently running process pid 24217 to a real-time priority of 40:

```
rtprio 40 -24217
```

AUTHOR

rtprio was developed by HP.

SEE ALSO

setprivgrp(1M), getprivgrp(2), rtprio(2).

NAME

rtsched - execute process with real-time priority

SYNOPSIS

rtsched -s *scheduler* **-p** *priority command* [*arguments*]

rtsched [-s *scheduler* **] -p** *priority* **-P pid**

DESCRIPTION

Rtsched executes *command* with POSIX or HP-UX real-time priority, or changes the real-time priority of currently executing process *pid*.

All POSIX real-time priority processes are of greater scheduling importance than processes with HP-UX real-time or HP-UX timeshare priority. All HP-UX real-time priority processes are of greater scheduling importance than HP-UX timeshare priority processes, but are of lesser importance than POSIX real-time processes. Neither POSIX nor HP-UX real-time processes are subject to degradation. POSIX real-time processes may be scheduled with one of three different POSIX real-time schedulers: SCHED_FIFO, SCHED_RR, or SCHED_RR2. See *rtsched*(2) for details.

Rtsched is a superset of **rtprio.** See *rtprio*(1).

Options

-s *scheduler* Specify the desired scheduler:

POSIX real-time schedulers:	SCHED_FIFO
	SCHED_RR
	SCHED_RR2
HP-UX real-time scheduler:	SCHED_RTPRIO
HP-UX timeshare scheduler:	SCHED_HPUX

-p *priority* Specify priority range; any integer within the inclusive priority range of the corresponding scheduler. **-p** *priority* is required for all schedulers except SCHED_HPUX. If scheduler is SCHED_HPUX, the *priority* argument is ignored. The default *priority* range of each scheduler is as follows:

scheduler	highest priority	lowest priority
SCHED_FIFO	31	0
SCHED_RR	31	0
SCHED_RR2	31	0
SCHED_RTPRIO	0	127
SCHED_HPUX	N/A	N/A

Note: Higher numerical values for the *priority* represent higher priorities under POSIX real-time schedulers, whereas lower numerical values for the *priority* represent higher priorities under HP-UX real-time scheduler.

-P Specify an already executing process ID (*pid*).

Command is not scheduled, or *pid*'s real-time priority is not changed, if the user is not a member of a group having **PRIV_RTSCHED** access and is not the user with appropriate privileges. When changing the real-time priority of a currently executing process, the effective user ID of the calling process must be the user with appropriate privileges, or the real or effective user ID must match the real or saved user ID of the process to be modified.

RETURN VALUE

rtsched returns exit status:

0 if *command* is successfully scheduled or if *pid*'s real-time priority is successfully changed;

1 if *command* is not executable, *pid* does not exist, or *priority* is not within the priority range for the corresponding scheduler;

2 if *command* (*pid*) lacks real-time capability, or the invoker's effective user ID is not a user who has appropriate privileges, or the real or effective user or the real or effective user ID does not match the real or saved user ID of the process being changed; or

5 if rtsched encountered an internal error or if rtsched is not supported by this release.

EXAMPLES

Execute file **a.out** with SCHED_FIFO at a priority of 10:

```
rtsched -s SCHED_FIFO -p 10 a.out
```

Execute file **a.out** with SCHED_RTPRIO at a priority of 127 (this is synonymous to **rtprio 127 a.out**):

```
rtsched -s SCHED_RTPRIO -p 127 a.out
```

Execute file **a.out** with the SCHED_HPUX scheduler:

```
rtsched -s SCHED_HPUX a.out
```

This is useful to spawn a timeshare priority command from a real-time priority shell.

Set the currently running process pid 24217 to execute with SCHED_RR2 at a priority of 20:

```
rtsched -s SCHED_RR2 -p 20 -P 24217
```

Now change its priority to 10 using the same scheduler:

```
rtsched -p 10 -P 24217
```

WARNINGS

Priority values used by **rtsched** may differ from those used by other commands. For example, *ps*(1) displays the internal representation of priority values.

AUTHOR

rtsched was developed by HP.

SEE ALSO

rtprio(1), setprivgrp(1M), getprivgrp(2), rtprio(2), rtsched(2).

NAME

rup - show host status of local machines (RPC version)

SYNOPSIS

rup [**-h**] [**-l**] [**-t**] [*host* ...]

DESCRIPTION

rup gives a status similar to **uptime** for remote machines. It broadcasts on the local network and displays the responses it receives. Though the listing is normally in the order responses are received, the order can be changed by using command-line options. The broadcast process takes about two minutes.

When *host* arguments are given, instead of broadcasting, **rup** only queries the list of specified hosts.

A remote host only responds if it is running the **rstatd** daemon (see *rstatd*(1M)).

Options

rup recognizes the following command-line options:

-h	Sort the display alphabetically by host name.
-l	Sort the display by load average.
-t	Sort the display by up time.

WARNINGS

Broadcasting does not work through gateways; therefore, **rup** does not report information about machines that are reachable only through gateways.

DIAGNOSTICS

The exit code of **rup** is the number of errors (eg. bad host names) encountered; zero for success.

AUTHOR

rup was developed by Sun Microsystems, Inc.

FILES

/etc/inetd.conf

SEE ALSO

ruptime(1), inetd(1M), rstatd(1M), services(4).

NAME

ruptime - show status of local machines

SYNOPSIS

ruptime [**-a**] [**-r**] [**-l**] [**-t**] [**-u**]

DESCRIPTION

ruptime outputs a status line for each machine on the local network that is running the **rwho** daemon. **ruptime**'s status lines are formed from packets broadcast once every 3 minutes between **rwho** daemons (see *rwhod*(1M)) on each host on the network. Each status line has a field for the name of the machine, the status of the machine (up or down), how long the machine has been up or down, the number of users logged into the machine, and the 1-, 5- and 15-minute load averages for the machine when the packet was sent.

The status of the machine is reported as "up" unless no report has been received from the machine for 11 minutes or more.

The length of time that the machine has been up is shown as:

days+*hours*:*minutes*

Load averages are the average number of jobs in the run queue over the last 1-, 5- and 15-minute intervals when the packet was sent.

An example status line output by **ruptime** might be:

```
machine1  up  1+5:15,  7 users, load  1.47, 1.16, 0.80
```

The above status line would be interpreted as follows:

machine1 is presently "up" and has been up for 1 day, 5 hours and 15 minutes. It currently has 7 users logged in. Over the last 1-minute interval, an average of 1.47 jobs were in the run queue. Over the last 5-minute interval, an average of 1.16 jobs were in the run queue. Over the last 15-minute interval, an average of 0.80 jobs were in the run queue.

If a user has not used the system for an hour or more, the user is considered idle. Idle users are not shown unless the **-a** option is specified.

Options

If no options are specified, the listing is sorted by host name. Options change sorting order as follows:

-l	Sort by load average.
-t	Sort by up time.
-u	Sort by the number of users.
-r	Reverse the sort order.

DIAGNOSTICS

no hosts!?!

No status report files in **/var/spool/rwho**. *Next Step*: Ask the system administrator to check whether the **rwho** daemon is running.

AUTHOR

ruptime was developed by the University of California, Berkeley.

FILES

/var/spool/rwho/whod.*	Data files

SEE ALSO

rwho(1), rwhod(1M).

NAME

rusers - determine who is logged in on machines on local network

SYNOPSIS

rusers [**-a**] [**-h**] [**-i**] [**-l**] [**-u**] [*host* ...]

DESCRIPTION

rusers produces output similar to the "quick" option of *who*(1), but for remote machines. It broadcasts on the local network and prints the responses it receives. Though the listing is normally in the order that responses are received, the order can be changed by specifying a command-line option. The broadcast process takes about two minutes.

When *host* arguments are given, instead of broadcasting, **rusers** only queries the list of specified hosts.

For each machine, the default is to print a line listing the host name and all users on that host. When the **-l** option is given, **rusers** uses an output format similar to *rwho*(1). If a user has not typed on the system for a minute or more, the idle time is reported.

A remote host only responds if it is running the *rusersd*(1M) daemon.

Options

rusers recognizes the following command-line options:

-a	Give a report for a machine even if no users are logged in on it.
-h	Sort alphabetically by host name.
-i	Sort by idle time.
-l	Give a longer listing in the style of **who**-R (see *who*(1)).
-u	Sort by number of users.

RETURN VALUE

rusers returns exit code zero if no errors are encountered; otherwise it returns the number of errors found.

AUTHOR

rusers was developed by Sun Microsystems, Inc.

WARNINGS

Broadcasting does not work through gateways; therefore, **rusers** does not report information about machines that are reached only through gateways.

r

FILES

/etc/inetd.conf

SEE ALSO

rwho(1), who(1), inetd(1M), rusersd(1M).

NAME

rwho - show who is logged in on local machines

SYNOPSIS

rwho [**-a**]

DESCRIPTION

rwho produces output similar to the output of the HP-UX **who** command for all machines on the local network that are running the **rwho** daemon (see *who*(1) and *rwhod*(1M)). If **rwhod** has not received a report from a machine for 11 minutes, **rwho** assumes the machine is down and **rwho** does not report users last known to be logged into that machine.

rwho's output line has fields for the name of the user, the name of the machine, the user's terminal line, the time the user logged in, and the amount of time the user has been idle. Idle time is shown as:

hours **:** *minutes*

If a user has not typed to the system for a minute or more, **rwho** reports this as idle time. If a user has not typed to the system for an hour or more, the user is omitted from **rwho**'s output unless the **-a** flag is given.

An example output line from **rwho** would look similar to:

```
joe_user   machine1:tty0p1  Sep 12  13:28   :11
```

This output line could be interpreted as **joe_user** is logged into **machine1** and his terminal line is **tty0p1**. **joe_user** has been logged on since September 12 at 13:28 (1:28 p.m.). **joe_user** has not typed anything into **machine1** for 11 minutes.

WARNINGS

rwho's output becomes unwieldy when the number of users for each machine on the local network running **rwhod** becomes large. One line of output occurs for each user on each machine on the local network that is running **rwhod**.

AUTHOR

rwho was developed by the University of California, Berkeley.

FILES

/var/spool/rwho/whod.* Information about other machines.

SEE ALSO

ruptime(1), rusers(1), rwhod(1M).

NAME

sact - print current SCCS file editing activity

SYNOPSIS

sact *file* ...

DESCRIPTION

The **sact** command informs the user of any impending deltas to a named SCCS file. This situation occurs when **get -e** has been previously executed without a subsequent execution of **delta** (see *delta*(1) and *get*(1)). If a directory is named on the command line, **sact** behaves as though each file in the directory were specified as a named file, except that non-SCCS files (last component of path name does not begin with **s.**) and unreadable files are silently ignored. If a name of **-** is given, the standard input is read with each line being taken as the name of an SCCS file to be processed.

The output for each named file consists of five fields separated by spaces.

Field 1	SID of a delta that currently exists in the SCCS file to which changes will be made to make the new delta.
Field 2	SID for the new delta to be created.
Field 3	Logname of the user making the delta (i.e., executed a **get** for editing).
Field 4	Date when **get -e** was executed.
Field 5	Time when **get -e** was executed.

DIAGNOSTICS

Use *sccshelp*(1) for explanations.

SEE ALSO

delta(1), get(1), sccshelp(1), unget(1).

STANDARDS CONFORMANCE

sact: SVID2, SVID3, XPG2, XPG3, XPG4

NAME

samlog_viewer - a tool for viewing and saving the SAM logfile

SYNOPSIS

/usr/sam/bin/samlog_viewer [**-s** *MMDDhhmm*[*YY*]] [**-e** *MMDDhhmm*[*YY*]] [**-l SDVC**] [**-u** *user*] [**-o** *ofile*] [**-t**] [**-n**] [*file*]

DESCRIPTION

The **samlog_viewer** command enables the viewing of part or all of the SAM logfile (or another file containing data in the same format) at varying levels of detail. This tool is run by SAM whenever the **View SAM Log** option is chosen. It can also be run independently of SAM, in either interactive or noninteractive mode.

The **samlog_viewer** command executes in either interactive or noninteractive mode, depending on the options given. In noninteractive mode, **samlog_viewer** filters the source file and writes the resulting data either to stdout or to a destination file. In interactive mode, **samlog_viewer** displays a graphical user interface enabling you to try different combinations of filtering, save one or more versions of the source file to other files, scroll back and forth among the logfile entries, etc.

Under no circumstances is **samlog_viewer** destructive to the contents of the SAM logfile (or whatever source file is specified by *file*). The contents of the source file are filtered and displayed (or output) according to the settings of the available filters. Multiple instances of **samlog_viewer** can be run simultaneously without harmful effects.

Filters

samlog_viewer supports three types of filters: level of detail, date/time, and user filters. These filters can be used in combination to provide highly selective logfile viewing.

The level of detail filters control how much detail is displayed. The SAM logfile may contain entries of many different types. The entry types currently supported are: summary, detail, verbose, error, and note. The level of detail filters display some or all of these entry types, depending on which filter is chosen. The level of detail filters are:

Summary	Displays only the higher level messages. These include *summary*, *error*, and *note* (warnings, other entries worthy of special attention) entry types.
Detail	Includes **Summary** level of detail, and adds *detail* log entries. If no level of detail is specified, this is the default.
Verbose	Includes **Detail** level of detail, and adds *verbose* log entries.
Commands Only	Displays only the literal commands that were executed. These commands may include HP-UX commands as well as SAM commands and scripts.

The date/time filters are used to ask for entries written since a specific date/time, before a specific date/time, or both.

The user filters are for viewing only those entries written by a particular user. If invoked with the **-u** option, its argument is used as the user whose entries should be shown. If **-u** is not specified, then the value of the environment variable **SAM_RESTRICTED_USER** is used as the name of the user. If neither **-u** or **SAM_RESTRICTED_USER** is present, then no user filtering is done and entries from all users are shown. If both **-u** and **SAM_RESTRICTED_USER** are in effect, the **-u** option overrides the setting of the environment variable.

If user filtering is selected, either by the **SAM_RESTRICTED_USER** environment variable or by the **-u** option, **samlog_viewer** displays only entries made by that user and disallows any changes to the user filtering. This is useful for allowing nonprivileged users to run **samlog_viewer** and see only those entries that pertain to them. Otherwise, **samlog_viewer** allows the user filtering to be changed, or completely disabled, from its interactive filtering screen.

Options

The following options enable you to set up filtering and other attributes. If **samlog_viewer** runs interactively, these attributes may also be set and modified in the various supported menus and displays. The available options are:

-s *MMDDhhmm*[*YY*] The **-s** option sets the start date/time filter to the date/time given by its argument. The date/time is specified in the same way as it is for the **date** command (see *date*(1)):

S

MM	Month specified as a two digit number (e.g., **08**).
DD	Day specified as a two digit number.
hh	Hour specified as a two digit number (24-hour clock form).
mm	Minute specified as a two digit number.
YY	The last two digits of the desired year. If this is not specified, the current year is used.

If no start time is given, the beginning of the log is used as the start time.

-e *MMDDhhmm*[*YY*]
: The **-e** option sets the end date/time filter to the date/time given by its argument. The date and time is specified as described above for the **-s** option. If no end time is given, then an end date/time of infinity (no end time) is used.

-l SDVC
: The **-l** option sets the desired level of detail. One of the letters **SDVC** must be specified as the required argument. The level of detail is set as follows:

S = **Summary**
D = **Detail**
V = **Verbose**
C = **Commands Only**

If the **-l** option is not specified, the **Detail** level of detail is used by default.

-u *user*
: The **-u** option sets the user filter to the user name or user ID specified by *user*. Only entries logged by this user are displayed. If the **-u** option is omitted, entries logged by all users are displayed by default.

-o *ofile*
: The **-o** option causes the filtered output to be written to the output file *ofile*. The **-o** option implies the **-n** option described below. If *ofile* is **-**, the output is written to stdout. If **-o** is omitted, the output is written to either stdout (if **-n** is specified) or to the interactive **samlog_viewer** display (if **-n** is omitted).

-t
: The **-t** option enables automatic timestamping. If specified, each log entry is tagged with the time of day at which it was written. Timestamping is disabled by default.

-n
: The **-n** option forces noninteractive behavior. If specified, **samlog_viewer** runs noninteractively, using the default or specified values for all supported options and source/destination files.

file
: Specifies the name of the file from which log data is read. The format of the data in the specified file must be the same as that used for raw SAM logfile data. If omitted, the SAM logfile is read. If *file* is **-**, stdin is read and **samlog_viewer** runs noninteractively. If given, *file* must be the last argument specified on the command line.

S

EXAMPLES

Capture the current contents of the SAM logfile using default filtering, and put into the file **sam.out**:

```
samlog_viewer -n >sam.out
```

The following example does the same thing:

```
samlog_viewer -o sam.out
```

View only the commands executed by SAM on behalf of user **tom**, between 8am June 5, 1994 and 10pm August 14, 1994, and view the data interactively:

```
samlog_viewer -s 0605080094 -e 0814220094 -lC -u tom
```

Noninteractively read data from stdin, timestamp it, and save the result in a file called **stdin.out**:

```
cat datafile | samlog_viewer -t -o stdin.out -
```

Do the same as above, but instead have the data appear on stdout:

```
cat datafile | samlog_viewer -t -o - -
```

or

```
cat datafile | samlog_viewer -tn -
```

FILES

/var/sam/log/samlog	SAM logfile.
/var/sam/log/samlog.old	Archived version of **samlog**, created when the logfile is automatically trimmed by SAM when its size becomes too large. Its contents are included in the log entries read by **samlog_viewer**.
/tmp/LFV_*<pid>*	Temporary files used by **samlog_viewer**.
/tmp/LFV_RUN*<pid>*	

S

NAME

sccs - front-end utility program for SCCS commands

SYNOPSIS

sccs [**-r**] [**-d** *rootpath*] [**-p** *dirpath*] *command* [*options*] [*file* ...]

DESCRIPTION

The **sccs** command is a straightforward front end to the various programs comprising the Source Code Control System. It includes the capability of running set-user-id to another user to allow shared access to the SCCS files. **sccs** reduces the need to explicitly reference the SCCS filenames. The SCCS filenames are generated by prepending the string **SCCS/s.** to the working *files* specified. The default SCCS subdirectory name can be overridden with the **-p** *dirpath* option.

The *command* supplied to the **sccs** command can either be an SCCS program or a pseudo command. The SCCS programs that **sscs** handles include **admin**, **cdc**, **comb**, **delta**, **get**, **help**, **prs**, **rmdel**, **sact**, **unget**, **val**, **what** and **sccsdiff**. The pseudo commands are:

check Prints a list of all files being edited. Returns a non-zero exit status if a file is being edited. The intent is to allow an 'install' entry in a makefile to verify that everything is included in the SCCS file before a version is installed. See the **info** pseudo command for a description of the **-b**, **-u** *user* and **-U** options.

clean Removes all files from the current directory or the named directory that can be recreated from the SCCS files. Does not remove files that are in the process of being edited. If **-b** is given, branches (i.e. SID's with three or more components) are ignored in determining which files are being edited. Therefore, any edits on branches can be lost.

create Creates the initial SCCS file, taking the contents from *file*. Any options to **admin** are accepted. If the files are created successfully, the original files are renamed with a **,** (comma) on the front. Read-only copies are retrieved with **get**. The renamed files should be removed after you have verified that the SCCS files have been created successfully.

delget Runs **delta** on the named files and then **get** the new versions. The new versions of the files have expanded identification keywords, and cannot be edited. The [**-mprsy**] options are passed to **delta**, and the [**-bceiklsx**] options are passed to **get**.

deledit Equivalent to **delget**, except that the **get** phase includes the **-e** option.

diffs Gives a **diff** listing between the current version of the files being edited and the versions in SCCS format. The [**-rcixt**] options are passed to **get**. The [**-lsefhb**] options are passed to **diff**. The **-C** option is passed to **diff** as **-c**.

edit Equivalent to **get -e**.

enter Equivalent to **create**, except **get** is omitted. This pseudo command is useful when you want to run the **edit** command immediately after creating the SCCS file.

fix Removes a named delta, but leaves a copy of the delta in the current directory. The **-r** *SID* option is required and must point to a leaf in the source tree. Since a record of the changes is not preserved, **fix** should be used carefully.

info Lists all the files being edited. The **-b** option ignores branches in determining which files are being edited. The **-u** *user* option lists only the files being edited by *user*. The **-U** option is equivalent to **-u** *current_user*.

print Prints information about named files. Equivalent to **prs -a** followed by **get -p -m -s**.

tell Lists all the files being edited, with a newline after each entry. See the **info** section for a description of the **-b**, **-u** *user* and **-U** options.

unedit Equivalent to **unget**. Any changes made since the last **get** are lost. Use with caution.

Certain commands, **admin**, **cdc**, **check**, **clean**, **diffs**, **info**, **rmdel**, **sccsdiff**, and **tell** cannot use the set-user-id feature, as this would allow anyone to change the authorizations. These commands are always run as the real user.

Options

The *options* supplied to the SCCS commands are documented in the corresponding SCCS man pages. The *options* supplied to the pseudo commands are documented in the above section. All other options preceding *command* are documented as follows:

-r	Runs **sccs** as the real user rather than the effective user **sccs** is set-user-id to.
-d *rootpath*	Gives the pathname to be used as the root directory for the SCCS files. *rootpath* defaults to the current directory. This flag takes precedence over the PROJECTDIR environment variable.
-p *dirpath*	Specifies the pathname for the SCCS files. The default is the SCCS directory. *dirpath* is appended to *rootpath* and is inserted before the final component of the pathname.

The command **sccs -d /usr -p cmd get src/b** converts to **get**/usr/src/cmd/s.b**.** This can be used to create aliases. For example, the command **alias syssccs="sccs -p /usr/src/cmd"** makes **syssccs** an alias that can be used in commands like **syssccs**get**b**.

EXTERNAL INFLUENCES

Environment Variables

If the **PROJECTDIR** environment variable is set, its value is used to determine the **-d** *rootpath* option value for *rootpath*. If **PROJECTDIR** begins with a / (slash), the value is used directly; otherwise, the value is assume to be a login name and the home directory corresponding to login name is examined for a subdirectory named **src** or **source**. If found, this directory path is used. Otherwise, the value is used as a relative path name.

LC_CTYPE determines the interpretation of text within file as single- and/or multi-byte characters.

LC_MESSAGES determines the language in which messages are displayed.

If **LC_CTYPE** or **LC_MESSAGES** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**.

If any internationalization variable contains an invalid setting, **sccs** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single-byte and multi-byte character code sets are supported.

EXAMPLES

To create a new SCCS file:

```
sccs create file
```

To get a file for editing, edit it, create a new delta and get file for editing:

```
sccs edit file.c
ex file.c
sccs deledit file.c
```

To get a file from another directory (**/usr/src/cmd/SCCS/s.cc.c**):

```
sccs -d /usr/src get cmd/cc.c
```

To make a delta of a large number of files in the current directory, enter:

```
sccs delta *.c
```

To get a list of files being edited that are not on branches, enter:

```
sccs info -b
```

To get a list of files being edited by you, enter:

```
sccs tell -u
```

In a makefile, to get source files from an SCCS file if it does not already exist, enter:

```
SRCS = <list of source files>
$ (SRCS) :
     sccs get $(REL) $@
```

S

RETURN VALUE

A successful completion returns 0. On error, **sccs** exists with a value from <sysexits.h> or the exit value from the *command* that was invoked. The only exception is the **check** pseudo command which returns a non-zero exit status if a file is being edited.

SEE ALSO

admin(1), cdc(1), comb(1), delta(1), get(1), prs(1), rmdel(1), sact(1), sccsdiff(1), sccshelp(1), unget(1), val(1), vc(1), what(1), sccsfile(4).

SCCS: Source Code Control System chapter in *HP-UX Linker and Libraries User's Guide*.

STANDARDS CONFORMANCE

sccs: XPG4

NAME

sccsdiff - compare two versions of an SCCS file

SYNOPSIS

sccsdiff -r*SID1* **-r***SID2* [**-p**] [**-s***n*] *file* ...

DESCRIPTION

The **sccsdiff** command compares two versions of an SCCS file, and generates the differences between the two versions. Any number of SCCS files may be specified, but arguments apply to all files.

-r*SID?*	*SID1* and *SID2* specify the deltas of an SCCS file that are to be compared. Versions are passed to **bdiff** in the order given (see *bdiff*(1)). The SIDs accepted, and the corresponding version retrieved for the comparison are the same as for **get** (see *get*(1)).
-p	Pipe output for each file through **pr** (see *pr*(1)).
-s*n*	*n* is the file segment size that **bdiff** passes to **diff** (see *diff*(1)). This is useful when **diff** fails due to a high system load.

EXTERNAL INFLUENCES

International Code Set Support

Single- and multi-byte character code sets are supported with the exception that multi-byte-character file names are not supported.

DIAGNOSTICS

file: **No differences**

The two versions are identical.

Use *sccshelp*(1) for explanations.

FILES

/tmp/get?????	Temporary files

SEE ALSO

bdiff(1), diff(1), get(1), pr(1), sccshelp(1).

S

NAME

sccshelp - ask for help on SCCS commands

SYNOPSIS

sccshelp [*arg*] ...

DESCRIPTION

The **sccshelp** command finds information to explain a message from an SCCS command or to explain the use of a SCCS command. Zero or more arguments can be supplied. If no arguments are given, **sccshelp** prompts for one:

```
What is the message number or SCCS command name?
```

The arguments can be either message numbers (which normally appear in parentheses following messages) or command names, of one of the following types:

type 1	Begins with nonnumerics, ends in numerics. The nonnumeric prefix is usually an abbreviation for the program or set of routines which produced the message (e.g., **ge6**, for message 6 from the **get** command).
type 2	Does not contain numerics (as a command, such as **get**).
type 3	Is all numeric (e.g., **212**).

The response of the program is the explanatory information related to the argument, if there is any.

You can use **sccshelp** to support other commands by means of the **helploc** file. To do this, create help files in the appropriate format and add the location of the helpfiles to **/usr/share/lib/sccshelp/helploc**.

EXTERNAL INFLUENCES

Environment Variables

LC_CTYPE determines the interpretation of text as single- and/or multibyte characters.

LANG determines the language in which messages are displayed.

If **LC_CTYPE** is not specified or is null, it defaults to the value of **LANG**. If **LANG** is not specified or is null, it defaults to "C" (see *lang*(5)). If any internationalization variable contains an invalid setting, all internationalization variables default to "C". See *environ*(5).

International Code Set Support

Single- and multibyte character code sets are supported.

DIAGNOSTICS

There is not a place to look for help on the subject of *arg*

There is not any help available on the subject of *arg*

When all else fails, try **sscshelp stuck**.

EXAMPLES

If you enter the SCCS **get** command without parameters, you would get the message:

```
ERROR: missing file arg (cm3)
```

If you request help for the command:

```
sccshelp get
```

it displays:

```
get:
     get [-r<SID>] [-c<cutoff>] [-i<list>] [-x<list>] [-a<seq no.>]
         [-k] [-e] [-l<p>] [-p] [-m] [-n] [-s] [-b] [-g] [-t]
         [-w<string>] file ...
```

If you request help for the error number:

```
sccshelp cm3
```

it displays:

S

```
cm3:
"missing file arg"
You left off the name of the file to be processed.
```

WARNINGS

Only SCCS commands currently use **sccshelp**.

FILES

/usr/share/lib/sccshelp	Directory containing files of message text
/usr/share/lib/sccshelp/cmds	List of commands supported by sccshelp
/usr/share/lib/sccshelp/helploc	File containing the locations of help files that are not in the **/usr/share/lib/sccshelp** directory

NAME

script - make typescript of terminal session

SYNOPSIS

script [**-a**] [*file*]

DESCRIPTION

script makes a typescript of everything printed on your terminal. It starts a shell named by the **SHELL** environment variable, or by default **/usr/bin/sh**, and silently records a copy of output to your terminal from that shell or its descendents, using a pseudo-terminal device (see *pty*(7)).

All output is written to *file*, or appended to *file* if the **-a** option is given. If no file name is given, the output is saved in a file named **typescript**. The recording can be sent to a line printer later with *lp*(1), or reviewed safely with the **-v** option of *cat*(1).

The recording ends when the forked shell exits (or the user ends the session by typing "exit") or the shell and all its descendents close the pseudo-terminal device.

This program is useful when operating a CRT display and a hard-copy record of the dialog is desired. It can also be used for a simple form of session auditing.

script respects the convention for login shells as described in *su*(1), *sh*(1), and *ksh*(1). Thus, if it is invoked with a command name beginning with a hyphen (**-**) (that is, **-script**), **script** passes a basename to the shell that is also preceded by a hyphen.

EXAMPLES

Save everything printed on the user's screen into file **scott**:

```
script scott
```

Append a copy of everything printed to the user's screen to file **temp**:

```
script -a temp
```

WARNINGS

A command such as **cat scott**, which displays the contents of the destination file, should not be issued while executing **script** because it would cause **script** to log the output of the **cat** command to itself until all available disk space is filled. Other commands, such as *more*(1), can cause the same problem but to a lesser degree.

script records all received output in the *file*, including typing errors, backspaces, and cursor motions. Note that it does not record typed characters; only echoed characters. Thus passwords are not recorded in the *file*. Responses other than simple echoes (such as output from screen-oriented editors and **ksh** command editing) are recorded as they appeared in the original session.

AUTHOR

script was developed by the University of California, Berkeley and HP.

NAME

sdiff - side-by-side difference program

SYNOPSIS

sdiff [*options* ...] *file1 file2*

DESCRIPTION

sdiff uses the output of *diff*(1) to produce a side-by-side listing of two files, indicating those lines that are different. Each line of the two files is printed with a blank gutter between them if the lines are identical, a **<** in the gutter if the line only exists in *file1*, a **>** in the gutter if the line only exists in *file2*, and a **|** for lines that are different.

For example:

```
abc    |    xyz
abc         abc
bca    <
cba    <
dcb         dcb
       >    cde
```

Options

sdiff recognizes the following options:

-w *n* Use the next argument, *n*, as the width of the output line. The default line length is 130 characters.

-l Only print on the left side when lines are identical.

-s Do not print identical lines.

-o *output* Use the next argument, *output*, as the name of a third file that is created as a user-controlled merging of *file1* and *file2*. Identical lines of *file1* and *file2* are copied to *output*. Sets of differences, as produced by *diff*(1), are printed; where a set of differences share a common gutter character. After printing each set of differences, **sdiff** prompts the user with a **%** and waits for one of the following user-typed commands:

l	append the left column to the output file
r	append the right column to the output file
s	turn on silent mode; do not print identical lines
v	turn off silent mode
e l	call the editor with the left column
e r	call the editor with the right column
e b	call the editor with the concatenation of left and right
e	call the editor with a zero length file
q	exit from the program

On exit from the editor, the resulting file is concatenated on the end of the *output* file.

EXAMPLES

Print a side-by-side diff of two versions of a file on a printer capable of printing 132 columns:

```
sdiff -w132 prog.c.old prog.c | lp -dlineprinter
```

Retrieve the most recently checked in version of a file from RCS and compare it with the version currently checked out:

```
co -p prog.c > /tmp/$$; sdiff /tmp/$$ prog.c | more; rm /tmp/$$
```

SEE ALSO

diff(1), ed(1).

S

NAME

sed - stream text editor

SYNOPSIS

sed [**-n**] *script* [*file* ...]

sed [**-n**] [**-e** *script*] ... [**-f** *script_file*] ... [*file* ...]

DESCRIPTION

sed copies the named text *file*s (standard input default) to the standard output, edited according to a script containing up to 100 commands. Only complete input lines are processed. Any input text at the end of a file that is not terminated by a new-line character is ignored.

Options

sed recognizes the following options:

-f *script_file*	Take script from file *script_file*.
-e *script*	Edit according to *script*. If there is just one **-e** option and no **-f** options, the flag **-e** can be omitted.
-n	Suppress the default output.

sed interprets all **-e***script* and **-f***script_file* arguments in the order given. Use caution, if mixing **-e** and **-f** options, to avoid unpredictable or incorrect results.

Command Scripts

A script consists of editor commands, one per line, of the following form:

[*address* [**,** *address*]] *function* [*arguments*]

In normal operation, **sed** cyclically copies a line of input into a *pattern space* (unless there is something left after a **D** command), applies in sequence all commands whose *addresses* select that pattern space, and, at the end of the script, copies the pattern space to the standard output (except under **-n**) and deletes the pattern space.

Some of the commands use a *hold space* to save all or part of the *pattern space* for subsequent retrieval.

Command Addresses

An *address* is either a decimal number that counts input lines cumulatively across files, a **$** which addresses the last line of input, or a context address; that is, a **/***regular expression***/** in the style of *ed*(1) modified thus:

- In a context address, the construction **\***?regular expression?*, where *?* is any character, is identical to **/***regular expression***/**. Note that in the context address **\xabc\xdefx**, the second **x** stands for itself, so that the regular expression is **abcxdef**.
- The escape sequence **\n** matches a new-line character embedded in the pattern space.
- A period (**.**) matches any character except the terminal new-line of the pattern space.
- A command line with no addresses selects every pattern space.
- A command line with one address selects each pattern space that matches the address.
- A command line with two addresses selects the inclusive range from the first pattern space that matches the first address through the next pattern space that matches the second (if the second address is a number less than or equal to the line number first selected, only one line is selected). Thereafter the process is repeated, looking again for the first address.

sed supports Basic Regular Expression syntax (see *regexp*(5)).

Editing commands can also be applied to only non-selected pattern spaces by use of the negation function **!** (described below).

Command Functions

In the following list of functions, the maximum number of permissible addresses for each function is indicated in parentheses. Other function elements are interpreted as follows:

text One or more lines, all but the last of which end with \ to hide the new-line. Backslashes in *text* are treated like backslashes in the replacement string of an **s** command, and can be used to protect initial blanks and tabs against the stripping that is done on every script line.

rfile Must terminate the command line, and must be preceded by exactly one blank.

wfile Must terminate the command line, and must be preceded by exactly one blank. Each *wfile* is created before processing begins. There can be at most 10 distinct *wfile* arguments.

sed recognizes the following functions:

(1)**a**
text Append. Place *text* on the output before reading next input line.

(2)**b** *label* Branch to the **:** command bearing *label*. If no *label* is specified, branch to the end of the script.

(2)**c**
text Change. Delete the pattern space. With 0 or 1 address or at the end of a 2-address range, place *text* on the output. Start the next cycle.

(2)**d** Delete pattern space and start the next cycle.

(2)**D** Delete initial segment of pattern space through first new-line and start the next cycle.

(2)**g** Replace contents of the pattern space with contents of the hold space.

(2)**G** Append contents of hold space to the pattern space.

(2)**h** Replace contents of the hold space with contents of the pattern space.

(2)**H** Append the contents of the pattern space to the hold space.

(1)**i**
text Insert. Place *text* on the standard output.

(2)**l** List the pattern space on the standard output in an unambiguous form. Non-printing characters are spelled in three-digit octal number format (with a preceding backslash), and long lines are folded.

(2)**n** Copy the pattern space to the standard output if the default output has not been suppressed (by the **-n** option on the command line or the **#n** command in the *script* file). Replace the pattern space with the next line of input.

(2)**N** Append the next line of input to the pattern space with an embedded new-line. (The current line number changes.)

(2)**p** Print. Copy the pattern space to the standard output.

(2)**P** Copy the initial segment of the pattern space through the first new-line to the standard output.

(1)**q** Quit. Branch to the end of the script. Do not start a new cycle.

(1)**r** *rfile* Read contents of *rfile* and place on output before reading the next input line.

(2)**s/***regular expression***/***replacement***/***flags*
Substitute *replacement* string for instances of *regular expression* in the pattern space. Any character can be used instead of **/**. For a fuller description see *ed*(1). *flags* is zero or more of:

n *n* = 1-2048 (**LINE_MAX**). Substitute for just the *n*th occurrence of *regular expression* in the pattern space.

g Global. Substitute for all non-overlapping instances of *regular expression* rather than just the first one.

p Print the pattern space if a replacement was made and the default output has been suppressed (by the **-n** option on the command line or the **#n** command in the *script* file).

w *wfile* Write. Append the pattern space to *wfile* if a replacement was made.

(2)**t** *label* Test. Branch to the **:** command bearing the *label* if any substitutions have been made since the most recent reading of an input line or execution of a **t**. If *label* is empty, branch to the end of the script.

S

(2)**w** *wfile* Write. Append the pattern space to *wfile*.

(2)**x** Exchange the contents of the pattern and hold spaces.

(2)**y/***string1***/***string2***/**
Transform. Replace all occurrences of characters in *string1* with the corresponding character in *string2*. The lengths of *string1* and *string2* must be equal.

(2)**!** *function*
Don't. Apply the *function* (or group, if *function* is **{**) only to lines *not* selected by the address or addresses.

(0)**:** *label* This command does nothing; it bears a *label* for **b** and **t** commands to branch to.

(1)**=** Place the current line number on the standard output as a line.

(2)**{** Execute the following commands through a matching **}** only when the pattern space is selected. The syntax is:

```
{ cmd1
cmd2
cmd3
  .
  .
  .
}
```

(0) An empty command is ignored.

(0)**#** If a **#** appears as the first character on the first line of a script file, that entire line is treated as a comment with one exception: If the character after the **#** is an **n**, the default output is suppressed. The rest of the line after **#n** is also ignored. A script file must contain at least one non-comment line.

EXTERNAL INFLUENCES

Environment Variables

LANG provides a default value for the internationalization variables that are unset or null. If **LANG** is unset or null, the default value of "C" (see *lang*(5)) is used. If any of the internationalization variables contains an invalid setting, **sed** will behave as if all internationalization variables are set to "C". See *environ*(5).

LC_ALL If set to a non-empty string value, overrides the values of all the other internationalization variables.

LC_CTYPE determines the interpretation of text as single and/or multi-byte characters, the classification of characters as printable, and the characters matched by character class expressions in regular expressions.

S

LC_MESSAGES determines the locale that should be used to affect the format and contents of diagnostic messages written to standard error and informative messages written to standard output.

NLSPATH determines the location of message catalogues for the processing of **LC_MESSAGES.**

International Code Set Support

Single- and multi-byte character code sets are supported.

EXAMPLES

Make a simple substitution in a file from the command line or from a shell script, changing **abc** to **xyz**:

```
sed 's/abc/xyz/' file1 >file1.out
```

Same as above but use shell or environment variables **var1** and **var2** in search and replacement strings:

```
sed "s/$var1/$var2/" file1 >file1.out
```

or

```
sed 's/'$var1'/'$var2'/' file1 >file1.out
```

Multiple substitutions in a single command:

```
sed -e 's/abc/xyz/' -e 's/lmn/rst/' file1 >file1.out
```

or

```
sed -e 's/abc/xyz/' \
-e 's/lmn/rst/' \
file1 >file1.out
```

WARNINGS

sed limits command scripts to a total of not more than 100 commands.

The hold space is limited to 8192 characters.

sed processes only text files. See the glossary for a definition of text files and their limitations.

AUTHOR

sed was developed by OSF and HP.

SEE ALSO

awk(1), ed(1), grep(1), environ(5), lang(5), regexp(5).

sed: *A Non-Interactive Streaming Editor* tutorial in the *Text Processing Users Guide*.

STANDARDS CONFORMANCE

sed: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

NAME

send_sound - play an audio file

SYNOPSIS

/opt/audio/bin/send_sound [-*format_switch*] [**-server** *system*] [**-loop** *number*]
[**-pri** *priority*] [**-srate** *rate*] [**-prate** *rate*] [**-stereo**]

DESCRIPTION

This command plays an audio file. **send_sound** is the command used when you double-click an audio file from the HP VUE File Manager. The file begins playing, according to the settings of the Audio Control Panel.

-*format_switch*
is one these formats:

au	Sun file format
snd	NeXT file format
wav	Microsoft RIFF Waveform file format
u	MuLaw format
al	ALaw
l16	linear 16-bit format
lo8	offset (unsigned) linear 8-bit format
l8	linear 8-bit format

If you omit the filename with this option, **send_sound** plays the audio data from **stdin**.

-server *system*
plays the file on the output of an audio server identified by *system* which is either a system name or a TCP/IP address.

-loop *number*
plays the file the *number* of times you supply. Note that you cannot use this option if the source is **stdin**; a filename is required.

-pri *priority* plays file with the *priority* you supply, either **urgent**, **hi**, **normal**, or **lo**.

-srate *rate* sets the sample *rate* of the source audio.

-prate *rate* plays the audio file at the sampling *rate* you enter.

-stereo plays a stereo file. This option is needed only for a raw data file with stereo data.

AUTHOR

send_sound was developed by HP.

NeXT is a trademark of NeXT Computers, Inc.

Microsoft is a trademark of Microsoft Corporation.

SEE ALSO

audio(5), asecure(1M), aserver(1M), attributes(1), convert(1).

NAME

serialize - force target process to run serially with other processes

SYNOPSIS

serialize *command* [*command_args*]

serialize [**-t**] [**-p** *pid*]

DESCRIPTION

The **serialize** command is used to force the target process to run serially with other processes also marked by this command. The target process can be referred to by *pid* value, or it can be invoked directly on the *command*. Once a process has been marked by **serialize**, the process stays marked until process completion unless **serialize** is reissued on the serialized process with the **-t** option. The **-t** option causes the *pid* specified with the **-p** option to return to normal timeshare scheduling algorithms.

This call is used to improve process throughput, since process throughput usually increases for large processes when they are executed serially instead of allowing each program to run for only a short period of time. By running large processes one at a time, the system makes more efficient use of the CPU as well as system memory, since each process does not end up constantly faulting in its working set, to only have the pages stolen when another process starts running. As long as there is enough memory in the system, processes marked by **serialize** behave no differently from other processes in the system. However, once memory becomes tight, processes marked by **serialize** are run one at a time with the highest priority processes being run first. Each process will run for a finite interval of time before another serialized process is allowed to run.

Options

serialize supports the following options:

-t	Indicates the process specified by *pid* should be returned to timeshare scheduling.
-p	Indicates the *pid* of the target process.

If neither option is specified, **serialize** is invoked on the command line passed in.

RETURN VALUE

serialize returns the following value:

0	Successful completion.
1	Invalid *pid* specification, nonnumeric entry, or <= PID_MAXSYS.
2	Could not execute the specified command.
3	No such process.
4	Must be root or a member of a group having SERIALIZE privilege to execute **serialize**.

ERRORS

serialize fails under the following condition and sets **errno** (see *errno*(2)) to the following value:

[ESRCH]	The *pid* passed in does not exist.

EXAMPLES

Use **serialize** to force a database application to run serially with other processes marked for serialization:

```
serialize database_app
```

Force a currently running process with a *pid* value of 215 to run serially with other processes marked for serialization:

```
serialize -p 215
```

Return a process previously marked for serialization to normal timeshare scheduling. The *pid* of the target process for this example is **174**:

```
serialize -t -p 174
```

WARNINGS

The user has no way of forcing an execution order on serialized processes.

AUTHOR

serialize was developed by HP.

SEE ALSO

setprivgrp(1M), getprivgrp(2), serialize(2), privilege(5).

NAME

sh, rsh - Bourne shell, the standard/restricted command programming language

SYNOPSIS

sh [**- -acefhiknrstuvx** ...] [*arg* ...]

rsh [**- -acefhiknrstuvx** ...] [*arg* ...]

DESCRIPTION

sh is a command programming language that executes commands read from a terminal or a file. **rsh** is a restricted version of the standard command interpreter **sh**; it is used to set up login names and execution environments whose capabilities are more controlled than those of the standard shell. See "Invocation" and "Special Commands" sections later in this entry for details about command line options and arguments, particularly the **set** command.

Definitions

A **blank** is a tab or a space character. A **name** is a sequence of letters, digits, or underscores beginning with a letter or underscore. A **parameter** is a name, a digit, or any of the characters *****, **@**, **#**, **?**, **-**, **$**, and **!**.

Commands

A **simple-command** is a sequence of non-blank **words** separated by **blanks**. The first word specifies the name of the command to be executed. Except as specified below, the remaining words are passed as arguments to the invoked command. The command name is passed as argument 0 (see *exec*(2)). The **value** of a simple-command is its exit status if it terminates normally, or (octal) 200+**status** if it terminates abnormally (see *signal*(5) for a list of status values).

A **pipeline** is a sequence of one or more **commands** separated by **|** (or, for historical compatibility, by **^**). The standard output of each command except the last is connected by a *pipe*(2) to the standard input of the next command. Each command is run as a separate process and the shell waits for the last command to terminate. The exit status of a pipeline is the exit status of the last command.

A **list** is a sequence of one or more pipelines separated by **;**, **&**, **&&**, or **||**, and optionally terminated by **;** or **&**. Of these four symbols, **;** and **&** have equal precedence, which is lower than that of **&&** and **||**. The symbols **&&** and **||** also have equal precedence. A semicolon (**;**) causes sequential execution of the preceding pipeline; an ampersand (**&**) causes asynchronous execution of the preceding pipeline (i.e., the shell does *not* wait for that pipeline to finish). The symbol **&&** (**||**) causes the *list* following it to be executed only if the preceding pipeline returns a zero (non-zero) exit status. An arbitrary number of new-line characters instead of semicolons can appear in a **list** to delimit commands.

A **command** is either a simple-command or one of the following. Unless otherwise stated, the value returned by a command is that of the last simple-command executed in the command.

for *name* [**in** *word* ...] **do** *list* **done**

Each time a **for** command is executed, *name* is set to the next *word* taken from the **in** *word* list. If **in** *word* ... is omitted, the **for** command executes the **do** *list* once for each positional parameter that is set (see *Parameter Substitution* below). Execution ends when there are no more words in the list.

case *word* **in** [*pattern* [| *pattern*] ... **)** *list* **;;**] ... **esac**

A **case** command executes the *list* associated with the first *pattern* that matches *word*. The form of the patterns is the Pattern Matching Notation as qualified for the **case** command (see *regexp*(5)).

if *list* **then** *list* [**elif** *list* **then** *list*] ... [**else** *list*] **fi**

The *list* following **if** is executed and, if it returns a zero exit status, the *list* following the first **then** is executed. Otherwise, the *list* following **elif** is executed and, if its value is zero, the *list* following the next **then** is executed. Failing that, the **else** *list* is executed. If no **else** *list* or **then** *list* is executed, the **if** command returns a zero exit status.

while *list* **do** *list* **done**

A **while** command repeatedly executes the **while** *list* and, if the exit status of the last command in the list is zero, executes the **do** *list*; otherwise the loop terminates. If no commands in the **do** *list* are executed, the **while** command returns a zero exit status; **until** can be used in place of **while** to negate the loop termination test.

(*list* **)** Execute *list* in a sub-shell.

{ *list* **;}** The purpose of using braces is to allow the aggregate output from *list* to be redirected elsewhere. If redirection is used as in:

{ *list* **; } >** *file*

a subshell is created to execute *list*. This implies that any shell variables set, created, or modified within the **{ }** do not retain their values outside the **{ }**. If no redirection is used, a subshell is not created, and shell modifications made within the **{ }** are preserved.

name **( ) {** *list* **; }**
Define a function which is referenced by *name*. The body of the function is the *list* of commands between **{** and **}**. Execution of functions is described below (see "Execution").

The following words are recognized only as the first word of a command, and when not quoted:

if then else elif fi case esac for while until do done { }

Comments

A word beginning with **#** causes that word and all the following characters up to a new-line character to be ignored.

Command Substitution

The standard output from a command enclosed in a pair of grave accents (**` `**) can be used as part or all of a word; trailing new-lines are removed.

Parameter Substitution

The **$** character is used to introduce substitutable *parameters*. There are two types of parameters, positional and keyword. If *parameter* is a digit, it is a positional parameter. Positional parameters can be assigned values by **set**. Keyword parameters (also known as variables) can be assigned values by writing:

*name***=***value*[*name***=***value*]...

Pattern-matching is not performed on *value*. Having a function and a variable with the same *name* is not allowed.

${ *parameter* **}**
The value, if any, of the parameter is substituted. Braces are required only when *parameter* is followed by a letter, digit, or underscore that is not to be interpreted as part of its name. If *parameter* is ***** or **@**, all positional parameters, starting with **$1**, are substituted (separated by spaces). Parameter **$0** is set from argument zero when the shell is invoked.

${*parameter* **:-** *word***}**
If *parameter* is set and is non-null, substitute its value; otherwise substitute *word*.

${*parameter* **:=***word***}**
If *parameter* is not set or is null, set it to *word*; the value of the parameter is then substituted. Positional parameters cannot be assigned to in this way.

${*parameter* **:?***word***}**
If *parameter* is set and is non-null, substitute its value; otherwise, print *word* and exit from the shell. If *word* is omitted, the message "parameter null or not set" is printed.

${*parameter* **:+***word***}**
If *parameter* is set and is non-null, substitute *word*; otherwise substitute nothing.

In the above, *word* is not evaluated unless it is to be used as the substituted string. Thus, in the following example, **pwd** is executed only if **d** is not set or is null:

```
echo ${d:-`pwd`}
```

If the colon (**:**) is omitted from the above expressions, the shell only checks whether *parameter* is set or not.

The following parameters are automatically set by the shell:

#	The number of positional parameters in decimal.
-	Flags supplied to the shell on invocation or by the **set** command.

S

?	The decimal value returned by the last synchronously executed command.
$	The process id of the last separately-invoked shell (i.e., a shell arising from direct invocation or **#!**). This parameter is not updated for subshells arising from **( )**, command substitution, etc.
!	The process number of the last background command invoked.
LINES **COLUMNS**	The number of lines and columns in the current display area. These parameters are set only under specific circumstances. See *Signals*.

The following parameters are used by the shell:

HOME	The default argument (home directory) for the **cd** command.
PATH	The search path for commands (see "Execution" below). Users executing under **rsh** cannot change **PATH**.
CDPATH	The search path for the **cd** command.
MAIL	If this parameter is set to the name of a mail file *and* the **MAILPATH** parameter is not set, the shell informs the user of the arrival of mail in the specified file.
MAILCHECK	This parameter specifies how often (in seconds) the shell will check for the arrival of mail in the files specified by the **MAILPATH** or **MAIL** parameters. The default value is 600 seconds (10 minutes). If set to 0, the shell checks before each prompt.
MAILPATH	A colon (**:**)-separated list of file names. If this parameter is set, the shell informs the user of the arrival of mail in any of the specified files. Each file name can be followed by **%** and a message to be printed when the modification time changes. The default message is **you have mail**.
PS1	Primary prompt string, by default "**$** ".
PS2	Secondary prompt string, by default "**>** ".
IFS	Internal field separators, normally **space**, **tab**, and **new-line**.
SHACCT	If this parameter is set to the name of a file writable by the user, the shell writes an accounting record in the file for each shell procedure executed. Accounting routines such as *acctcom*(1M) and *acctcms*(1M) can be used to analyze the data collected.
SHELL	When the shell is invoked, it scans the environment (see "Environment" below) for this name. If it is found and there is an **r** in the file name part of its value, the shell becomes a restricted shell. **SHELL** is also used by some processors to determine which command interpreter to run.

The shell gives default values to **PATH**, **PS1**, **PS2**, **MAILCHECK**, and **IFS**. **HOME** and **MAIL** are set by *login*(1).

S

Blank Interpretation

After parameter and command substitution, the results of substitution are scanned for internal field separator characters (those found in **IFS**) and split into distinct arguments where such characters are found. Explicit null arguments (**""** or **´´**) are retained. Implicit null arguments (those resulting from *parameters* that have no values) are removed.

File Name Generation

Following substitution, each command *word* is processed as a pattern for file name expansion. The form of the patterns is the Pattern Matching Notation defined by *regexp*(5).

Quoting

The following characters have a special meaning to the shell and cause termination of a word unless quoted:

;, **&**, **(**, **)**, **|**, **^**, **<**, **>**, new-line, space, tab, **#** (comment)

A character can be **quoted** (i.e., made to stand for itself) by preceding it with a ****. The pair *****new-line* is ignored. All characters enclosed between a pair of single quote marks (**´´**), except a single quote, are quoted. Inside double quote marks (**""**), parameter and command substitution occurs and **** quotes the characters ****, **`**, **"**, and **$**. **"$*"** is equivalent to **"$1 $2** ...**"**, whereas **"$@"** is equivalent to **"$1""$2"**

. . . .

Prompting

When used interactively, the shell prompts with the value of **PS1** before reading a command. If at any time a new-line is typed and further input is needed to complete a command, the secondary prompt (i.e., the value of **PS2**) is issued.

Input/Output

Before a command is executed, its input and output can be redirected using a special notation interpreted by the shell. The following can appear anywhere in a simple-command or can precede or follow a *command* and are *not* passed on to the invoked command; substitution occurs before *word* or *digit* is used:

<*word*	Use file *word* as standard input (file descriptor 0).
>*word*	Use file *word* as standard output (file descriptor 1). If the file does not exist then it is created; otherwise, it is truncated to zero length.
>>*word*	Use file *word* as standard output. If the file exists then output is appended to it (by first seeking to the end-of-file); otherwise, the file is created.
<<[**-**]*word*	The shell input is read up to a line that is the same as *word*, or to an end-of-file. The resulting document becomes the standard input. If any character of *word* is quoted, no interpretation is placed upon the characters of the document; otherwise, parameter and command substitution occurs, (unescaped) **\***new-line* is ignored, and \ must be used to quote the characters \, **$**, **`**, and the first character of *word*. If **-** is appended to **<<**, all leading tabs are stripped from *word* and from the document.
<&*digit*	Use the file associated with file descriptor *digit* as standard input. Similarly for the standard output using **>&***digit* (see *dup*(2)). Note that *digit* must be in the range 0 through 9.
<&-	The standard input is closed. Similarly for the standard output using **>&-**.

If any of the above is preceded by a digit in the range 0 through 9, the file descriptor that becomes associated with the file is that specified by the digit (rather than the default 0 or 1). For example:

```
... 2>&1
```

associates file descriptor 2 with the file currently associated with file descriptor 1. Note that this type of I/O redirection is necessary when *synchronously* collecting standard output and standard error output in the same file. Redirecting standard output and standard error separately causes asynchronous collection of data at the destination (information written to standard output can be subsequently over-written by information written to standard error and vice-versa).

The order in which redirections are specified is significant. The shell evaluates redirections left-to-right. For example:

```
... 1>xxx 2>&1
```

first associates file descriptor 1 with file *xxx*. It associates file descriptor 2 with the file associated with file descriptor 1 (i.e. *xxx*). If the order of redirections is reversed, file descriptor 2 is associated with the terminal (assuming file descriptor 1 was originally) and file descriptor 1 is associated with file *xxx*.

If a command is followed by **&**, the default standard input for the command is the empty file **/dev/null**. Otherwise, the environment for executing a command contains the file descriptors of the invoking shell as modified by input/output specifications.

Redirection of output is not allowed in the restricted shell.

S

Environment

The **environment** (see *environ*(5)) is a list of name-value pairs that is passed to an executed program in the same way as a normal argument list. The shell interacts with the environment in several ways. On invocation, the shell scans the environment and creates a parameter for each name found, giving it the corresponding value. Executed commands inherit the same environment. If the user modifies the value of any of these parameters or creates new parameters, none of these affects the environment unless the **export** command is used to bind the shell's parameter to the environment (see also **set -a**). To remove a parameter from the environment, use the **unset** command. The environment seen by any executed command is thus composed of any unmodified name-value pairs originally inherited by the shell, minus any pairs removed by **unset**, plus any modifications or additions, all of which must be noted in **export** commands.

The environment for any *simple-command* can be augmented by prefixing it with one or more assignments to parameters. Thus:

```
TERM=450 cmd
```

and

```
(export TERM; TERM=450; cmd)
```

are equivalent (as far as the execution of *cmd* is concerned).

If the **-k** option is set, *all* keyword arguments are placed in the environment, even if they occur after the command name. The following first prints **a=b c** and then **c**:

```
echo a=b c
set -k
echo a=b c
```

Signals

The **INTERRUPT** and **QUIT** signals for an invoked command are ignored if the command is followed by **&** and the command does not override the **SIGINT** and **SIGQUIT** signal settings it inherited from **/usr/old/bin/sh**; otherwise signals have the values inherited by the shell from its parent, with the exception of signal 11 (but see also the **trap** command below).

If a SIGWINCH signal is received, **sh** determines the new size of the display area and resets the values of the **LINES** and **COLUMNS** variables appropriately. Note that the value of either variable is modified only if that variable exists at the time SIGWINCH is received. **sh** does not create these variables. Any traps set on SIGWINCH are executed immediately after **LINES** and **COLUMNS** are reset.

Execution

Each time a command is executed, the above substitutions are carried out. If the command name matches one of the *Special Commands* listed below, it is executed in the shell process. If the command name does not match a *Special Command*, but matches the name of a defined function, the function is executed in the shell process (note how this differs from the execution of shell procedures). The positional parameters **$1**, **$2**, ... are set to the arguments of the function. If the command name matches neither a *Special Command* nor the name of a defined function, a new process is created and an attempt is made to execute the command via *exec*(2).

The shell parameter **PATH** defines the search path for the directory containing the command. Alternative directory names are separated by a colon (**:**). The default path is **:/usr/bin** (specifying the current directory and **/usr/bin**, in that order). Note that the current directory is specified by a null path name, which can appear immediately after the equal sign or between the colon delimiters anywhere else in the path list. If the command name contains a **/** the search path is not used (such commands are not executed by the restricted shell). Otherwise, each directory in the path is searched for an executable file. If the file has execute permissions but is not a directory or an executable object code file, it is assumed to be a script file which is a file of data for an interpreter. If the first two characters of the script file are **#!**, **exec** (see *exec*(2)) expects an interpreter path name to follow. **exec** then attempts to execute the specified interpreter as a separate process to read the entire script file. If a call to **exec** fails, **/usr/old/bin/sh** is spawned to interpret the script file.

A parenthesized command is also executed in a sub-shell. The location in the search path where a command was found is remembered by the shell (to help avoid unnecessary **exec**s later). If the command was found in a relative directory, its location must be re-determined whenever the current directory changes. The shell forgets all remembered locations whenever the **PATH** variable is changed or the **hash -r** command is executed (see below).

Special Commands

The following commands are executed in the shell process. Input/output redirection is permitted for these commands. File descriptor 1 is the default output location.

:	No effect; the command does nothing. A zero exit code is returned.
. *file*	Read and execute commands from *file* and return. The search path specified by **PATH** is used to find the directory containing *file*. Note that this command does not spawn another shell to execute *file*, and thus differs in behavior and output from executing *file* as a shell script. It is not necessary that the execute permission bit be set

S

for *file*.

break [*n*] Exit from the enclosing **for** or **while** loop, if any. If *n* is specified, break *n* levels.

continue [*n*] Resume the next iteration of the enclosing **for** or **while** loop. If *n* is specified, resume at the *n*-th enclosing loop.

cd [*arg*] Change the current directory to *arg*. The shell parameter **HOME** is the default *arg*. The shell parameter **CDPATH** defines the search path for the directory containing *arg*. Alternative directory names are separated by a colon (**:**). The default path is null (meaning the current directory). Note that the current directory is specified by a null path name, which can appear immediately after the equal sign or between the colon delimiters anywhere else in the path list. If *arg* begins with a **/** the search path is not used. Otherwise, each directory in the path is searched for *arg*. The **cd** command cannot be executed by **rsh**.

echo [*arg* ...] Echo arguments. See *echo*(1) for usage and description.

eval [*arg* ...] The arguments are read as input to the shell and the resulting command(s) executed.

exec [*arg* ...] The command specified by the arguments is executed in place of this shell without creating a new process. Input/output arguments can appear and, if no other arguments are given, cause the shell input/output to be modified.

exit [*n*] Causes a shell to exit with the exit status specified by *n*. If *n* is omitted, the exit status is that of the last command executed (an end-of-file also causes the shell to exit.)

export [*name* ...]
The given *name*s are marked for automatic export to the *environment* of subsequently-executed commands. If no arguments are given, a list of names currently included in the environment are printed. Function names *cannot* be exported.

hash [**-r**] [*name* ...]
For each *name*, the location in the search path of the command specified by *name* is determined and remembered by the shell. The **-r** option causes the shell to forget all remembered locations. If no arguments are given, information about remembered commands is presented. *hits* is the number of times a command has been invoked by the shell process. *cost* is a measure of the work required to locate a command in the search path. Certain situations require that the stored location of a command be recalculated. Commands for which this will be done are indicated by an asterisk (*****) adjacent to the *hits* information. *cost* is incremented when the recalculation is done.

newgrp [*arg* ...] Equivalent to **exec newgrp** *arg* .... See *newgrp*(1) for usage and description.

pwd Print the current working directory. **read** *name* ... One line is read from the standard input and the first word is assigned to the first *name*, the second word to the second *name*, etc., with leftover words assigned to the last *name*. The return code is 0 unless an end-of-file is encountered.

Note that although the read command is a built-in command and is generally executed directly by the shell, this is not the case when it is used in a pipeline. In a pipeline, a new shell is forked to execute the read command with the result that any shell variables set are not available to the parent shell when the pipeline is finished. This has the effect of making the read command useless in a pipeline.

readonly [*name* ...]
The given *name*s are marked *readonly* and the values of the these *name*s cannot be changed by subsequent assignment. If no arguments are given, a list of all *readonly* names is printed.

return [*n*] Causes a function to exit with the return value specified by *n*. If *n* is omitted, the return status is that of the last command executed.

set [**- -aefhkntuvx** [*arg* ...]]
set sets or unsets options, and resets the values of the positional parameters to the args given, if any. The option list is terminated by the first argument that does not begin with **-** or **+**, or upon encountering an argument consisting entirely of **- -**. Recognized options are:

- **-a** Mark variables which are modified or created for export.
- **-e** Exit immediately if a command exits with a non-zero exit status.
- **-f** Disable file name generation
- **-h** Locate and remember function commands as functions are defined (function commands are normally located when the function is executed).
- **-k** All keyword arguments are placed in the environment for a command, not just those that precede the command name.
- **-n** Read commands but do not execute them.
- **-t** Exit after reading and executing one command.
- **-u** Treat unset variables as an error when substituting.
- **-v** Print shell input lines as they are read.
- **-x** Print commands and their arguments as they are executed.
- **- -** Do not change any of the options; useful when **$1** is to be set to a string beginning with **-** or **+**.

Using **+** rather than **-** causes these options to be unset. These options can also be used upon invocation of the shell. The current set of options can be found in **$-**. The remaining arguments are positional parameters and are assigned, in order, to **$1**, **$2**, ... If no arguments are given, the values of all names are printed.

shift [*n*] The positional parameters from **$n+1** ... are renamed **$1** ... . If *n* is not given, it is assumed to be 1.

test Evaluate conditional expressions. See *test*(1) for usage and description. Note that **[** ... **]** in an **if** *list* is interpreted the same as **test** ... . There must be blanks around the brackets.

times Print the accumulated user and system times for processes run from the shell.

trap [*arg*] [*n*] ... The command *arg* is a command to be read and executed when the shell receives signal(s) *n*. (Note that *arg* is scanned once when the trap is set and once when the trap is taken.) Trap commands are executed in order of signal number. Any attempt to set a trap on a signal that was ignored on entry to the current shell is ineffective. An attempt to trap on signal 11 (memory fault) or signal 18 (death of child) produces an error. If *arg* is absent then all trap(s) *n* are reset to their original values. If *arg* is the null string, this signal is ignored by the shell and by the commands it invokes. If *n* is 0, the command *arg* is executed on exit from the shell. The **trap** command with no arguments prints a list of commands associated with each signal number.

type [*name* ...] For each *name*, indicate how it would be interpreted if used as a command name.

ulimit [**-f** [*n*]] If the **-f** *n* option is used, a size limit of *n* blocks is imposed on files written by child processes (files of any size can be read). If *n* is not specified, the current limit is printed. If no option is specified, **-f** is assumed.

umask [*nnn*] The user file-creation mask is set to *nnn* (see *umask*(2)). If *nnn* is omitted, the current value of the mask is printed.

unset [*name* ...] For each *name*, remove the corresponding variable or function. The variables **PATH**, **PS1**, **PS2**, **MAILCHECK**, and **IFS** cannot be unset.

wait [*n*] Wait for the specified process and report its termination status. If *n* is not given all currently active child processes are waited for and the return code is zero.

Invocation

Options can be specified in a single argument or in multiple arguments, but in all cases each option argument must begin with **-**. (All options except **c**, **s**, **i**, and **r** can also be prefaced with a **+**, which turns off the associated option or options, but this is redundant when invoking a new shell because all options are turned off by default).

If the first character of argument zero is **-**, commands are initially read from **/etc/profile**, then from **$HOME/.profile**, if the files exist. Thereafter, commands are read as described below.

The options below are interpreted by the shell at invocation (thus they cannot be used with the **set** command). Unless the **-c** or **-s** option is specified, the first non-option argument is assumed to be the name of a file containing commands, and the remaining arguments are passed as positional parameters to that command file.

-c *string* If the **-c** option is present then commands are read from *string*.

-s If the **-s** option is present or if no arguments remain, commands are read from the standard input. Any remaining arguments specify the positional parameters. Shell output (except for "Special Commands") is written to file descriptor 2.

-i If the **-i** option is present or if the shell input and output are attached to a terminal, this shell is *interactive*. In this case TERMINATE is ignored (so that **kill 0** does not kill an interactive shell) and INTERRUPT is caught and ignored (so that **wait** is interruptible). In all cases, QUIT is ignored by the shell.

-r If the **-r** option is present the shell is a restricted shell.

The remaining options and arguments are described under the **set** command above.

rsh Only

rsh is used to set up login names and execution environments whose capabilities are more controlled than those of the standard shell. The actions of **rsh** are identical to those of **sh**, except that the following are disallowed:

- Changing directory (see *cd*(1)),
- Setting the value of **$PATH,**
- Specifying path or command names containing **/**,
- Redirecting output (**>** and **>>**).

The restrictions above are enforced after **.profile** is interpreted.

When a command to be executed is found to be a shell procedure, **rsh** invokes **sh** to execute it. Thus, it is possible to provide to the end-user shell procedures that have access to the full power of the standard shell, while imposing a limited menu of commands; this scheme assumes that the end-user does not have write and execute permissions in the same directory.

The net effect of these rules is that the writer of the **.profile** has complete control over user actions, by performing guaranteed setup actions and leaving the user in an appropriate directory (probably *not* the login directory).

The system administrator often sets up a directory of commands (such as **/usr/rbin**) that can be safely invoked by **rsh**. Commands such as **vi**, **sh**, **ksh**, **csh**, and such that can break **rsh** restrictions should not be included in that directory. Some systems also provide a restricted editor **red**.

S

EXTERNAL INFLUENCES

Environment Variables

LC_COLLATE determines the collating sequence used in evaluating pattern matching notation for file name generation.

LC_CTYPE determines the interpretation of text as single and/or multi-byte characters, the classification of characters as letters, and the characters matched by character class expressions in pattern matching notation.

LANG determines the language in which messages are displayed.

If LC_COLLATE or LC_CTYPE is not specified in the environment or is set to the empty string, the value of LANG is used as a default for each unspecified or empty variable. If LANG is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of LANG. If any internationalization variable contains an invalid setting, **sh** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported.

RETURN VALUE

The error codes returned by the shell are:

0 Success.

1 Built-in command failure (see *Special Commands*).
2 Syntax error.
3 Signal received that is not trapped.

If the shell is non-interactive, it terminates and passes one of the above as its exit status. If it is interactive, it does not terminate, but **$?** is set to one of the above values.

Whenever a child process of the shell dies due to a signal, the shell returns an exit status of 80 hexadecimal plus the number of the signal.

WARNINGS

If a command is executed and a command having the same name is installed in a directory in the search path before the directory where the original command was found, the shell continues to **exec** the original command. Use the **hash** command to correct this situation.

When the shell encounters **>>**, it does not open the file in append mode. Instead, it opens the file for writing and seeks to the end.

If you move the current directory or one above it, **pwd** cannot give the correct response. Use the **cd** command with a full path name to correct this situation.

The command **readonly** (without arguments) produces the same output as the command **export**.

Failure (non-zero exit status) of a special command preceding a **||** symbol prevents the *list* following **||** from executing.

In an international environment, character ordering is determined by the setting of LC_COLLATE, rather than by the binary ordering of character values in the machine collating sequence. This brings with it certain attendant dangers, particularly when using range expressions in file-name-generation patterns. For example, the command,

```
rm [a-z]*
```

might be expected to match all file names beginning with a lowercase alphabetic character. However, if dictionary ordering is specified by LC_COLLATE, it also matches file names beginning with an uppercase character (as well as those beginning with accented letters). Conversely, it fails to match letters collated after **z** in languages such as Norwegian.

The correct (and safe) way to match specific character classes in an international environment is to use a pattern of the form:

```
rm [[:lower:]]*
```

This uses LC_CTYPE to determine character classes and works predictably for all supported languages and codesets. For shell scripts produced on non-internationalized systems (or without consideration for the above dangers), it is recommended that they be executed in a non-NLS environment. This requires that LANG, LC_COLLATE, etc., be set to "C" or not set at all.

sh implements command substitution by creating a pipe between itself and the command. If the root file system is full, the substituted command cannot write to the pipe. As a result, the shell receives no input from the command, and the result of the substitution is null. In particular, using command substitution for variable assignment under such circumstances results in the variable being silently assigned a NULL value.

The signal SIGSEGV should not be blocked when executing **sh**.

sh reserves file descriptor 59 for internal use. Reducing the number of available file descriptors below 60 causes **sh** to fail.

Each time a function is executed in a shell script, any arguments given to the function overwrite the values of the positional parameters for the entire script. If the values of the positional parameters must be preserved, they must be explicitly saved before each function call.

AUTHOR

sh was developed by AT&T and HP.

FILES

$HOME/.profile
/dev/null
/etc/profile
/tmp/sh*

SEE ALSO

cd(1), echo(1), env(1), login(1), newgrp(1), pwd(1), test(1), umask(1), acctcms(1M), acctcom(1M), dup(2), exec(2), fork(2), pipe(2), ulimit(2), umask(2), wait(2), a.out(4), profile(4), environ(5), lang(5), regexp(5), signal(5).

Bourne Shell tutorial in *Shells Users Guide*.

STANDARDS CONFORMANCE

`.`: SVID2, SVID3, XPG2, XPG3

`:`: SVID2, SVID3, XPG2, XPG3

`break`: SVID2, SVID3, XPG2, XPG3

`case`: SVID2, SVID3, XPG2, XPG3

`continue`: SVID2, SVID3, XPG2, XPG3

`eval`: SVID2, SVID3, XPG2, XPG3

`exec`: SVID2, SVID3, XPG2, XPG3

`exit`: SVID2, SVID3, XPG2, XPG3

`export`: SVID2, SVID3, XPG2, XPG3

`for`: SVID2, SVID3, XPG2, XPG3

`if`: SVID2, SVID3, XPG2, XPG3

`read`: SVID2, SVID3, XPG2, XPG3

`return`: SVID2, SVID3, XPG2, XPG3

`rsh`: SVID2, SVID3

`set`: SVID2, SVID3, XPG2, XPG3

`sh`: SVID2, SVID3, XPG2, XPG3

`shift`: SVID2, SVID3, XPG2, XPG3

`time`: SVID2, SVID3, XPG2, XPG3

`trap`: SVID2, SVID3, XPG2, XPG3

`unset`: SVID2, SVID3, XPG2, XPG3

`until`: SVID2, SVID3, XPG2, XPG3

`while`: SVID2, SVID3, XPG2, XPG3

NAME

sh, rsh - standard and restricted POSIX.2-conformant command shells

SYNOPSIS

sh [{**-**|**+**}**aefhikmnprstuvx**] [{**-**|**+**}**o** *option*]... [**-c** *string*] *arg*]...

rsh [{**-**|**+**}**aefhikmnprstuvx**] [{**-**|**+**}**o** *option*]... [**-c** *string*] [*arg*]...

Remarks

This shell is intended to conform to the shell specification of the POSIX.2 *Shell and Utility* standards. Check any standards conformance documents shipped with your system for information on the conformance of this shell to any other standards.

List of Subheadings in DESCRIPTION

Shell Invocation	Tilde Substitution	Environment
Options	Command Substitution	Functions
rsh Restrictions	Parameter Substitution	Jobs
Definitions	Blank Interpretation	Signals
Commands	File Name Generation	Execution
Simple Commands	Quoting	Command Reentry
Compound Commands	Arithmetic Evaluation	Command Line Editing
Special Commands	Prompting	emacs Editing Mode
Comments	Conditional Expressions	vi Editing Mode
Aliasing	Input/Output	

DESCRIPTION

sh is a command programming language that executes commands read from a terminal or a file.

rsh is a restricted version of **sh**. See the "rsh Restrictions" subsection below.

Shell Invocation

If the shell is invoked by an **exec*****()** system call and the first character of argument zero (shell parameter **0**) is dash (**-**), the shell is assumed to be a login shell and commands are read first from **/etc/profile**, then from either **.profile** in the current directory or **$HOME/.profile** if either file exists, and finally from the file named by performing parameter substitution on the value of the environment parameter **ENV**, if the file exists. If the **-s** option is not present and an *arg* is, a path search is performed on the first *arg* to determine the name of the script to execute. When running **sh** with *arg*, the script *arg* must have read permission and any **setuid** and **setgid** settings will be ignored. Commands are read as described below.

Shell output, except for the output of some of the commands listed in the "Special Commands" subsection, is written to standard error (file descriptor 2).

S

Options

The following options are interpreted by the shell when it is invoked.

-c *string* Read commands from *string*.

-i If **-i** is present or if the shell input and output are attached to a terminal (as reported by **tty()**), the shell is interactive. In this case **SIGTERM** is ignored and **SIGINT** is caught and ignored (so that **wait** is interruptible). In all cases, **SIGQUIT** is ignored by the shell. See *signal*(5).

-r The shell is a restricted shell.

-s If **-s** is present or if no arguments remain, commands are read from the standard input.

The remaining options and arguments are described under the **set** command in the "Special Commands" subsection.

rsh Restrictions

rsh is used to set up login names and execution environments where capabilities are more controlled than those of the standard shell. The actions of **rsh** are identical to those of **sh**, except that the following are forbidden:

- Changing directory (see the **cd** special command and *cd*(1))

- Setting the value of **SHELL**, **ENV**, or **PATH**
- Specifying path or command names containing **/**
- Redirecting output (**>**, **>|**, **<>**, and **>>**)

The restrictions above are enforced after the **.profile** and **ENV** files are interpreted.

When a command to be executed is found to be a shell procedure, **rsh** invokes **sh** to execute it. Thus, the end-user is provided with shell procedures accessible to the full power of the standard shell, while being restricted to a limited menu of commands. This scheme assumes that the end-user does not have write and execute permissions in the same directory.

These rules effectively give the writer of the **.profile** file complete control over user actions, by performing guaranteed set-up actions and leaving the user in an appropriate directory (probably not the login directory).

The system administrator often sets up a directory of commands (usually **/usr/rbin**) that can be safely invoked by **rsh**. HP-UX systems provide a restricted editor **red** (see *ed*(1)), suitable for restricted users.

Definitions

metacharacter	One of the following characters: **; & ( ) | < >** newline space tab
blank	A tab or a space.
identifier	A sequence of letters, digits, or underscores starting with a letter or underscore. Identifiers are used as names for **functions** and **named parameters**.
word	A sequence of **characters** separated by one or more nonquoted **metacharacters**.
command	A sequence of characters in the syntax of the shell language. The shell reads each command and carries out the desired action, either directly or by invoking separate utilities.
special command	A command that is carried out by the shell without creating a separate process. Except for documented side effects, most special commands can be implemented as separate utilities.
#	Comment delimiter. A word beginning with **#** and all following characters up to a newline are ignored.
parameter	An **identifier**, a decimal number, or one of the characters **!**, **#**, **$**, *****, **-**, **?**, **@**, and **_**. See the "Parameter Substitution" subsection.
named parameter	A **parameter** that can be assigned a value. See the "Parameter Substitution" subsection.
variable	A **parameter**.
environment variable	A **parameter** that is known outside the local shell, usually by means of the **export** special command.

S

Commands

A command can be a simple command that executes an executable file, a special command that executes within the shell, or a compound command that provides flow of control for groups of simple, special, and compound commands.

Simple Commands

A simple command is a sequence of blank-separated words that may be preceded by a parameter assignment list. (See the "Environment" subsection). The first word specifies the name of the command to be executed. Except as specified below, the remaining words are passed as arguments to the invoked command. The command name is passed as argument **0** (see *exec*(2)). The *value* of a simple-command is its exit status if it terminates normally, or **128+***errorstatus* if it terminates abnormally (see *signal*(5) for a list of *errorstatus* values).

A **pipeline** is a sequence of one or more commands separated by a bar (**|**) and optionally preceded by an exclamation mark (**!**). The standard output of each command but the last is connected by a pipe (see *pipe*(2)) to the standard input of the next command. Each command is run as a separate process; the shell waits for the last command to terminate. If **!** does not precede the pipeline, the exit status of the pipeline is the exit status of the last command in the pipeline. Otherwise, the exit status of the pipeline is the logical negation of the exit status of the last command in the pipeline.

A *list* is a sequence of one or more pipelines separated by **;**, **&**, **&&**, or **||**, and optionally terminated by **;**, **&**, or **|&**.

; Causes sequential execution of the preceding pipeline. An arbitrary number of newlines can appear in a *list*, instead of semicolons, to delimit commands.

& Causes asynchronous execution of the preceding pipeline (that is, the shell does not wait for that pipeline to finish).

|& Causes asynchronous execution of the preceding command or pipeline with a two-way pipe established to the parent shell. The standard input and output of the spawned command can be written to and read from by the parent shell using the **-p** option of the special commands **read** and **print**.

&& Causes the *list* following it to be executed only if the preceding pipeline returns a zero value.

|| Causes the *list* following it to be executed only if the preceding pipeline returns a nonzero value.

Of these five symbols, **;**, **&**, and **|&** have equal precedence, which is lower than that of **&&** and **||**. The symbols **&&** and **||** also have equal precedence.

Compound Commands

Unless otherwise stated, the value returned by a compound command is that of the last simple command executed in the compound command. The **;** segment separator can be replaced by one or more newlines.

The following keywords are recognized only as the first word of a command and when not quoted:

```
!          }          elif       for        then
[[         case       else       function   time
]]         do         esac       if         until
{          done       fi         select     while
```

A compound command is one of the following.

case *word* **in** [[**;**] [**(**] *pattern* [**|** *pattern*]**...)** *list* **;;**]... **; esac**

Execute the *list* associated with the first *pattern* that matches *word*. The form of the patterns is identical to that used for file name generation (see the "File Name Generation" subsection). The **;;** case terminator cannot be replaced by newlines.

for *identifier* [**in** *word* **...] ; do** *list* **; done**

Set *identifier* to each *word* in sequence and execute the **do** *list*. If **in** *word* ... is omitted, set *identifier* to each set positional parameter instead. See the "Parameter Substitution" subsection. Execution ends when there are no more positional parameters or words in the list.

function *identifier* **{** *list* **; }**
identifier **() {** *list* **; }**

Define a function named by *identifier*. A function is called by executing its identifier as a command. The body of the function is the *list* of commands between **{** and **}**. See the "Functions" subsection.

if *list* **; then** *list* **;** [**elif** *list* **; then** *list* **;**]... [**else** *list* **;**] **fi**

Execute the **if** *list* and, if its exit status is zero, execute the first **then** *list*. Otherwise, execute the **elif** *list* (if any) and, if its exit status is zero, execute the next **then** *list*. Failing that, execute the **else** *list* (if any). If no **else** *list* or **then** *list* is executed, **if** returns a zero exit status.

select *identifier* [**in** *word* ...] **; do** *list* **; done**

Print the set of *word*s on standard error (file descriptor 2), each preceded by a number. If **in** *word* ... is omitted, print the positional parameters instead (see the "Parameter Substitution" subsection). Print the **PS3** prompt and read a line from standard input into the parameter **REPLY**. If this line consists of the number of one of the listed *word*s, set *identifier* to the corresponding *word*, execute *list*, and repeat the **PS3** prompt. If the line is empty, print the selection list again, and repeat the **PS3** prompt. Otherwise, set *identifier* to null, execute *list*, and repeat the **PS3** prompt. The select loop repeats until a **break** special command or end-of-file is encountered.

S

time *pipeline*

Execute the *pipeline* and print the elapsed time, the user time, and the system time on standard error. See also *time*(1).

until *list* **; do** *list* **; done**

Execute the **until** *list*. If the exit status of the last command in the list is nonzero, execute the **do** *list* and execute the **until** *list* again. When the exit status of the last command in the **until** *list* is zero, terminate the loop. If no commands in the **do** *list* are executed, **until** returns a zero exit status.

while *list* **; do** *list* **; done**

Execute the **while** *list*. If the exit status of the last command in the list is zero, execute the **do** *list* and execute the **while** *list* again. When the exit status of the last command in the **while** *list* is nonzero, terminate the loop. If no commands in the **do** *list* are executed, **while** returns a nonzero exit status.

(*list* **)**

Execute *list* in a separate environment. If two adjacent open parentheses are needed for nesting, a space must be inserted between them to avoid arithmetic evaluation.

{ *list* **; }**

Execute *list*, but not in a separate environment. Note that **{** is a keyword and requires a trailing blank to be recognized.

[[*expression* **]]**

Evaluate *expression* and return a zero exit status when *expression* is true. See the "Conditional Expressions" subsection for a description of *expression*. Note that **[[** and **]]** are keywords and require blanks between them and *expression*.

Special Commands

Special commands are simple commands that are executed in the shell process. They permit input/output redirection. Unless otherwise indicated, file descriptor 1 (standard output) is the default output location and the exit status, when there are no syntax errors, is zero.

Commands that are marked with % are treated specially in the following ways:

1. Variable assignment lists preceding the command remain in effect when the command completes.
2. I/O redirections are processed after variable assignments.
3. Errors cause a script that contains them to abort.

Words following commands marked with & that are in the format of a variable assignment are expanded with the same rules as a variable assignment. This means that tilde substitution is performed after the **=** sign and word-splitting and file-name generation are not performed.

S

% **:** [*arg*]...

(colon) Only expand parameters. A zero exit status is returned.

% **.** *file* [*arg*]...

(period) Read and execute commands from *file* and return. The commands are executed in the current shell environment. The search path specified by **PATH** is used to find the directory containing *file*. If any arguments *arg* are given, they become the positional parameters. Otherwise, the positional parameters are unchanged. The exit status is the exit status of the last command executed.

& **alias** [**-tx**] [*name*[**=***value*]]...

With *name***=***value* specified, define *name* as an alias and assign it the value *value*. A trailing space in *value* causes the next word to be checked for alias substitution.

With *name***=***value* omitted, print the list of aliases in the form *name***=***value* on standard output.

With *name* specified without **=***value*, print the specified alias.

With **-t**, set tracked aliases. The value of a tracked alias is the full path name corresponding to the given *name*. The value of a tracked alias becomes undefined when the value of **PATH** is reset, but the alias remains tracked. With *name***=***value* omitted, print the list of tracked aliases in the form *name***=***pathname* on standard output.

With **-x**, set exported aliases. An exported alias is defined across subshell environments. With *name*=*value* omitted, print the list of exported aliases in the form *name*=*value* on standard output.

Alias returns true unless a *name* is given for which no alias has been defined.

bg [*job*]...

Put the specified *jobs* into the background. The current job is put in the background if *job* is unspecified. See the "Jobs" subsection for a description of the format of *job*.

% **break** [*n*]

Exit from the enclosing **for**, **select**, **until**, or **while** loop, if any. If *n* is specified, exit from *n* levels.

cd [**-L**|**-P**] [*arg*]
cd *old new*

In the first form, change the current working directory (**PWD**) to *arg*. If *arg* is **-**, the directory is changed to the previous directory (**OLDPWD**).

With **-L** (default), preserve logical naming when treating symbolic links. **cd -L ..** moves the current directory one path component closer to the root directory.

With **-P**, preserve the physical path when treating symbolic links. **cd -P ..** changes the working directory to the actual parent directory of the current directory.

The shell parameter **HOME** is the default *arg*. The parameter **PWD** is set to the current directory.

The shell parameter **CDPATH** defines the search path for the directory containing *arg*. Alternative directory names are separated by a colon (**:**). If **CDPATH** is null or undefined, the default value is the current directory. Note that the current directory is specified by a null path name, which can appear immediately after the equal sign or between the colon delimiters anywhere else in the path list. If *arg* begins with a **/**, the search path is not used. Otherwise, each directory in the path is searched for *arg*. See also *cd*(1).

The second form of **cd** substitutes the string *new* for the string *old* in the current directory name, **PWD**, and tries to change to this new directory.

command [*arg*]...

Treat *arg* as a command, but disable function lookup on *arg*. See *command*(1) for usage and description.

% **continue** [*n*]

Resume the next iteration of the enclosing **for**, **select**, **until**, or **while** loop. If *n* is specified, resume at the *n*th enclosing loop.

echo [*arg*]...

Print *arg* on standard output. See *echo*(1) for usage and description. See also the **print** special command.

% **eval** [*arg*]...

Read the arguments as input to the shell and execute the resulting commands. Allows parameter substitution for keywords and characters that would otherwise be unrecognized in the resulting commands.

% **exec** [*arg*]...

Parameter assignments remain in effect after the command completes. If *arg* is given, execute the command specified by the arguments in place of this shell without creating a new process. Input/output arguments may appear and affect the current process. If no arguments are given, modify file descriptors as prescribed by the input/output redirection list. In this case, any file descriptor numbers greater than 2 that are opened with this mechanism are closed when another program is invoked.

% **exit** [*n*]

Exit from the shell with the exit status specified by *n*. If *n* is omitted, the exit status is that of the last command executed. An end-of-file also causes the shell to exit, except when a shell has the **ignoreeof** option set. (See the **set** special command.)

%& **export** [*name*[**=***value*]]...
%& **export -p**

Mark the given variable *name*s for automatic export to the environment of subsequently executed commands. Optionally, assign values to the variables.

With **-p**, write the names and values of all exported variables to standard output, in a format with the proper use of quoting, so that it is suitable for re-input to the shell as commands that achieve the same exporting results.

fc [**-r**] [**-e** *ename*] [*first* [*last*]]
fc -l [**-nr**] [*first* [*last*]]
fc -s [*old***=***new*] [*first*]
fc -e - [*old***=***new*] [*command*]

List, or edit and reexecute, commands previously entered to an interactive shell. A range of commands from *first* to *last* is selected from the last **HISTSIZE** commands typed at the terminal. The arguments *first* and *last* can be specified as a number or string. A given string is used to locate the most recent command. A negative number is used to offset the current command number.

With **-l**, list the commands on standard output. Without **-l**, invoke the editor program *ename* on a file containing these keyboard commands. If *ename* is not supplied, the value of the parameter **FCEDIT** (default **/usr/bin/ed**) is used as the editor. Once editing has ended, the commands (if any) are executed. If *last* is omitted, only the command specified by *first* is used. If *first* is not specified, the default is the previous command for editing and –16 for listing.

With **-r**, reverse the order of the commands.

With **-n**, suppress command numbers when listing.

With **-s**, reexecute the command without invoking an editor.

The *old***=***new* argument replaces the first occurrence of string *old* in the command to be reexecuted by the string *new*.

fg [*job*]...

Bring each *job* into the foreground in the order specified. If no *job* is specified, bring the current job into the foreground. See the "Jobs" subsection for a description of the format of *job*.

getopts *optstring name* [*arg*]...

Parse the argument list, or the positional parameters if no arguments, for valid options. On each execution, return the next option in *name*. See *getopts*(1) for usage and description.

An option begins with a **+** or a **-**. An argument not beginning with **+** or **-**, or the argument **--**, ends the options. *optstring* contains the letters that **getopts** recognizes. If a letter is followed by a **:**, that option is expected to have an argument. The options can be separated from the argument by blanks.

S

For an option specified as **-***letter*, *name* is set to *letter*. For an option specified as **+***letter*, *name* is set to **+***letter*. The index of the next *arg* is stored in **OPTIND**. The option argument, if any, is stored in **OPTARG**. If no option argument is found, or the option found does not take an argument, **OPTARG** is unset.

A leading **:** in *optstring* causes **getopts** to store the letter of an invalid option in **OPTARG**, and to set *name* to **?** for an unknown option and to **:** when a required option argument is missing. Otherwise, **getopts** prints an error message. The exit status is nonzero when there are no more options.

& **hash** [*utility*]...
& **hash -r**

Affect the way the current shell environment remembers the locations of utilities. With *utility*, add utility locations to a list of remembered locations. With no arguments, print the contents of the list. With **-r**, forget all previously remembered utility locations.

jobs [**-lnp**] [*job*]...

List information about each given job, or all active jobs if *job* is not specified. With **-l**, list process IDs in addition to the normal information. With **-n**, display only jobs that have stopped or exited since last notified. With **-p**, list only the process group. See the "Jobs" subsection for a description of the format of *job*.

kill [**-s** *signal*] *process* ...
kill -l
kill [**-***signal*] *process* ...

Send either signal 15 (**SIGTERM**, terminate) or the specified *signal* to the specified jobs or processes. See *kill*(1) for usage and description.

With **-l**, list the signal names and numbers.

let *arg* ...
((*arg* ...**))**

Evaluate each *arg* as a separate arithmetic expression. See the "Arithmetic Evaluation" subsection for a description of arithmetic expression evaluation. The exit status is 0 if the value of the last expression is nonzero, and 1 otherwise.

% **newgrp** [**-**] [*group*]

Replace the current shell with a new one having *group* as the user's group. The default group is the user's login group. With **-**, the user's **.profile** and **$ENV** files are also executed. See *newgrp*(1) for usage and description. Equivalent to **exec newgrp** *arg* ....

print [**-nprRsu**[*n*]] [*arg*]...

The shell output mechanism. With no options or with option **-** or **--**, print the arguments on standard output as described in *echo*(1). See also *printf*(1).

With **-n**, do not add a newline character to the output.

With **-p**, write the arguments onto the pipe of the process spawned with **|&** instead of standard output.

With **-R** or **-r** (raw mode), ignore the escape conventions of **echo**. With **-R**, print all subsequent arguments and options other than **-n**.

With **-s**, write the arguments into the history file instead of to standard output.

With **-u**, specify a one-digit file descriptor unit number *n* on which the output will be placed. The default is **1** (standard output).

pwd [**-L**|**-P**]

Print the name of the current working directory (equivalent to **print -r - $PWD**). With **-L** (the default), preserve the logical meaning of the current directory. With **-P**, preserve the physical meaning of the current directory if it is a symbolic link. See also the **cd** special command, *cd*(1), *ln*(1), and *pwd*(1).

read [**-prsu**[*n*]] [*name***?***prompt*] [*name*]...

The shell input mechanism. Read one line (by default, from standard input) and break it up into words using the characters in **IFS** as separators. The first word is assigned to the first *name*, the second word to the second *name*, and so on; the remaining words are assigned to the last *name*. See also *read*(1). The return code is **0**, unless an end-of-file is encountered.

With **-p**, take the input line from the input pipe of a process spawned by the shell using **|&**. An end-of-file with **-p** causes cleanup for this process so that another process can be spawned.

With **-r** (raw mode), a **** at the end of a line does not signify line continuation.

With **-s**, save the input as a command in the history file.

With **-u**, specify a one-digit file descriptor unit to read from. The file descriptor can be opened with the **exec** special command. The default value of *n* is **0** (standard input). If *name* is omitted, **REPLY** is used as the default *name*.

If the first argument contains a **?**, the remainder of the argument is used as a *prompt* when the shell is interactive.

If the given file descriptor is open for writing and is a terminal device, the prompt is placed on that unit. Otherwise, the prompt is issued on file descriptor 2 (standard error).

%& **readonly** [*name*[**=***value*]]...
%& **readonly -p**

Mark the given *name*s read only. These names cannot be changed by subsequent assignment.

With **-p**, write the names and values of all read-only variables to standard output in a format with the proper use of quoting so that it is suitable for re-input to the shell as commands that achieve the same attribute-setting results.

% **return** [*n*]

Cause a shell function to return to the invoking script with the return status specified by *n*. If *n* is omitted, the return status is that of the last command executed. Only the low 8 bits of *n* (decimal 0 to 255) are passed back to the caller. If **return** is invoked while not in a function or a **.** script (see the **.** special command), it has the same effect as an **exit** command.

% **set** [{**-**|**+**}**abCefhkmnopstuvx**] [{**-**|**+**}**o** *option*]... [{**-**|**+**}**A** *name*] [*arg*]...

Set (**-**) or clear (**+**) execution options or perform array assignments (**-A**, **+A**). All options except **-A** and **+A** can be supplied in a shell invocation (see the SYNOPSIS section and the "Shell Invocation" subsection).

Using **+** instead of **-** before an option causes the option to be turned off. These options can also be used when invoking the shell. The current list of set single-letter options is contained in the shell variable **-**. It can be examined with the command **echo $-**.

The **-** and **+** options can be intermixed in the same command, except that there can be only one **-A** or **+A** option.

Unless **-A** or **+A** is specified, the remaining *arg* arguments are assigned consecutively to the positional parameters **1**, **2**, ...

The **set** command with neither arguments nor options displays the names and values of all shell parameters on standard output. See also *env*(1).

The options are defined as follows.

-A Array assignment. Unset the variable *name* and assign values sequentially from the list *arg*. With **+A**, do not unset the variable *name* first.

-a Automatically export subsequently defined parameters.

-b Cause the shell to notify the user asynchronously of background jobs as they are completed. When the shell notifies the user that a job has been completed, it can remove the job's process ID from the list of those known in the current shell execution environment.

-C Prevent redirection **>** from truncating existing files. Requires **>|** to truncate a file when turned on.

S

-e Execute the **ERR** trap, if set, and exit *if* a command has a nonzero exit status, and is not part of the compound list following a **if**, **until**, or **while** keyword, and is not part of an AND or OR list, and is not a pipeline preceded by the **!** reserved word. This mode is disabled while reading profiles.

-f Disable file name generation.

-h Specify that each command whose name is an *identifier* becomes a tracked alias when first encountered.

-k Place all parameter assignment arguments (not just those that precede the command name) into the environment for a command.

-m Run background jobs in a separate process group and print a line upon completion. The exit status of background jobs is reported in a completion message. This option is turned on automatically for interactive shells.

-n Read commands and check them for syntax errors, but do not execute them. The **-n** option is ignored for interactive shells.

-o Set an *option* argument from the following list. Repeat the **-o** option to specify additional *option* arguments.

allexport Same as **-a**.

bgnice	Run all background jobs at a lower priority.
errexit	Same as **-e**.
emacs	Use a **emacs**-style inline editor for command entry.
gmacs	Use a **gmacs**-style inline editor for command entry.
ignoreeof	Do not exit from the shell on end-of-file (*eof* as defined by **stty**; default is **^D**). The **exit** special command must be used.
keyword	Same as **-k**.
markdirs	Append a trailing **/** to all directory names resulting from file name generation.
monitor	Same as **-m**.
noclobber	Same as **-C**.
noexec	Same as **-n**.
noglob	Same as **-f**.
nolog	Do not save function definitions in history file.
notify	Same as **-b**.
nounset	Same as **-u**.
privileged	Same as **-p**.
verbose	Same as **-v**.
trackall	Same as **-h**.
vi	Use a **vi**-style inline editor for command entry.
viraw	Process each character as it is typed in **vi** mode (always on).
xtrace	Same as **-x**.

-p Disable processing of the **$HOME/.profile** file and uses the file **/etc/suid_profile** instead of the **ENV** file. This mode is on whenever the effective user ID (group ID) is not equal to the real user ID (group ID). Turning this off causes the effective user ID and group ID to be set to the real user ID and group ID.

-s Sort the positional parameters.

-t Exit after reading and executing one command.

-u Treat unset parameters as an error when substituting.

-v Print shell input lines as they are read.

-x Print commands and their arguments as they are executed.

- Turn off **-x** and **-v** options and stop examining arguments for options.

-- Do not change any of the options; useful in setting parameter **1** to a value beginning with **-**. If no arguments follow this option, the positional parameters are unset.

% **shift** [*n*]

Rename the positional parameters from *n*+**1** ... to **1** .... The default value of *n* is **1**. *n* can be any arithmetic expression that evaluates to a nonnegative number less than or equal to **$#**.

test [*expr*]

Evaluate conditional expression *expr*. See *test*(1) for usage and description. The arithmetic comparison operators are not restricted to integers. They allow any arithmetic expression. The following additional primitive expressions are allowed:

-L *file*	True if *file* is a symbolic link.
-e *file*	True if *file* exists.
file1 **-nt** *file2*	True if *file1* is newer than *file2*.
file1 **-ot** *file2*	True if *file1* is older than *file2*.
file1 **-ef** *file2*	True if *file1* has the same device and i-node number as *file2*.

% **times**

Print the accumulated user and system times for the shell and for processes run from the shell.

% **trap** [*arg*] [*sig*]...

Set *arg* as a command that is read and executed when the shell receives a *sig* signal. (Note that *arg* is scanned once when the trap is set and once when the trap is taken.) Each *sig* can be given as the number or name of a signal. Letter case is ignored. For example, **3**, **QUIT**, **quit**, and **SIGQUIT** all specify the same signal. Use **kill -l** to get a list of signals.

Trap commands are executed in signal number order. Any attempt to set a trap on a signal that was ignored upon entering the current shell is ineffective. Traps remain in effect for a given shell until explicitly changed with another **trap** command; that is, a trap set within a function will remain in effect even after the function returns.

If *arg* is **-** (or if *arg* is omitted and the first *sig* is numeric), reset all traps for each *sig* to their original values.

If *arg* is the null string (**''** or **""**), each *sig* is ignored by the shell and by the commands it invokes.

If *sig* is **DEBUG**, then *arg* is executed after each command. If *sig* is **ERR**, *arg* is executed whenever a command has a nonzero exit code. If *sig* is **0** or **EXIT,** the command *arg* is executed on exit from the shell.

With no arguments, print a list of commands associated with each signal name.

& **typeset** [{**-**|**+**}**LRZfilrtux**[*n*]] [*name*[**=***value*]]...
*name***=***value* [*name***=***value*]...

Assign types and a value to a local named parameter *name*. See also the **export** special command. Parameter assignments remain in effect after the command completes. When invoked inside a function, create a new instance of the parameter *name*. The parameter value and type are restored when the function completes.

The following list of attributes can be specified. Use **+** instead of **-** to turn the options off.

-L Left justify and remove leading blanks from *value*. If *n* is nonzero, it defines the width of the field; otherwise, it is determined by the width of the value of first assignment. When *name* is assigned, the value is filled on the right with blanks or truncated, if necessary, to fit into the field. Leading zeros are removed if the **-Z** option is also set. The **-R** option is turned off. Flagged as **leftjust** *n*.

-R Right justify and fill with leading blanks. If *n* is nonzero, it defines the width of the field; otherwise, it is determined by the width of the value of first assignment. The field is left-filled with blanks or truncated from the end if the parameter is reassigned. The **-L** option is turned off. Flagged as **rightjust** *n*.

-Z Right justify and fill with leading zeros if the first nonblank character is a digit and the **-L** option has not been set. If *n* is nonzero it defines the width of the field; otherwise, it is determined by the width of the value of first assignment. Flagged as **zerofill** *n* plus the flag for **-L** or **-R**.

-f Cause *name* to refer to function names rather than parameter names. No assignments can be made to the *name* declared with the **typeset** statement. The only other valid options are **-t** (which turns on execution tracing for this function) and **-x** (which allows the function to remain in effect across shell procedures executed in the same process environment). Flagged as **function**.

S

-i Parameter is an integer. This makes arithmetic faster. If *n* is nonzero it defines the output arithmetic base; otherwise, the first assignment determines the output base. Flagged as **integer** [**base** *n*].

-l Convert all uppercase characters to lowercase. The uppercase **-u** option is turned off. Flagged as **lowercase**.

-r Mark any given *name* as "read only". The name cannot be changed by subsequent assignment. Flagged as **readonly**.

-t Tag the named parameters. Tags are user-definable and have no special meaning to the shell. Flagged as **tagged**.

-u Convert all lowercase characters to uppercase characters. The lowercase **-l** option is turned off. Flagged as **uppercase**.

-x Mark any given *name* for automatic export to the environment of subsequently executed commands. Flagged as **export**.

typeset alone displays a list of parameter names, prefixed by any flags specified above.

typeset - displays the parameter names followed by their values. Specify one or more of the option letters to restrict the list. Some options are incompatible with others.

typeset + displays the parameter names alone. Specify one or more of the option letters to restrict the list. Some options are incompatible with others.

ulimit [**-HSacdfnst**] [*limit*]

Set or display a resource limit. The limit for a specified resource is set when *limit* is specified. The value of *limit* can be a number in the unit specified with each resource, or the keyword **unlimited**.

The **-H** and **-S** flags specify whether the hard limit or the soft limit is set for the given resource. A hard limit cannot be increased once it is set. A soft limit can be increased up to the hard limit. If neither **-H** nor **-S** is specified, the limit applies to both. The current resource limit is printed when *limit* is omitted. In this case, the soft limit is printed unless **-H** is specified. When more than one resource is specified, the limit name and unit are printed before the value.

If no option is given, **-f** is assumed.

-a	List all of the current resource limits.
-c	The number of 512-byte blocks in the size of core dumps.
-d	The number of kilobytes in the size of the data area.
-f	The number of 512-byte blocks in files written by child processes (files of any size can be read).
-n	The number of file descriptors.
-s	The number of kilobytes in the size of the stack area.
-t	The number of seconds to be used by each process.

umask [**-S**] [*mask*]

Set the user file-creation mask *mask*. *mask* can be either an octal number or a symbolic value as described in *umask*(1). A symbolic value shows permissions that are unmasked. An octal value shows permissions that are masked off.

Without *mask*, print the current value of the mask. With **-S**, print the value in symbolic format. Without **-S**, print the value as an octal number. The output from either form can be used as the *mask* of a subsequent invocation of **umask**.

unalias *name* ...
unalias -a

Remove each *name* from the alias list. With **-a**, remove all **alias** definitions from the current shell execution environment.

% **unset** [**-fv**] *name* ...

Remove the named shell parameters from the parameter list. Their values and attributes are erased. Read-only variables cannot be unset. With **-f**, *name*s refer to function names. With **-v**, *name*s refer to variable names. Unsetting _, **ERRNO**, **LINENO**, **MAILCHECK**, **OPTARG**, **OPTIND**, **RANDOM**, **SECONDS**, and **TMOUT** removes their special meaning, even if they are subsequently assigned to.

wait [*job*]

Wait for the specified *job* to terminate or stop, and report its status. This status becomes the return code for the **wait** command. Without *job*, wait for all currently active child processes to terminate or stop. The termination status returned is that of the last process. See the "Jobs" subsection for a description of the format of *job*.

whence [**-pv**] *name* ...

For each *name*, indicate how it would be interpreted if used as a command name. With **-v**, produce a more verbose report. With **-p** do a path search for *name*, disregarding any use as an alias, a function, or a reserved word.

Comments

A **word** beginning with **#** causes that word and all the following characters up to a newline to be ignored.

Aliasing

The first word of each command is replaced by the text of an **alias**, if an **alias** for this word has been defined. An **alias** name consists of any number of characters excluding metacharacters, quoting characters, file expansion characters, parameter and command substitution characters, and =. The replacement string can contain any valid shell script, including the metacharacters listed above. The first word of each command in the replaced text, other than any that are in the process of being replaced, will be tested for additional aliases. If the last character of the alias value is a **blank**, the word following the alias is also checked for alias substitution. Aliases can be used to redefine special commands, but cannot be used to redefine the keywords listed in the "Compound Commands" subsection. Aliases can be created, listed, and exported with the **alias** command and can be removed with the **unalias** command. Exported aliases remain in effect for subshells but must be reinitialized for separate invocations of the shell (see the "Shell Invocation" subsection).

Aliasing is performed when scripts are read, not while they are executed. Therefore, for it to take effect, an **alias** must be executed before the command referring to the alias is read.

Aliases are frequently used as a shorthand for full path names. An option to the aliasing facility allows the value of the alias to be automatically set to the full path name of the corresponding command. These aliases are called **tracked** aliases. The value of a tracked alias is defined the first time the identifier is read and becomes undefined each time the **PATH** variable is reset. These aliases remain tracked so that the next reference will redefine the value. Several tracked aliases are compiled into the shell. The **-h** option of the **set** command converts each command name that is an **identifier** into a tracked alias.

The following **exported aliases** are compiled into the shell but can be unset or redefined:

```
autoload='typeset -fu'
command='command '
functions='typeset -f'
history='fc -l'
integer='typeset -i'
local=typeset
nohup='nohup '
r='fc -e -'
stop='kill -STOP'
suspend='kill -STOP $$'
type='whence -v'
```

Tilde Substitution

After alias substitution is performed, each word is checked to see if it begins with an unquoted tilde (~). If it does, the word up to a **/** is checked to see if it matches a user name in the **/etc/passwd** file. If a match is found, the ~ and the matched login name are replaced by the login directory of the matched user. If no match is found, the original text is left unchanged. A ~ alone or before a **/** is replaced by the value of the **HOME** parameter. A ~ followed by a **+** or **-** is replaced by the value of the parameter **PWD** and **OLDPWD**, respectively. In addition, tilde substitution is attempted when the value of a parameter assignment begins with a ~.

S

Command Substitution

The standard output from a command enclosed in parenthesis preceded by a dollar sign (**$(**...**)**) or a pair of grave accents (**‘**...**‘**) can be used as part or all of a word; trailing newlines are removed. In the second (archaic) form, the string between the accents is processed for special quoting characters before the command is executed. See the "Quoting" subsection. The command substitution **$(cat file)** can be replaced by the equivalent but faster **$(<file)**. Command substitution of most special commands that do not perform input/output redirection are carried out without creating a separate process.

An arithmetic expression enclosed in double parenthesis preceded by a dollar sign (**$((**...**))**) is replaced by the value of the arithmetic expression within the double parenthesis. See the "Arithmetic Evaluation" subsection for a description of arithmetic expressions.

Parameter Substitution

A **parameter** is an identifier, one or more decimal digits, or one of the characters **!**, **#**, **$**, *****, **-**, **?**, **@**, and _. A **named parameter** (a parameter denoted by an identifier) has a value and zero or more attributes. Named parameters can be assigned values and attributes with the **typeset** special command. Exported parameters pass values and attributes to the environment.

The shell supports a limited one-dimensional array facility. An element of an array parameter is referenced by a subscript. A subscript is denoted by a **[**, followed by an arithmetic expression, followed by a **]**. See the "Arithmetic Evaluation" subsection. To assign values to an array, use **set -A** *name value* .... The value of all subscripts must be in the range of **0** through **1023**. Arrays need not be declared. Any reference to a named parameter with a valid subscript is legal and an array is created if necessary. Referencing an array parameter without a subscript is equivalent to referencing the first element.

If the **-i** integer attribute is set for *name*, the *value* is subject to arithmetic evaluation.

Positional parameters, parameters denoted by a number, can be assigned values with the **set** special command. Parameter **0** is set from argument zero when the shell is invoked.

Use the prefix character **$** to specify the value of a parameter for substitution.

$*parameter*
${*parameter***}**
${*parameter***[***subscript***]}**
: Substitute the value of the parameter, if any. Braces are required when *parameter* is followed by a letter, digit, or underscore that should not be interpreted as part of its name or when a named parameter is subscripted. If *parameter* is one or more digits, it is a positional parameter. A positional parameter of more than one digit must be enclosed in braces. The shell reads all the characters from **${** to the matching **}** as part of the same word, even if it contains braces or metacharacters.

  If *parameter* is ***** or **@**, all the positional parameters, starting with **1**, are substituted (separated by a field separator character). See the "Quoting" subsection.

  If an array parameter with subscript ***** or **@** is used, the value for each element is substituted (separated by a field separator character).

${#*parameter***}**
: If *parameter* is ***** or **@**, the number of positional parameters is substituted. Otherwise, the length of the value of the *parameter* is substituted.

${#*parameter***[*]}**
: Substitute the number of elements in the array.

${*parameter***:-***word***}**
: If *parameter* is set and is nonnull, substitute its value; otherwise, substitute *word*.

${*parameter***:=***word***}**
: If *parameter* is not set or is null, set it to *word*; then substitute the value of the parameter. Positional parameters may not be assigned in this way.

${*parameter***:?***word***}**
: If *parameter* is set and is nonnull, substitute its value; otherwise, print *word* and exit from the shell. If *word* is omitted, a standard message is printed.

${*parameter***:+***word***}**
: If *parameter* is set and is nonnull, substitute *word*; otherwise, substitute nothing.

${*parameter***#***pattern***}**
${*parameter***##***pattern***}**
: If the shell *pattern* matches the beginning of the value of *parameter*, the value of this substitution is the value of the *parameter* with the matched portion deleted; otherwise, the value of this *parameter* is substituted. In the former case, the smallest matching pattern is deleted; in the latter case, the largest matching pattern is deleted.

${*parameter***%***pattern***}**
${*parameter***%%***pattern***}**
: If the shell *pattern* matches the end of the value of *parameter*, the value of *parameter* with the matched part is deleted; otherwise, substitute the value of *parameter*. In the former, the smallest matching pattern is deleted; in the latter, the largest matching pattern is deleted.

In the above, *word* is not evaluated unless it is used as the substituted string. Thus, in the following example, **pwd** is executed only if **d** is not set or is null:

```
echo  ${d:-$(pwd)}
```

If the colon (**:**) is omitted from the above expressions, the shell only checks to determine whether or not *parameter* is set.

• The following parameters are set automatically by the shell:

0	The string used to call the command or script, set from invocation argument zero.
1, **2**, ...	The positional parameters.

S

*****, **@**	All the set positional parameters, separated by a field separator character. See the "Quoting" subsection.
#	The number of set positional parameters in decimal.
-	Flags supplied to the shell on invocation or by the **set** command.
?	The decimal exit status returned by the last executed command.
$	The process number of this shell.
_	Initially, the absolute path name of the shell or script being executed, as passed in the environment. Subsequently, it is assigned the last argument of the previous command. This parameter is not set for commands which are asynchronous. This parameter is also used to hold the name of the matching **MAIL** file when checking for mail.
!	The process number of the last background command invoked.
ERRNO	The value of **errno** as set by the most recently failed system call. This value is system-dependent and is intended for debugging purposes.
LINENO	The line number of the current line within the script or function being executed.
OLDPWD	The previous working directory set by the **cd** command.
OPTARG	The value of the last option argument processed by the **getopts** special command.
OPTERR	If set to 0, **OPTERR** will suppress error messages from the **getopts** special command. **OPTERR** is initially set to 1.
OPTIND	The index of the last option argument processed by the **getopts** special command.
PPID	The process number of the parent of the shell.
PWD	The present working directory set by the **cd** command.
RANDOM	Each time this parameter is evaluated, a random integer, uniformly distributed between 0 and 32767, is generated. The sequence of random numbers can be initialized by assigning a numeric value to **RANDOM**.
REPLY	Set by the **select** compound command, and by the **read** special command when no *name* is supplied.
SECONDS	Each time this parameter is referenced, the number of seconds since shell invocation is returned. If this parameter is assigned a value, the value returned upon reference is the value that was assigned plus the number of seconds since the assignment.

• The following parameters are used by the shell:

CDPATH	The search path for the **cd** command, a list of directories separated by colons.
COLUMNS	If this variable is set, its value is used to define the width of the edit window for the shell edit modes and for printing **select** lists.
EDITOR	If the value of this variable ends in **emacs**, **gmacs**, or **vi** and the **VISUAL** variable is not set, the corresponding option is turned on (see the **set** special command.
ENV	If this parameter is set, parameter substitution is performed on the value to generate the path name of the script to be executed when the shell is invoked (see the "Invocation" subsection). This file is typically used for **alias** and **function** definitions.
FCEDIT	The default editor name for the **fc** command.
FPATH	The search path for function definitions, a list of directories separated by colons. This path is searched when a function with the **-u** attribute is referenced and when a command is not found. If an executable file is found, then it is read and executed in the current environment.
IFS	Internal field separators, normally space, tab, and newline, that are used to separate command words resulting from command or parameter substitution and for separating words with the special command **read**. The first character of the **IFS** parameter is used to separate arguments for the **$*** substitution (see the "Quoting" subsection). If the value of **IFS** is space, tab, and newline, or if **IFS** is unset and it is being used to separate the results of command or parameter substitution, any sequence of **IFS** characters serves to delimit words; otherwise, each occurrence of a character in **IFS** serves to delimit a word.

S

If the value of **IFS** is null, no word splitting is done.

HISTFILE	If this parameter is set when the shell is invoked, its value is the path name of the file that is used to store the command history. The default value is **$HOME/.sh_history**. If the user is a superuser and no **HISTFILE** is given, then no history file is used. See the "Command Reentry" subsection and the WARNINGS section.
HISTSIZE	If this parameter is set when the shell is invoked, the number of previously entered commands accessible to this shell will be greater than or equal to this number. The default is 128.
HOME	The default argument (home directory) for the **cd** command.
LANG	The locale of your system, which is made up of three parts: language, territory, and code set. The default is the **C** locale. See *environ*(5).
LC_ALL	The overriding value for **LANG** and the **LC_*** variables. See *environ*(5).
LC_COLLATE	The collating sequence to use when sorting names and when character ranges occur in patterns. See *environ*(5).
LC_CTYPE	The character classification information to use. Changing the value of **LC_CTYPE** after the shell has started does not affect the lexical processing of shell commands in the current shell execution environment or its subshells. See *environ*(5).
LC_MESSAGES	The language in which system messages appear, and the language that the system expects for user input of **yes** and **no** strings. See *environ*(5).
LC_MONETARY	The currency symbol and monetary value format. See *environ*(5).
LC_NUMERIC	The numeric format. See *environ*(5).
LC_TIME	The date and time format. See *environ*(5).
LINES	If this variable is set, the value is used to determine the column length for printing **select** lists. **select** lists print vertically until about two-thirds of **LINES** lines are filled.
MAIL	If this parameter is set to the name of a mail file and the **MAILPATH** parameter is not set, the shell informs the user of arrival of mail in the specified file.
MAILCHECK	How often (in seconds) the shell checks for changes in the modification time of any of the files specified by the **MAILPATH** or **MAIL** parameters. The default value is 600 seconds. When the time has elapsed, the shell checks before issuing the next prompt.
MAILPATH	A list of file names separated by colons. If this parameter is set, the shell informs the user of any modifications to the specified files that have occurred within the last **MAILCHECK** seconds. Each file name can be followed by a **?** and a message to be printed, in which case the message will undergo parameter and command substitution with the parameter **$_** defined as the name of the changed file. The default message is **you have mail in $_**.
NLSPATH	The search path for message catalogs, a list of directories separated by colons.
PATH	The search path for commands, a list of directories separated by colons. See the "Execution" subsection.
PS1	The value of this parameter is expanded for parameter substitution, to define the primary prompt string. The default value is "**$** ". The character **!** in the primary prompt string is replaced by the command number. See the "Command Reentry" subsection.
PS2	Secondary prompt string for command completion. The default value is "**>** ".
PS3	Selection prompt string used within a **select** loop. If unset, it defaults to "**#?** ".
PS4	Execution trace string that precedes each line of an execution trace. See the **set -x** special command. If unset, it defaults to "**+** ".
SHELL	The path name of the shell is kept in the environment. When invoked, the shell is restricted if the value of this variable contains an **r** in the base name.

TMOUT	If set to a value greater than zero, the shell will terminate if a command is not entered within the prescribed number of seconds after issuing the **PS1** prompt. (Note that the shell can be compiled with a maximum bound for this value which cannot be exceeded.)
VISUAL	Invokes the corresponding option when the value of this variable ends in **emacs**, **gmacs**, or **vi**. See the **set -o** special command.

The shell gives default values to **IFS**, **MAILCHECK**, **PATH**, **PS1**, **PS2**, and **TMOUT**. On the other hand, **MAIL**, **ENV**, **HOME**, and **SHELL** are never set automatically by the shell (although **HOME**, **MAIL**, and **SHELL** are set by **login**; see *login*(1)).

Blank Interpretation

After parameter and command substitution, the results of substitution are scanned for field separator characters (defined in **IFS**), and split into distinct arguments when such characters are found. **sh** retains explicit null arguments (**""** or **''**) but removes implicit null arguments (those resulting from parameters that have null values).

File Name Generation

Following substitution, each command *word* is processed as a pattern for file name expansion unless expansion has been disabled with the **set -f** special command. The form of the patterns is the Pattern Matching Notation defined in *regexp*(5). The word is replaced with sorted file names matching the pattern. If no file name is found that matches the pattern, the word is left unchanged.

In addition to the notation described in *regexp*(5), **sh** recognizes composite patterns made up of one or more pattern lists separated from each other with a **|**. Composite patterns can be formed with one or more of the following:

?(*pattern-list***)**	Matches any one of the given patterns.
(pattern-list***)**	Matches zero or more occurrences of the given patterns.
+(*pattern-list***)**	Matches one or more occurrences of the given patterns.
@(*pattern-list***)**	Matches exactly one of the given patterns.
!(*pattern-list***)**	Matches anything, except one of the given patterns.

Quoting

Each of the metacharacters (see the "Definitions" subsection) has a special meaning to the shell and terminates a word unless quoted. A character may be **quoted** (that is, made to stand for itself) by preceding it with a backslash (\). The pair \newline is ignored; the current and following lines are concatenated.

All characters enclosed between a pair of apostrophes (**'**...**'**) are quoted. An apostrophe cannot appear within apostrophes.

Parameter and command substitution occurs inside quotation marks (**"**...**"**). \ quotes the characters \, **‘**, **"**, and **$**.

Inside grave accent marks (**‘**...**‘**), \ quotes the characters \, **‘**, and **$**. If the grave accents occur within quotation marks, \ also quotes the character **"**.

The meanings of **$*** and **$@** are identical when not quoted or when used as a parameter assignment value or as a file name. However, when used as a command argument, **"$*"** is equivalent to **"$1***d***$2***d*...**"**, whereas **"$@"** is equivalent to **"$1"***d***"$2"***d*... (where *d* is the first character of **IFS**),

The special meaning of keywords or aliases can be removed by quoting any character of the name. The recognition of function names or special command names cannot be altered by quoting them.

Arithmetic Evaluation

Integer arithmetic is provided with the special command **let**. Evaluations are performed using long integer arithmetic. Constants take the form *base***#***n* or *n*, where *base* is a decimal number between two and thirty-six representing the arithmetic base and *n* is a number in that base. If *base***#** is omitted, base 10 is used.

An arithmetic expression uses the same syntax, precedence, and associativity of expression as the C language. All the integral operators, other than **++**, **--**, **?:**, and **,** are supported. Variables can be referenced by name within an arithmetic expression without using the parameter substitution syntax. When a variable is referenced, its value is evaluated as an arithmetic expression.

A variable can be typed as an integer with the **-i** option of the **typeset** special command, as in **typeset -i**[*base*] *name*. Arithmetic evaluation is performed on the value of each assignment to a variable with the **-i** attribute. If you do not specify an arithmetic base, the first assignment to the variable

S

determines the arithmetic base. This base is used when parameter substitution occurs.

Since many of the arithmetic operators require quoting, an alternative form of the **let** command is provided. For any command beginning with **((**, all characters until the matching **))** are treated as a quoted expression. More precisely, **((**...**))** is equivalent to **let "**...**"**.

Prompting

When used interactively, the shell prompts with the value of **PS1** before reading a command. Whenever a newline is received and further input is needed to complete a command, the secondary prompt (the value of **PS2**) is issued.

Conditional Expressions

A **conditional expression** is used with the **[[** compound command to test attributes of files and to compare strings. Word splitting and file name generation are not performed on the words between **[[** and **]]**. Each expression can be constructed from one or more of the following unary or binary expressions:

-a *file*	True, if *file* exists.
-b *file*	True, if *file* exists and is a block special file.
-c *file*	True, if *file* exists and is a character special file.
-d *file*	True, if *file* exists and is a directory.
-e *file*	True, if *file* exists.
-f *file*	True, if *file* exists and is an ordinary file.
-g *file*	True, if *file* exists and has its setgid bit set.
-h *file*	True, if *file* exists and is a symbolic link.
-k *file*	True, if *file* exists and has its sticky bit set.
-n *string*	True, if length of *string* is nonzero.
-o *option*	True, if the set option named *option* is on.
-p *file*	True, if *file* exists and is a fifo special file or a pipe.
-r *file*	True, if *file* exists and is readable by current process.
-s *file*	True, if *file* exists and has a size greater than zero.
-t *fildes*	True, if file descriptor number *fildes* is open and is associated with a terminal device.
-u *file*	True, if *file* exists and has its setuid bit set.
-w *file*	True, if *file* exists and is writable by the current process.
-x *file*	True, if *file* exists and is executable by the current process. If *file* exists and is a directory, then the current process has permission to search in the directory.
-z *string*	True, if length of *string* is zero.
-L *file*	True, if *file* exists and is a symbolic link.
-O *file*	True, if *file* exists and is owned by the effective user ID of this process.
-G *file*	True, if *file* exists and its group matches the effective group ID of this process.
-S *file*	True, if *file* exists and is a socket.
file1 **-nt** *file2*	True, if *file1* exists and is newer than *file2*.
file1 **-ot** *file2*	True, if *file1* exists and is older than *file2*.
file1 **-ef** *file2*	True, if *file1* and *file2* exist and refer to the same file.
string **=** *pattern*	True, if *string* matches *pattern*.
string **!=** *pattern*	True, if *string* does not match *pattern*.
string **<** *string2*	True, if *string1* comes before *string2* based on the ASCII value of their characters.
string **>** *string2*	True, if *string1* comes after *string2* based on the ASCII value of their characters.
exp1 **-eq** *exp2*	True, if *exp1* is equal to *exp2*.
exp1 **-ne** *exp2*	True, if *exp1* is not equal to *exp2*.
exp1 **-lt** *exp2*	True, if *exp1* is less than *exp2*.
exp1 **-gt** *exp2*	True, if *exp1* is greater than *exp2*.
exp1 **-le** *exp2*	True, if *exp1* is less than or equal to *exp2*.
exp1 **-ge** *exp2*	True, if *exp1* is greater than or equal to *exp2*.

A compound expression can be constructed from these primitives by using any of the following, listed in decreasing order of precedence.

(*exp***)**	True, if *exp* is true. Used to group expressions.
!*exp*	True, if *exp* is false.
exp1 **&&** *exp2*	True, if *exp1* and *exp2* are both true.
exp1 **||** *exp2*	True, if either *exp1* or *exp2* is true.

S

Input/Output

Before a command is executed, its input and output can be redirected using a special notation interpreted by the shell. The following can appear anywhere in a simple-command or may precede or follow a command and are not passed on to the invoked command. Command and parameter substitution occurs before *word* or *digit* is used, except as noted below. File name generation occurs only if the pattern matches a single file and blank interpretation is not performed.

<*word*	Use file *word* as standard input (file descriptor **0**).
>*word*	Use file *word* as standard output (file descriptor **1**). If the file does not exist, it is created. If the file exists, and the **noclobber** option is on, an error occurs; otherwise, the file is truncated to zero length.
>| *word*	Same as **>**, except that it overrides the **noclobber** option.
>>*word*	Use file *word* as standard output. If the file exists, output is appended to it (by first searching for the end-of-file); otherwise, the file is created.
<>*word*	Open file *word* for reading and writing as standard input.
<<[**-**]*word*	The shell input is read up to a line that matches *word*, or to an end-of-file. No parameter substitution, command substitution or file name generation is performed on *word*. The resulting document, called a *here-document*, becomes the standard input. If any character of *word* is quoted, no interpretation is placed upon the characters of the document. Otherwise, parameter and command substitution occurs, \newline is ignored, and \ must be used to quote the characters \, **$**, **'**, and the first character of *word*. If **-** is appended to **<<**, all leading tabs are stripped from *word* and from the document.
<&*digit*	The standard input is duplicated from file descriptor *digit* (see *dup*(2)).
>&*digit*	The standard output is duplicated to file descriptor *digit* (see *dup*(2)).
<&-	The standard input is closed.
>&-	The standard output is closed.
<&p	The input from the coprocess is moved to standard input.
>&p	The output to the coprocess is moved to standard output.

If any of the above redirections is preceded by a digit (**0** to **9**), the file descriptor used is the one specified by the digit, instead of the default **0** (standard input) or **1** (standard output). For example:

2>&1

means open file descriptor 2 for writing as a duplicate of file descriptor 1. Output directed to file descriptor 2 is written in the same location as output to file descriptor 1.

Order is significant in redirection. The shell evaluates each redirection in terms of the (*file descriptor*, *file*) assignment at the time of evaluation. For example:

1>*fname* **2>&1**

first assigns file descriptor 1 to file *fname*. It then assigns file descriptor 2 to the file assigned to file descriptor 1 (that is, *fname*).

If the order of redirection is reversed, as in

2>&1 1>*fname*

file descriptor 2 is assigned to the file assigned to file descriptor 1 (probably the terminal) and then file descriptor 1 is assigned to file *fname*.

By using the redirection operators above, the input and output of a *coprocess* may be moved to a numbered file descriptor, allowing other commands to write to them and read from them. If the input of the current coprocess is moved to a numbered file descriptor, another coprocess may be started.

If a command is followed by **&** and job control is inactive, the default standard input for the command is the empty file **/dev/null**. Otherwise, the environment for the execution of a command contains the file descriptors of the invoking shell as modified by input/output specifications.

Environment

The **environment** (see *environ*(5)) is a list of name-value pairs passed to an executed program much like a normal argument list. The names must be identifiers and the values are character strings. The shell interacts with the environment in several ways. When invoked, the shell scans the environment and creates a parameter for each name found, gives it the corresponding value and marks it **export**. Executed commands inherit the environment. If the user modifies the values of these parameters or creates new ones by using the **export** or **typeset -x** special commands, the values become part of the environment. The environment seen by any executed command is thus composed of any name-value pairs originally inherited by the shell, whose values may be modified by the current shell, plus any additions which must be noted in **export** or **typeset -x** commands.

The environment for any simple command or function can be augmented by prefixing it with one or more parameter assignments. A parameter assignment argument takes the form *identifier*=*value*. For example, both the following

```
TERM=450 cmd args
(export TERM; TERM=450; cmd args)
```

are equivalent (as far as the execution of *cmd* is concerned, except for the special commands that are preceded by a percent sign (%).

If the **-k** option is set, all parameter assignment arguments are placed in the environment, even if they occur after the command name. The following echo statement prints **a=b c**. After the **-k** option is set, the second echo statement prints only **c**:

```
echo a=b c → a=b c
set -k
echo a=b c → c
```

This feature is intended for use with scripts written for early versions of the shell and its use in new scripts is strongly discouraged. It is likely to disappear someday.

Functions

The **function** command (described in the "Compound Commands" subsection) defines shell functions. Shell functions are read and stored internally. Alias names are resolved when the function is read. Functions are executed like commands, with the arguments passed as positional parameters. (See the "Execution" subsection.)

Functions execute in the same process as the caller and share all files and current working directory with the caller. Traps defined by the caller remain in effect within the function until another **trap** command is executed. Traps set within a function remain in effect after the function returns. Ordinarily, variables are shared between the calling program and the function. However, the **typeset** special command can be used within a function to define local variables whose scope includes the current function and all functions it calls.

The **return** special command is used to return from function calls. Errors within functions return control to the caller.

Function identifiers can be listed with the **+f** option of the **typeset** special command. Function identifiers and the associated text of the functions can be listed with the **-f** option. Functions can be undefined with the **-f** option of the **unset** special command.

Ordinarily, functions are unset when the shell executes a shell script. The **-xf** option of the **typeset** command allows a function to be exported to scripts that are executed without reinvoking the shell. Functions that must be defined across separate invocations of the shell should be placed in the **ENV** file.

Jobs

If the **monitor** option of the **set** command is turned on, an interactive shell associates a *job* with each pipeline. It keeps a table of current jobs, printed by the **jobs** command, and assigns them small integer numbers. When a job is started asynchronously with **&**, the shell prints a line that looks like:

```
[1] 1234
```

indicating that job number 1 was started asynchronously and had one (top-level) process whose process ID was 1234.

If you are running a job and wish to do something else, you can type the suspend character (the **susp** character defined with **stty**; see *stty*(1)) to send a **SIGSTOP** signal to the current job. The shell then indicates that the job has been **Stopped**, and prints another prompt. Then you can manipulate the state of this job by putting it in the background with the **bg** command, running other commands, and eventually returning the job to the foreground with the **fg** command. A suspend takes effect immediately and resembles an interrupt, since pending output and unread input are discarded when the suspend is entered.

A job running in the background stops if it tries to read from the terminal. Background jobs normally are allowed to produce output, but can be disabled with the **stty tostop** command. If the user sets this terminal option, background jobs stop when trying to produce output.

There are several ways to refer to jobs in the shell. A job can be referred to by the process ID of any process in the job or by one of the following:

%*number*	The job with the given number.
%*string*	Any job whose command line begins with *string*.
%?*string*	Any job whose command line contains *string*.
%%	Current job.
%+	Equivalent to **%%**.
%-	Previous job.

The shell learns immediately when a process changes state. It informs the user when a job is blocked and prevented from further progress, but only just before it prints a prompt.

When the monitor mode is on, each background job that completes triggers any trap set for **SIGCHLD**.

If you try to exit from shell while jobs are stopped, you are warned with the message **You have stopped jobs.** You can use the **jobs** command to identify them. If you immediately try to exit again, the shell will not warn you a second time, and the stopped jobs will be terminated.

If you try to leave the shell while jobs are running, you are not warned. The shell exits silently and sets the parent of the running jobs to the **init** process (number 1).

Signals

The **SIGINT** and **SIGQUIT** signals for an invoked command are ignored if the command is followed by **&** and the **monitor** option is off. Otherwise, signals have the values inherited by the shell from its parent, with the exception of signal **SIGSEGV** (but see also the **trap** special command).

Execution

Substitutions are made each time a command is executed. **sh** checks the command name to determine whether it matches a special command. If it does, it is executed within the current shell process.

Next, **sh** checks the command name to determine whether it matches one of the user-defined functions. If it does, **sh** saves the positional parameters, then sets them to the arguments of the function call. The positional parameter **0** is unchanged. When the function completes or issues a **return**, **sh** restores the positional parameter list. The value of a function is the value of the last command executed. A function is executed in the current shell process.

S

If a command name is not a user-defined function or a special command, **sh** creates a process and attempts to execute the command using an **exec***() system call (see *exec*(2)).

The shell parameter **PATH** defines the search path for the directory containing the command. Alternative directory names are separated by a colon (**:**). The default path is **/usr/bin:** (specifying **/usr/bin**, and the current directory, in that order). Note that the current directory is specified by a null path name, which can appear immediately after the equal sign, between colon delimiters, or at the end of the path list. The search path is not used if the command name contains a **/**. Otherwise, each directory in the path is searched for an executable file. If the file has execute permissions but is not a directory or an executable object code file, it is assumed to be a script file, which is a file of data for an interpreter. If the first two characters of the script file are **#!**, **exec***() expects an interpreter path name to follow. **exec***() then attempts to execute the specified interpreter as a separate process to read the entire script file. If a call to **exec***() fails, **sh** is spawned to interpret the script file. All nonexported aliases, functions, and named parameters are removed in this case. If the shell command file does not have read permission, or if the **setuid** and/or **setgid** bits are set on the file, the shell executes an agent to set up the permissions and execute the shell with the shell command file passed down as an open file. A parenthesized command is also executed in a subshell without removing nonexported quantities.

Command Reentry

The text of the last **HISTSIZE** (default 128) commands entered from a terminal device is saved in a history file. The file **$HOME/.sh_history** is used if the **HISTFILE** variable is not set or writable. A shell can access the commands of all interactive shells that use the same named **HISTFILE**. The special command **fc** is used to list or edit a portion of this file. The portion of the file to be edited or listed can be selected by number or by giving the first character or characters of the command. A single command or range of commands can be specified. If you do not specify an editor program as an argument to **fc**, the value of the parameter **FCEDIT** is used. If **FCEDIT** is not defined, **/usr/bin/ed** is used. The edited command is printed and reexecuted upon leaving the editor. The editor name **-** is used to skip the editing phase and to reexecute the command. In this case, a substitution parameter of the form *old*=*new* can be used to modify the command before execution. For example, if **r** is aliased to **fc -e -**, typing **r bad=good c** reexecutes the most recent command that starts with the letter **c** and replaces the first occurrence of the string **bad** with the string **good**.

Command Line Editing

Normally, each command line typed at a terminal device is followed by a newline or return. If one of the **emacs**, **gmacs**, **vi**, or **viraw**, options is set, you can edit the command line. An editing option is automatically selected each time the **VISUAL** or **EDITOR** variable is assigned a value ending in one of these option names.

The editing features require that the user's terminal accept return without line feed and that a space (" ") must overwrite the current character on the screen. ADM terminal users should set the "space – advance" switch to "space". Hewlett-Packard terminal users should set the straps to "bcGHxZ etX".

The editing modes enable the user to look through a window at the current line. The default window width is 80, unless the value of **COLUMNS** is defined. If the line is longer than the window width minus two, a mark displayed at the end of the window notifies the user. The mark is one of:

>	The line extends to the right.
<	The line extends to the left.
*****	The line extends to both sides of the window.

As the cursor moves and reaches the window boundaries, the window is centered about the cursor.

The search commands in each edit mode provide access to the history file. Only strings are matched, not patterns, although a leading **^** in the string restricts the match to begin at the first character in the line.

emacs Editing Mode

This mode is invoked by either the **emacs** or **gmacs** option. Their sole difference is their handling of **^T**. To edit, the user moves the cursor to the point needing correction and inserts or deletes characters or words. All editing commands are control characters or escape sequences. The notation for control characters is caret (**^**) followed by a character. For example, **^F** is the notation for Control-F. This is entered by holding down the Ctrl (control) key and pressing **f**. The shift key is *not* pressed. The notation **^?** indicates the delete (DEL) key.

The notation for escape sequences is **M-** followed by a character. For example, **M-f** (pronounced *meta f*) is entered by pressing the escape key (ESC) followed by pressing **f**. **M-F** is the notation for escape followed by shift (capital) **F**.

All edit commands operate from any place on the line (not only at the beginning). Neither the return (**^M**) nor the newline (**^J**) key is entered after edit commands, except when noted.

^F	Move cursor forward (right) one character.
M-f	Move cursor forward one word. (The editor's idea of a word is a string of characters consisting of only letters, digits and underscores.)
^B	Move cursor backward (left) one character.
M-b	Move cursor backward one word.
^A	Move cursor to start of line.
^E	Move cursor to end of line.
^]*char*	Move cursor forward to character *char* on current line.
M-^]*char*	Move cursor backward to character *char* on current line.
^X^X	Interchange the cursor and mark.
erase	Delete previous character. (User-defined erase character as defined by the **stty** command, usually **^H** or **#**.)
^D	Delete current character.
eof	Terminate the shell if the current line is null. (User-defined end-of-file character as defined by the **stty** command, usually **^D**.)
M-d	Delete current word.
M-^H	Delete previous word. (meta-backspace)

S

M-h	Delete previous word.
M-^?	Delete previous word. (meta-delete) If your interrupt character is **^?** (DEL, the default), this command will not work.
^T	In **emacs** mode, transpose current character with next character. In **gmacs** mode, transpose two previous characters.
^C	Capitalize current character.
M-c	Capitalize current word.
M-l	Change the current word to lowercase.
^K	Delete from the cursor to the end of the line. If preceded by a numerical parameter whose value is less that the current cursor position, then delete from the given position up to the cursor. If preceded by a numerical parameter whose value is greater than the current cursor position, then delete from the cursor up to the given position.
^W	Kill from the cursor to the mark.
M-p	Push the region from the cursor to the mark on the stack.
kill	Kill the entire current line. If two kill characters are entered in succession, all subsequent consecutive kill characters cause a line feed (useful when using paper terminals). (User-defined kill character as defined by the **stty** command, usually **^X** or **@**.)
^Y	Restore last item removed from line. (Yank item back to the line.)
^L	Line feed and print current line.
^@	Set mark. (null character)
M-	Set mark. (meta-space)
^J	Execute the current line. (newline)
^M	Execute the current line. (return)
^P	Fetch previous command. Each time **^P** is entered, the previous command in the history list is accessed.
^N	Fetch next command. Each time **^N** is entered the next command in the history list is accessed.
M-<	Fetch the least recent (oldest) history line.
M->	Fetch the most recent (youngest) history line.
^R*string*	Reverse search history for a previous command line containing *string*. If a parameter of zero is given, the search is forward. *string* is terminated by a return or newline. If *string* is preceded by a **^**, the matched line must begin with *string*. If *string* is omitted, the next command line containing the most recent *string* is accessed. In this case, a parameter of zero reverses the direction of the search.
^O	Execute the current line and fetch the next line relative to current line from the history file.
M-*digits*	Define a numeric parameter. The digits are taken as a parameter to the next command. The commands that accept a parameter are *erase*, **^B**, **^C**, **^D**, **^F**, **^K**, **^N**, **^P**, **^R**, **^]**, **M-^H**, **M-.**, **M-_**, **M-b**, **M-c**, **M-d**, **M-f**, **M-h**, and **M-l**.
M-*letter*	Your alias list is searched for an alias by the name *_letter* (underscore-letter). If an alias of this name is defined, its value is inserted on the input queue. This *letter* must not be one of the above metafunctions.
M-.	The last word of the previous command is inserted on the line. If preceded by a numeric parameter, the value of this parameter determines which word to insert rather than the last word.
M-_	Same as **M-.**.
M-*	Attempt file name generation on the current word.
M-^[	File name completion. (meta-escape.) Replaces the current word with the longest common prefix of all file names matching the current word with an asterisk appended. If the match is unique, a **/** is appended if the file is a directory and a space is appended if the file is not a directory.
M-=	List files matching current word pattern as if an asterisk were appended.
^U	Multiply parameter of next command by 4.
\	Escape next character. Editing characters and your erase, kill, and interrupt characters may be entered in a command line or in a search string, if preceded by a \. The \ removes the next character's editing features (if any).
^V	Display version of the shell.
M-#	Insert a **#** at the beginning of the line and execute it. This causes a comment to be inserted in the history file.

vi Editing Mode

The editor starts in insert mode until an escape (ESC) is received. This puts you in control mode in which you can move the cursor and perform editing commands. A return in either mode sends the line.

Most control commands accept an optional repeat *count* prior to the command.

In **vi** mode on most systems, canonical processing is initially enabled and the command is echoed again if the speed is 1200 baud or greater and contains any control characters, or if less than one second has elapsed since the prompt was printed. The escape (ESC) character terminates canonical processing for the remainder of the command and you can then modify the command line. This scheme has the advantages of canonical processing with the typeahead echoing of raw mode.

Setting the **viraw** option always disables canonical processing on the terminal. This mode is implicit for systems that do not support two alternate end-of-line delimiters, and may be helpful for certain terminals.

Insert Edit Commands

By default, the editor is in insert mode.

erase	Delete previous inserted character. The *erase* character is user-definable with the **stty** command, usually set to **^H**. The system default is **#**.
kill	Delete all current inserted characters. The *kill* character is user-definable with the **stty** command, usually set to **^X** or **^U**. The system default is **@**.
\	Escape the next *erase* or *kill* character.
^D	Terminate the shell.
^V	Escape next character. Editing characters and erase or kill characters may be entered in a command line or in a search string if preceded by a **^V**, which removes the next character's editing features (if any).
^W	Delete the previous blank-separated word.

Motion Edit Commands

These commands move the cursor. The use of *count* causes a repetition of the command the cited number of times.

[*count*]**l**	Cursor forward (right) one character.
[*count*]**w**	Cursor forward one alphanumeric word.
[*count*]**W**	Cursor forward to the beginning of the next word that follows a blank.
[*count*]**e**	Cursor forward to the end of the word.
[*count*]**E**	Cursor forward to end of the current blank-delimited word.
[*count*]**h**	Cursor backward (left) one character.
[*count*]**b**	Cursor backward one word.
[*count*]**B**	Cursor backward to preceding blank-separated word.
[*count*]**|**	Cursor to column *count*. Default is 1.
[*count*]**f***c*	Find the next character *c* in the current line.
[*count*]**F***c*	Find the previous character *c* in the current line.
[*count*]**t***c*	Equivalent to **f***c* followed by **h**.
[*count*]**T***c*	Equivalent to **F***c* followed by **l**.
[*count*]**;**	Repeat the last single-character find command, **f**, **F**, **t**, or **T**.
[*count*]**,**	Reverses the last single character find command.
0	Cursor to start of line.
^	Cursor to first nonblank character in line.
$	Cursor to end of line.

History Search Commands

These commands access your command history file.

[*count*]**k**	Fetch previous command. Each time **k** is entered, the next earlier command in the history list is accessed.
[*count*]**-**	Equivalent to **k**.
[*count*]**j**	Fetch next command. Each time **j** is entered, the next later command in the history list is accessed.
[*count*]**+**	Equivalent to **j**.
[*count*]**G**	The command number *count* is fetched. The default is the first command in the history list.
/*string*	Search backward through history for a previous command containing *string*. *string* is terminated by a return or newline. If *string* is preceded by a **^**, the matched line

S

	must begin with *string*. If *string* is null, the previous string is used.
?*string*	Same as **/**, but search in the forward direction.
n	Search for next match of the last pattern to the **/** or **?** commands.
N	Search for next match of the last pattern to **/** or **?**, but in reverse direction.

Text Modification Edit Commands

These commands will modify the line.

a	Enter insert mode after the current character.
A	Append text to the end of the line. Equivalent to **$a**.
[*count*]**c***motion* **c**[*count*]*motion*	Move cursor forward to the character position specified by *motion*, deleting all characters between the original cursor position and the new position, and enter insert mode. If *motion* is **c**, the entire line is deleted.
C	Delete from the current character through the end of line and enter insert mode. Equivalent to **c$**.
S	Equivalent to **cc**.
[*count*]**d***motion* **d**[*count*]*motion*	Move cursor to the character position specified by *motion*, deleting all characters between the original cursor position and the new position. If *motion* is **d**, the entire line will be deleted.
D	Delete from the current character through the end of line. Equivalent to **d$**.
i	Enter insert mode before the current character.
I	Enter insert mode before the beginning of the line. Equivalent to the two-character sequence **0i**.
[*count*]**P**	Insert the previous text modification before the cursor.
[*count*]**p**	Insert the previous text modification after the cursor.
R	Enter insert mode and replace characters on the screen with characters you type, overlay fashion.
[*count*]**r***c*	Replace the current character with *c*.
[*count*]**x**	Delete the current character.
[*count*]**X**	Delete the preceding character.
[*count*]**.**	Repeat the previous text modification command.
~	Invert the case of the current character and advance the cursor.
[*count*]**_**	Append the *count* word of the previous command at the current cursor location and enter insert mode at the end of the appended text. The last word is used if *count* is omitted.
*****	Append an ***** to the current word and attempt file name generation. If no match is found, ring the bell. If a match is found, replace the word with the matching string of file names and enter insert mode.
escape \	Attempt file name completion on the current word. Replace the current word with the longest common prefix of all file names matching the current word with an asterisk appended. If the match is unique, append a **/** if the file is a directory or append a space if the file is not a directory.

Other Edit Commands

[*count*]**y***motion* **y**[*count*]*motion*	Yank current character through character that *motion* would move the cursor to and put them into the delete buffer. The text and cursor are unchanged.
Y	Yank from current position to end of line. Equivalent to **y$**.
u	Undo the last text-modifying command.
U	Undo all the text-modifying commands performed on the line.
[*count*]**v**	Execute the command **fc -e ${VISUAL:-${EDITOR:-vi}}** *count* in the input buffer. If *count* is omitted, the current line is used. This executes an editor with the current line as the input "file". When you exit from the editor, the result is executed.
^L	Line feed and print current line.
^J	Execute the current line, regardless of mode. (newline)

^M	Execute the current line, regardless of mode. (return)
#	Insert a **#** at the beginning of the current line and after each embedded newline, and execute the line. Useful for inserting the current command line in the history list without executing it.
=	List the file names that match the current word if an asterisk were appended to it.
@*letter*	Search your alias list for an alias with the name _*letter* (underscore letter). If an alias of this name is defined, its value is executed as a command sequence on the current line. This provides a simple macro capability.

EXTERNAL INFLUENCES

Environment Variables

LC_COLLATE determines the collating sequence used in evaluating pattern matching notation for file name generation. If it is not defined or is empty, it defaults to the value of **LANG**.

LC_CTYPE determines the classification of characters as letters, and the characters matched by character class expressions in pattern matching notation. If it is not defined or is empty, it defaults to the value of **LANG**.

If **LANG** is not defined or is empty, it defaults to **C** (see *lang*(5)).

If any internationalization variable contains an invalid value, they all default to **C** (see *environ*(5)).

International Code Set Support

Single- and multibyte character code sets are supported.

RETURN VALUE

Errors detected by the shell, such as syntax errors, cause the shell to return a nonzero exit status. Otherwise, the shell returns the exit status of the last command executed. See also the **exit** special command. If the shell is being used noninteractively, the execution of the shell file is abandoned. Runtime errors detected by the shell are reported by printing the command or function name and the error condition. If the line number on which the error occurred is greater than one, the line number is also printed in brackets (**[]**) after the command or function name.

WARNINGS

Some file descriptors are used internally by the POSIX shell. For HP-UX releases 10.10 and beyond, file descriptors 24 through 30 are reserved. HP-UX releases 10.00 and 10.01 reserve descriptors 54 through 60. Applications using these and forking a subshell should not depend upon them surviving in the subshell or its descendants.

If a command that is a tracked alias is executed, and a command with the same name is installed in a directory in the search path before the directory where the original command was found, the shell will continue to load and execute the original command. Use the **-t** option of the **alias** command to correct this situation.

If you move the current directory or one above it, **pwd** may not give the correct response. Use the **cd** command with a full path name to correct this situation.

Some very old shell scripts use a caret (**^**) as a synonym for the pipe character (**|**). **sh** *does not* recognize the caret as a pipe character.

If a command is piped into a shell command, all variables set in the shell command are lost when the command completes.

Using the **fc** built-in command within a compound command will cause the entire command to disappear from the history file.

The dot (**.**) special command, as in **.** *file*, reads the entire file before any commands are executed. Therefore, **alias** and **unalias** commands in the file will not apply to any functions defined in the file.

Traps are not processed while the shell is waiting for a foreground job. Thus, a trap on **SIGCHLD** is not executed until the foreground job terminates.

The **export** special command does not handle arrays properly. Only the first element of an array is exported to the environment.

Background processes started from a noninteractive shell cannot be accessed with job control commands.

The value of the **IFS** variable in the user's environment affects the behavior of scripts.

Collating Order

In an international environment, character ordering is determined by the value of **LC_COLLATE**, rather than by the binary ordering of character values in the machine collating sequence. This brings with it certain attendant dangers, particularly when using range expressions in file name generation patterns. For example, the command,

```
rm [a-z]*
```

might be expected to match all file names beginning with a lowercase alphabetic character. However, if dictionary ordering is specified by **LC_COLLATE**, it would also match file names beginning with an uppercase character (as well as those beginning with accented letters). Conversely, it would fail to match letters collated after **z** in languages such as Danish or Norwegian.

The correct (and safe) way to match specific character classes in an international environment is to use a pattern (see *regexp*(5)) of the form:

```
rm [[:lower:]]*
```

This uses **LC_CTYPE** to determine character classes and works predictably for all supported languages and codesets. For shell scripts produced on noninternationalized systems (or without consideration for the above dangers), it is recommended that they be executed in a non-NLS environment. This requires that **LANG**, **LC_COLLATE**, and so on, be set to **C** or not set at all.

The history file does not support mixing of locales in the same file. For users of multiple locales, you can assign a unique history file for each locale by setting **HISTFILE** as:

```
HISTFILE=$HOME/.sh_hist_${LANG}
```

AUTHOR

sh was developed by AT&T, OSF, and HP.

FILES

$HOME/.profile	Read to set up user's custom environment
/etc/passwd	To find home directories
/etc/profile	Read to set up system environment
/etc/suid_profile	Security profile
/tmp/sh*	For here-documents
/usr/bin/sh	Shell executable program location

SEE ALSO

cat(1), cd(1), command(1), echo(1), ed(1), env(1), getopts(1), kill(1), ln(1), login(1), newgrp(1), printf(1), pwd(1), read(1), stty(1), test(1), time(1), umask(1), vi(1), dup(2), exec(2), fork(2), pipe(2), stty(2), ulimit(2), umask(2), wait(2), rand(3C), a.out(4), profile(4), environ(5), lang(5), regexp(5), signal(5).

STANDARDS CONFORMANCE

sh: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

.: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2
:: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2
break: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2
case: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2
continue: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2
eval: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2
exec: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2
exit: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2
export: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2
for: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2
if: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2
read: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2
return: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2
set: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2
shift: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2
time: SVID2, SVID3, XPG2, XPG3, XPG4
trap: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2
unset: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

until: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2
while: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

NAME

sh - overview of various system shells

SYNOPSIS

POSIX Shell:

sh [±**aefhikmnoprstuvx**] [±**o** *option*] ... [**-c** *string*] [*arg* ...]

rsh [±**aefhikmnoprstuvx**] [±**o** *option*] ... [**-c** *string*] [*arg* ...]

Bourne Shell:

sh [**- -acefhiknrstuvx** ...] [*arg* ...]

rsh [**- -acefhiknrstuvx** ...] [*arg* ...]

Korn Shell:

ksh [±**aefhikmnoprstuvx**] [±**o** *option*] ... [**-c** *string*] [*arg* ...]

rksh [±**aefhikmnoprstuvx**] [±**o** *option*] ... [**-c** *string*] [*arg* ...]

C Shell:

csh [**-cefinstvxTVX**] [*command_file*] [*argument_list* ...]

Key Shell:

keysh

DESCRIPTION

Remarks:

The POSIX.2 standard requires that, on a POSIX-compliant system, executing the command **sh** activates the POSIX shell (located in file **/usr/bin/sh** on HP-UX systems), and executing the command **man sh** produces an on-line manual entry that displays the syntax of the POSIX shell command-line.

However, the **sh** command has historically been associated with the conventional Bourne shell, which could confuse some users. To meet standards requirements and also clarify the relationships of the various shells and where they reside on the system, this entry provides command-line syntax and a brief description of each shell, and lists the names of the manual entries where each shell is described in greater detail.

Shell Descriptions

The HP-UX operating system supports the following shells:

sh	POSIX-conforming command programming language and command interpreter residing in file **/usr/bin/sh**. Can execute commands read from a terminal or a file. This shell conforms to current POSIX standards in effect at the time the HP-UX system release was introduced, and is similar to the Korn shell in many respects. Similar in many respects to the Korn shell, the POSIX shell contains a history mechanism, supports job control, and provides various other useful features.
sh	Bourne-shell command programming language and commands interpreter residing in file **/usr/old/bin/sh**. Can execute commands read from a terminal or a file. This shell lacks many features contained in the POSIX and Korn shells. The Bourne shell will be obsoleted. Users are strongly encouraged to switch to the POSIX shell. The Bourne shell will still be available as **/usr/old/bin/sh**, for those users have to use it.
ksh	Korn-shell command programming language and commands interpreter residing in file **/usr/bin/ksh**. Can execute commands read from a terminal or a file. This shell, like the POSIX shell, contains a history mechanism, supports job control, and provides various other useful features.
csh	A command language interpreter that incorporates a command history buffer, C-language-like syntax, and job control facilities.
rsh	Restricted version of the POSIX or Bourne shell command interpreter. Sets up a login name and execution environment whose capabilities are more controlled (restricted) than normal user shells.
rksh	restricted version of the Korn-shell command interpreter Sets up a login name and execution environment whose capabilities are more controlled (restricted) than normal user shells.

keysh An extension of the standard Korn Shell that uses hierarchical softkey menus and context-sensitive help.

To obtain:	Use the command:
POSIX Shell	**/usr/bin/sh** ...
Korn Shell	**/usr/bin/ksh** ...
C Shell	**/usr/bin/csh** ...
Key Shell	**/usr/bin/keysh**
Bourne Shell	**/usr/old/bin/sh** ...

These shells can also be the default invocation, depending on the entry in the **/etc/passwd** file. See also *chsh*(1).

Whether the **sh** command invokes the Bourne Shell or the POSIX Shell depends on the setting of the **PATH** environment variable.

The default **PATH** in file **/etc/profile** is set to invoke the POSIX shell.

WARNINGS

Many manual entries contain descriptions of shell behavior or describe program or application behavior similar to "the shell" with a reference to "see *sh*(1)".

SEE ALSO

For more information on the various individual shells, see:

sh-bourne(1)	Bourne Shell (**/usr/old/bin/sh**) description.
ksh(1)	Korn Shell (**/usr/bin/ksh**) description.
sh-posix(1)	POSIX Shell (**/usr/bin/sh**) description.
csh(1)	C Shell (**/usr/bin/csh**) description.
keysh(1)	Key Shell (**/usr/bin/keysh**) description.

S

NAME

shar - make a shell archive package

SYNOPSIS

shar [*options*] [*file* | *dir*] ... **>** *package*

DESCRIPTION

The **shar** command bundles the named files and directories into a single distribution package suitable for mailing or moving. The files can contain any data, including executables. The resulting package, written to standard output, is a shell script file that can be edited (to add messages at the beginning, etc.).

To unpack *package*, use the *sh*(1) command with the package name as an argument as follows:

sh *package*

When unpacking, the files and directories in *package* are written to the path names recorded in the archive.

If a directory is specified and the **-d** option is not given, all files beneath that directory are archived.

If a special file is specified, the appropriate **mknod** commands are emitted to recreate the file (see *mknod*(1)).

shar protects the contained files from mail processing, if necessary, by inserting an **@** character at the beginning of each line. If the file contains unusual data, the data is transformed into **uuencode** format, and a **uudecode** script is included in *package* so that the package can still be unpacked correctly by **sh**. See WARNINGS for more information about mailers and file modifications.

Access modes are preserved for both directories and files.

Options

shar recognizes the following options:

-a Assume that files can be shipped, regardless of their contents; do not protect them specially. **shar** is conservative, and might decide to **uuencode** a file containing special characters (such as Ctrl-G) that the user knows do not need protection.

-A Suppress warning messages regarding optional access control list entries. **shar** does not archive optional access control list entries in a file's access control list (see *acl*(5)). Normally, a warning message is printed for each file having optional access control list entries.

-b Archive files under their base names, regardless of the original path names specified. The contents are thus unpacked into the current directory instead of to the originally specified path names. This allows you to archive files from many directories but unpack them into a single directory. It also allows you to unpack, for example, **/usr/share/lib/termcap** into **./termcap** instead of overwriting the original one in **/etc**.

-c Append to the package a simple data-integrity check using **wc** to ensure that the contents were not damaged in transit (see *wc*(1)). This check is performed automatically after unpacking. Also see WARNINGS below.

-C Insert a line of the form **--- cut here ---** before the archive.

-d If a directory is specified, do not transmit its contents, but rather only create the empty directory.

-D*dir* Cause the archive to contain code that notifies the user if his or her current directory is not the same as *dir*, which must be an absolute path. If the user is not in *dir*, the unpacking can be continued by responding **yes** to the archive's question.

-e Cause the archive to contain code that prevents **shar** from unpacking files that would overwrite existing files.

-f*file* Read a list of file names from *file* and archive those files as if they were given as arguments.

-h Follow symbolic links as if they were normal files or directories. If this option is not specified, **shar** archives the link.

-m Retain modification and access times on files when they are unpacked.

S

-o Preserve user and group ownership on files and directories.

-r Cause the archive to contain code requiring that the user unpacking it be **root**. This is useful for processing system archives.

-s Perform error checking using **sum** (see *sum*(1)). Both **-c** and **-s** can be specified for better error checking. Also see WARNINGS below.

-t Write diagnostics and messages directly to your terminal instead of to the standard error. This is useful when invoking **shar** from programs (such as **vi** that normally combine standard error with standard output. Specifying **-t** also invokes the **-v** (verbose) option.

-u Assume that the remote site has **uudecode** for unpacking. If this option is not specified, a version of **uudecode** is sent and compiled if any non-ASCII files are archived.

-v Announce archived file names as they are packed. The **-t** option determines the destination for these announcements.

-Z Compress files using **compress** (see *compress*(1)).

Most options are flagged in the header of the resulting package, thereby recording the format of the archive. The name of the archiver, system, and time/date of the archive are also recorded in the header.

EXAMPLES

To archive all files under your home directory, type:

```
cd; shar -cmos .
```

or

```
shar -cmos $HOME
```

To preserve your **/dev** directory, type:

```
shar -mor /dev >save_dev_files
```

To send your newest programs in directory **newstuff** in your home directory to a friend, type:

```
cd; shar -cmos newstuff | mailx -s ’new source’ friend
```

RETURN VALUE

shar returns zero if successful; nonzero if problems with arguments occur.

DIAGNOSTICS

If the **-b** option is specified, **shar** refuses to archive directories.

WARNINGS

The modification and access time restoration does not take time zones into account.

Files with newline characters in their names scramble the table of contents.

Non-ASCII files with white space in their names do not unpack.

If a mailer such as *elm*(1) is used to transfer *package* to another system and the mailer is configured to expand tabs (by default or otherwise), any file in the archive will be modified if it contains tabs. If the **-c** or **-s** option is used to create the archive, the data-integrity check will fail during unpacking of any files in *package* that contain tab characters that were converted to spaces. (Some mailers that expand tabs when transferring files over a network may or may not expand tabs when transferring files to the sender or other users on the local system.) If an editor is used to modify any of the files in *package*, the data-integrity check will also fail for the files that were changed.

AUTHOR

shar was invented in the public domain. This version of **shar** was developed by HP.

FILES

/dev/tty
/tmp/unpack$$* (for unpacking non-ASCII files)

SEE ALSO

ar(1), compress(1), cpio(1), find(1), tar(1), acl(5).

NAME

shl - shell layer manager

SYNOPSIS

shl

DESCRIPTION

shl provides a means for interacting with more than one shell from a single terminal by using shell layers. A layer is a shell that is bound to a virtual device. The virtual device can be manipulated like an actual terminal by using **stty** and **ioctl()** (see *stty*(1) and *ioctl*(2)). Each layer has its own process group ID. The user controls these layers by using the commands described below.

The current layer is the layer that can receive input from the keyboard. Other layers attempting to read from the keyboard are blocked. Output from multiple layers is multiplexed onto the terminal. To block the output of a layer when it is not current, the **stty** option **loblk** can be set within the layer.

The **stty** character **swtch** (set to **^Z** if NUL) is used to switch control to **shl** from a layer. **shl** has its own prompt, **>>>**, to distinguish it from a layer.

Definitions

A **name** is a sequence of characters delimited by a space, tab, or new-line character. Only the first eight characters are significant. When provided as an argument to the **create**, **login**, or **name** commands, *name* cannot be of the form *n* or **(***n***)**, where *n* is a decimal number.

Commands

The following commands can be issued from the **shl** prompt level. Any unique prefix is accepted.

create [**-**[*name*] | *name* [*command*]]
: Create a layer called *name* and make it the current layer. If no argument is given, a layer is created with a name of the form **(***n***)**, where *n* is the number of the next available slot in an internal table. Future references to this layer can be made with or without the parentheses. If *name* is followed by a command, that command is executed in the layer instead of a shell. If **-** is the first argument, a "login shell" is created in the layer. The shell prompt variable **PS1** is set to the name of the layer followed by a space.

login [*name*]
: Create a layer called *name* and make it the current layer. If no argument is given, a layer is created with a name of the form **(***n***)**, where *n* is the number of the next available slot in an internal table. Future references to this layer can be made with or without the parentheses. *name* is a "login" type of layer in which the user receives a prompt for user name and password.

name [*oldname*] *newname*
: Rename the layer *oldname*, calling it *newname*. If *oldname* is not specified, the current layer name is changed.

! [*command*]
: Invoke a sub-shell and execute *command*. If no *command* is given, a shell is executed according to the **SHELL** environment variable.

block *name* [*name* ...]
: For each *name*, block the output of the corresponding layer when it is not the current layer. This is equivalent to setting the **stty**loblk option within the layer.

delete *name* [*name* ...]
: For each *name*, delete the corresponding layer. All processes in the process group of the layer are sent the **SIGHUP** signal (see *signal*(5)).

help (or **?**)
: Print the syntax of the **shl** commands.

layers [**-l**] [*name* ...]
: For each *name*, list the layer name and its process group. The **-l** option produces a *ps*(1)-like listing. If no arguments are given, information is presented for all existing layers.

resume [*name*]
: Change the status of the layer referred to by *name* to that of current layer. If no argument is given, the last existing current layer is changed.

toggle Change the status of the previous current layer to that of current layer.

unblock *name* [*name* ...]
For each *name*, do not block the output of the corresponding layer when it is not the current layer. This is equivalent to setting the **stty**-loblk option within the layer.

quit Exit **shl**. All layers are sent the **SIGHUP** signal.

name Change the status of the layer referred to by *name* to that of current layer. Any unique prefix is accepted.

WARNINGS

Commands

The behavior of the **block** and **unblock shl** commands is not guaranteed when the SHELL environment variable is set to **/usr/bin/csh** (for *csh*(1)) or **/usr/bin/ksh** (for *ksh*(1)), or when the shell saves and restores the tty state (defined in *termio*(7)) before and after each command is invoked interactively from that shell. For both **/usr/bin/csh** and **/usr/bin/ksh**, the **loblk** or **-loblk** options of **stty** can be used from within the layer to block or unblock the output of that layer.

Ptydaemon

For **shl** to function properly, the **ptydaemon** process must be running on the system. If your system has been installed with the Desktop HP-UX product, then **ptydaemon** will not be started by default. In order to start this daemon, change **PTYDAEMON_START** from a "0" to a "1" in the **/etc/rc.config.d/ptydaemon** file. The system must either be rebooted for this change to take effect, or you can manually start this daemon by typing :

```
/usr/sbin/ptydaemon
```

Note that **ptydaemon** will also be disabled if the *DesktopConfig.LITECONFIG* fileset has been installed on the system, or if the system administrator has previously run the **SAM** utility and selected the **Apply Lite HP-UX Configuration Action** from within any of **SAM's Kernel Configuration** screens.

DEPENDENCIES

Series 800

The **login** command is not currently supported.

FILES

$SHELL Variable containing path name of the shell to use (default is **/usr/bin/sh**).

SEE ALSO

sh(1), stty(1), ioctl(2), signal(5).

STANDARDS CONFORMANCE

shl: SVID2, SVID3, XPG2

NAME

size - print section sizes of object files

SYNOPSIS

size [**-d**] [**-o**] [**-x**] [**-V**] [**-v**] [**-f**] [**-F**] [**-n**] [**-U**] *files*

DESCRIPTION

size produces section size information for each section in the object files. The size of the text, data and bss (uninitialized data) sections are printed along with the total size of the object file. If an archive file is input to the **size** command, the information for all archive members is displayed.

Options

size recognizes the following options:

-d Print sizes in decimal. This is the default.

-o Print sizes in octal.

-x Print sizes in hexadecimal.

-V Print version information about the **size** command.

-v Print a verbose list of the subspaces in the object files. Each subspace is listed on a separate line with its size, physical address, and virtual address.

-f Print the size of each allocatable section (ELF only).

-F Print the size and permission bits of each loadable segment (ELF only).

-n Print the sizes of non loadable segments or non allocatable sections (ELF only).

-U Print the usage menu.

EXTERNAL INFLUENCES

Environment Variables

The following internationalization variables affect the execution of **size**:

LANG
Determines the locale category for native language, local customs and coded character set in the absence of **LC_ALL** and other **LC_*** environment variables. If **LANG** is not specified or is set to the empty string, a default of **C** (see *lang*(5)) is used instead of **LANG**.

LC_ALL
Determines the values for all locale categories and has precedence over **LANG** and other **LC_*** environment variables.

LC_MESSAGES
Determines the locale that should be used to affect the format and contents of diagnostic messages written to standard error.

LC_NUMERIC
Determines the locale category for numeric formatting.

LC_CTYPE
Determines the locale category for character handling functions.

ST_SIZECAT
NLSPATH
Determines the location of message catalogues for the processing of **LC_MESSAGES**.

If any internationalization variable contains an invalid setting, **size** behaves as if all internationalization variables are set to **C**. See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported.

S

DIAGNOSTICS

size: *name***: cannot open** *name* cannot be read.

size: *name***: bad magic** *name* is not an appropriate object file.

EXAMPLES

Compare the sizes of the text, data, and bss sections for two versions of a program:

```
size ./version1 ./version2
```

SEE ALSO

System Tools:

as(1)	translate assembly code to machine code
cc(1)	invoke the HP-UX C compiler
ld(1)	invoke the link editor

Miscellaneous:

a.out(4)	assembler, compiler, and linker output
ar(4)	archive format

STANDARDS CONFORMANCE

`size`: SVID2, SVID3, XPG2, XPG4

NAME

sleep - suspend execution for an interval

SYNOPSIS

sleep *time*

DESCRIPTION

sleep suspends execution for *time* seconds. It is used to execute a command after a certain amount of time, as in:

```
(sleep 105; command)&
```

or to execute a command periodically, as in:

```
while true
do
  command
    sleep 37
done
```

RETURN VALUE

sleep exits with one of the following values:

0 The execution was successfully suspended for *time* seconds, or a SIGALRM signal was received.

>0 If the *time* operand is missing, is not a decimal integer, is negative, or is greater than **UINT_MAX**, **sleep** returns with exit status 2.

SEE ALSO

alarm(2), sleep(3C).

STANDARDS CONFORMANCE

sleep: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

NAME

slp - set printing options for a non-serial printer

SYNOPSIS

slp [**-a**] [**-b**] [**-c** *cols*] [**-d**] [**-i** *indent*] [**-k**] [**-l** *lines*] [**-n**] [**-o**] [**-r**] [**-C** *pages*] [**-O** *pages*]

DESCRIPTION

slp sets printer formatting options such as the number of lines per page, number of characters per line, and indentation. These characteristics are controlled by the printer driver as described in *lp*(7). **slp** acts on the current standard output.

Options

slp recognizes the following options and arguments:

-a	Report all option settings.
-b	Specify a character printer where backspace characters pass through the driver unchanged. The absence of this option indicates a line printer. The driver takes the necessary action to accommodate backspace characters.
-o	Resets the printer back to line-printer mode.
-c*cols*	Limit the number of columns to be printed to *cols*. Characters beyond the specified limit are truncated.
-d	Reset options to default for the device. This action is not taken until the next open occurs on the device.
-i*indent*	Indent *indent* columns before printing the first column.
-k	Select cooked mode. Cooked mode must be used with a cooked device special file which is identified by an lp mnemonic that is not preceded by the character **r**.
-l*lines*	Specify the number of *lines* per page. The last new-line character of each page is changed to a form-feed.
-n	Set the page size to infinity. Since the last new-line of the page is never encountered, no new-line characters are changed to form-feeds.
-r	Select a raw mode for graphics dumps. All other options are ignored except **-a**. If the **-r** option is not given, **-k** is assumed.
-C*pages*	Eject zero or more *pages* after the final close of the device.
-O*pages*	Eject zero or more *pages* when the device is opened.

EXAMPLES

In a typical case, the printer is set to 80 columns, no indentation, with no form-feeds between pages:

```
slp -c80 -i0 -n >/dev/lp
```

WARNINGS

Use of the **slp** command in conjunction with the **lp** spooler (see *lp*(1)) might cause undesirable side effects. The spooler model files make assumptions regarding the configuration and can get confused when the default values are altered. Although most options can be altered without difficulty, special problems sometimes result from adjusting the number of lines and the number of columns per page.

AUTHOR

slp was developed by HP.

SEE ALSO

lp(1), ioctl(2), lp(7).

NAME

sna - SNAplus2: SNA and APPN communications for HP-UX.

DESCRIPTION

The following descriptions are provided as a very brief overview of selected SNA (Systems Network Architecture) utilities that are available as optional software for use with the HP-UX operating system. For more information refer to system documentation provided with the software. For purchasing information, contact your nearest HP Sales and Support Office.

SNAplus2 is a communications software product which permits users of a HP-UX computer to communicate with remote computers using IBM SNA protocols. It implements a LEN Node, APPN End Node, or APPN Network Node according to its configuration; this includes SNA node type 2.0 and 2.1 support for communicating with host and peer computers.

SNAplus2 may be used on a single HP-UX computer, or as a Client-Server system with users and programs distributed across a LAN (communicating using TCP/IP). The core communications components run on HP-UX servers; client computers may be running either HP-UX or Windows.

SNAplus2 supports communications using the following connectivity types:

SDLC
QLLC
TR
FDDI
ETH

It allows the following types of communications tasks:

3270 and 3179G emulation
Your terminal acts as a 3270 terminal, or as a 3179G graphics display terminal, allowing you to access programs on host computers.

LU 6.2 Communications (APPC and CPI-C)
Advanced Program-to-Program Communications (APPC) and Common Programming Interface for Communications (CPI-C) both allow communication between transaction programs (TPs) on different systems across the SNA network.

Conventional LU API (LUA)
LUA allows application programs to communicate with existing IBM host applications using any of the LU types 0, 1, 2, or 3.

Remote Job Entry (RJE)
RJE allows SNAplus2 users to submit jobs to a host Job Entry Subsystem for processing.

In addition to APPC, CPI-C, and LUA, SNAplus2 also provides the following APIs for use by application programs:

High-Level Language Application Programming Interface (HLLAPI)
HLLAPI interacts with a 3270 emulation session by performing the tasks that would normally be performed by a 3270 user, such as searching for data on the screen and entering keystrokes. It can be used to automate standard 3270 tasks.

Common Service Verbs (CSV)
The CSV API provides utility functions such as character translation, trace control, and message logging.

Management Services (MS)
The MS API allows a SNAplus2 application to communicate with other Management Services products or applications across the network, by sending and receiving management data either in NMVT format or in MDS_MU format.

Node Operator Facility (NOF)
The NOF API allows an application to configure and manage SNAplus2 resources; the application can define, modify, or delete resources, start or stop resources, or obtain information on the configuration or current status of resources.

SNAplus2 also provides the following features:

TN Server
SNAplus2 allows TN3270 programs (3270 emulation programs that communicate over TCP/IP)

running on other computers to access a 3270 host computer over a SNAplus2 host connection. Both the SNAplus2 TN3270 program (provided as a separate package from the main SNAplus2 product) and other vendors' TN3270 programs are supported.

Remote Command Facility (RCF)
SNAplus2 provides facilities to allow an operator using the NetView program at a host to issue commands from the NetView console, either to manage SNAplus2 components or to issue HP-UX commands on the SNAplus2 computer.

SEE ALSO

SNA system software documentation provided with optional SNA software.

* IBM is a trademark of International Business Machines Corporation.

NAME

soelim - eliminate .so's from nroff input

SYNOPSIS

soelim [*file* ...]

DESCRIPTION

soelim reads the specified files or the standard input and performs the textual inclusion implied by **nroff** directives of the form

.so *some_file*

when they appear at the beginning of input lines. This is useful when using programs such as *tbl*(1) that do not normally do this, allowing placement of individual tables or other text objects in separate files to be run as a part of a large document.

An argument consisting of a single minus (**-**) is taken to be a file name corresponding to the standard input.

Note that inclusion can be suppressed by using **'** instead of **.** at the start of the line as in:

```
'so /usr/share/lib/tmac/tmac.s
```

EXAMPLES

soelim is often used in a context similar to the following:

```
soelim exum?.n | tbl | nroff -mm | col | lp
```

WARNINGS

The format of the source commands must involve no strangeness — exactly one blank must precede and no blanks follow the file name.

SEE ALSO

more(1), nroff(1), tbl(1).

NAME

softbench - SoftBench Software Development Environment

DESCRIPTION

The **SoftBench** and **Encapsulator** products are designed to improve programmer and team productivity by providing a software development environment that:

- Facilitates rapid, interactive program development and simplifies program maintenance and porting in a distributed computing environment.
- Is easy to learn and use.
- Can be easily customized to leverage a customer's existing environment, processes and tools.

SoftBench

SoftBench is an integrated set of X11/Motif window-based programming tools and a Tool Integration Platform. Together they provide an integrated software development environment targeted at the program construction, test, and maintenance phases of software development.

The SoftBench product is composed of:

- A language-sensitive **Program Editor** and **Program Builder** to support rapid, interactive program construction.
- A **Static Analyzer** and **Program Debugger** for understanding the structure and behavior of complex applications in order to support program test, maintenance, and porting.
- A **Development Manager** used to manage the versions of files the SoftBench tools operate on.

SoftBench programming tools are integrated via services that include:

- A tool communication architecture designed to support a task-oriented environment of cooperative tools.
- Support for a distributed software development environment allowing remote tool execution and remote data access.
- A graphical, OSF/Motif user interface.
- A pervasive, interactive help system.

Encapsulator

Encapsulator delivers the customizability benefit of SoftBench. It allows users to customize and extend the SoftBench environment by:

- Automating custom development processes. Encapsulator is used to define actions to be executed whenever specific events occur in the SoftBench environment.
- Adding the SoftBench graphical user interface to existing HP-UX utilities, customer tools, and third-party tools.

These extensions are made to the environment without modifying the source code of the tools that are encapsulated.

Product Information

SoftBench and Encapsulator run on HP 9000 computers under HP-UX and on other platforms. Contact your HP Sales Representative for ordering information.

INTERNATIONAL SUPPORT

SoftBench supports both 8-bit and 16-bit languages.

AUTHOR

SoftBench was developed by HP.

SEE ALSO

The following references are contained in other manuals shipped with SoftBench software, and are not included in this manual: softbench(1), encapsulate(1), encaprun(1), softbench(5), softbuild(1), softdebug(1), softdm(1), softedit(1), softeditsrv(1), softmsg(1), softstatic(1).

NAME

sort - sort or merge files

SYNOPSIS

sort [**-m**] [**-o** *output*] [**-bdfinruM**] [**-t** *char*] [**-k** *keydef*] [**-y** [*kmem*]] [**-z** *recsz*] [**-T** *dir*] [*file* ...]

sort [**-c**] [**-AbdfinruM**] [**-t** *char*] [**-k** *keydef*] [**-y** [*kmem*]] [**-z** *recsz*] [**-T** *dir*] [*file* ...]

DESCRIPTION

sort performs one of the following functions:

1. Sorts lines of all the named files together and writes the result to the specified output.
2. Merges lines of all the named (presorted) files together and writes the result to the specified output.
3. Checks that a single input file is correctly presorted.

The standard input is read if **-** is used as a file name or no input files are specified.

Comparisons are based on one or more sort keys extracted from each line of input. By default, there is one sort key, the entire input line. Ordering is lexicographic by characters using the collating sequence of the current locale. If the locale is not specified or is set to the **POSIX** locale, then ordering is lexicographic by bytes in machine-collating sequence. If the locale includes multi-byte characters, single-byte characters are machine-collated before multi-byte characters.

Behavior Modification Options

The following options alter the default behavior:

-A Sorts on a byte-by-byte basis using each character's encoded value. On some systems, extended characters will be considered negative values, and so sort before ASCII characters. If you are sorting ASCII characters in a non-C/POSIX locale, this flag performs much faster.

-c Check that the single input file is sorted according to the ordering rules. No output is produced; the exit code is set to indicate the result.

-m Merge only; the input files are assumed to be already sorted.

-o *output* The argument given is the name of an output file to use instead of the standard output. This file can be the same as one of the input files.

-u Unique: suppress all but one in each set of lines having equal keys. If used with the **-c** option, check to see that there are no lines with duplicate keys, in addition to checking that the input file is sorted.

-y [*kmem*] The amount of main memory used by the sort can have a large impact on its performance. If this option is omitted, **sort** begins using a system default memory size, and continues to use more space as needed. If this option is presented with a value, *kmem*, **sort** starts using that number of kilobytes of memory, unless the administrative minimum or maximum is violated, in which case the corresponding extremum will be used. Thus, **-y** *0* is guaranteed to start with minimum memory. By convention, **-y** (with no argument) starts with maximum memory.

-z *recsz* The size of the longest line read is recorded in the sort phase so that buffers can be allocated during the merge phase. If the sort phase is omitted via the **-c** or **-m** options, a popular system default size will be used. Lines longer than the buffer size will cause **sort** to terminate abnormally. Supplying the actual number of bytes in the longest line to be merged (or some larger value) will prevent abnormal termination.

-T *dir* Use *dir* as the directory for temporary scratch files rather than the default directory, which is is one of the following, tried in order: the directory as specified in the **TMPDIR** environment variable; **/var/tmp**, and finally, **/tmp**.

Ordering Rule Options

When ordering options appear before restricted sort key specifications, the ordering rules are applied globally to all sort keys. When attached to a specific sort key (described below), the ordering options override all global ordering options for that key.

The following options override the default ordering rules:

-d Quasi-dictionary order: only alphanumeric characters and blanks (spaces and tabs), as defined by **LC_CTYPE** are significant in comparisons (see *environ*(5)).

(XPG4 only.) The behavior is undefined for a sort key to which -i or -n also applies.

-f Fold letters. Prior to being compared, all lowercase letters are effectively converted into their uppercase equivalents, as defined by **LC_CTYPE**.

-i In non-numeric comparisons, ignore all characters which are non-printable, as defined by **LC_CTYPE**. For the ASCII character set, octal character codes 001 through 037 and 0177 are ignored.

-n The sort key is restricted to an initial numeric string consisting of optional blanks, an optional minus sign, zero or more digits with optional radix character, and optional thousands separators. The radix and thousands separator characters are defined by **LC_NUMERIC**. The field is sorted by arithmetic value. An empty (missing) numeric field is treated as arithmetic zero. Leading zeros and plus or minus signs on zeros do not affect the ordering. The **-n** option implies the **-b** option (see below).

-r Reverse the sense of comparisons.

-M Compare as months. The first several non-blank characters of the field are folded to uppercase and compared with the *langinfo*(5) items **ABMON_1** < **ABMON_2** < ... < **ABMON_12**. An invalid field is treated as being less than **ABMON_1** string. For example, American month names are compared such that **JAN** < **FEB** < ... < **DEC**. An invalid field is treated as being less than all months. The **-M** option implies the **-b** option (see below).

Field Separator Options

The treatment of field separators can be altered using the options:

-t *char* Use *char* as the field separator character; *char* is not considered to be part of a field (although it can be included in a sort key). Each occurrence of *char* is significant (for example, <*char*><*char*> delimits an empty field). If **-t** is not specified, <blank> characters will be used as default field separators; each maximal sequence of <blank> characters that follows a non-<blank> character is a field separator.

-b Ignore leading blanks when determining the starting and ending positions of a restricted sort key. If the **-b** option is specified before the first **-k** option (**+***pos1* argument), it is applied to all **-k** options (**+***pos1* arguments). Otherwise, the **-b** option can be attached independently to each **-k** *field_start* or *field_end* option (**+***pos1* or (**-***pos2* argument; see below). Note that the **-b** option is only effective when restricted sort key specifications are given.

Restricted Sort Key

-k *keydef* The *keydef* argument defines a restricted sort key. The format of this definition is

field_start[*type*][**,***field_end*[*type*]]

which defines a key field beginning at *field_start* and ending at *field_end*. The characters at positions *field_start* and *field_end* are included in the key field, providing that *field_end* does not precede *field_start*. A missing *field_end* means the end of the line. Fields and characters within fields are numbered starting with **1**. Note that this is different than the obsolete form of restricted sort keys, where numbering starts at **0**. See WARNINGS below.

Specifying *field_start* and *field_end* involves the notion of a field, a minimal sequence of characters followed by a field separator or a new-line. By default, the first blank of a sequence of blanks acts as the field separator. All blanks in a sequence of blanks are considered to be part of the next field; for example, all blanks at the beginning of a line are considered to be part of the first field.

The arguments *field_start* and *field_end* each have the form *m***.***n* which are optionally followed by one or more of the *type* options **b**, **d**, **f**, **i**, **n**, **r**, or **M**. These modifiers have the functionality for this key only, that their command-line counterparts have for the entire record.

A *field_start* position specified by *m*.*n* is interpreted to mean the *n*th character in the *m*th field. A missing *n* means **.1**, indicating the first character of the *m*th field. If the **-b** option is in effect, *n* is counted from the first non-blank character in the *m*th field.

A *field_end* position specified by *m*.*n* is interpreted to mean the *n*th character in the *m*th field. If *n* is missing, the *m*th field ends at the last character of the field. If the **-b** option is in effect, *n* is counted from the first non-<blank> character in the *m*th field.

Multiple **-k** options are permitted and are significant in command line order. A maximum of 9 **-k** options can be given. If no **-k** option is specified, a default sort key of the entire line is used. When there are multiple sort keys, later keys are compared only after all earlier keys compare equal. Lines that otherwise compare equal are ordered with all bytes significant. If all the specified keys compare equal, the entire record is used as the final key.

The **-k** option is intended to replace the obsolete [**+***pos1* [**+***pos2*]] notation, using *field_start* and *field_end* respectively. The fully specified [**+***pos1* [**+***pos2*]] form:

+*w***.***x***-***y***.***z*

is equivalent to:

-k *w***+1.***x***+1,***y***.0** (if *z* == 0)
-k *w***+1.***x***+1,***y***+1.***z* (if *z* > 0)

Obsolete Restricted Sort Key

The notation **+***pos1* **-***pos2* restricts a sort key to one beginning at *pos1* and ending at *pos2*. The characters at positions *pos1* and *pos2* are included in the sort key (provided that *pos2* does not precede *pos1*). A missing **-***pos2* means the end of the line.

Specifying *pos1* and *pos2* involves the notion of a field, a minimal sequence of characters followed by a field separator or a new-line. By default, the first blank (space or tab) of a sequence of blanks acts as the field separator. All blanks in a sequence of blanks are considered to be part of the next field; for example, all blanks at the beginning of a line are considered to be part of the first field.

pos1 and *pos2* each have the form *m*.*n* optionally followed by one or more of the flags **bdfinrM**. A starting position specified by **+***m*.*n* is interpreted to mean character *n*+1 in field *m*+1. A missing **.***n* means **.**0, indicating the first character of field *m*+1. If the **b** flag is in effect, *n* is counted from the first non-blank in field *m*+1; **+***m***.0b** refers to the first non-blank character in field *m*+1.

A last position specified by **-***m*.*n* is interpreted to mean the *n*th character (including separators) after the last character of the *m th* field. A missing **.***n* means **.**0, indicating the last character of the *m*th field. If the **b** flag is in effect, *n* is counted from the last leading blank in field *m*+1; **-***m***.1b** refers to the first non-blank in field *m*+1.

EXTERNAL INFLUENCES

Environment Variables

LC_COLLATE determines the default ordering rules applied to the sort.

LC_CTYPE determines the locale for interpretation of sequences of bytes of text data as characters (e.g., single- verses multibyte characters in arguments and input files) and the behavior of character classification for the **-b**, **-d**, **-f**, **-i**, and **-n** options.

LC_NUMERIC determines the definition of the radix and thousands separator characters for the **-n** option.

LC_TIME determines the month names for the **-M** option.

LC_MESSAGES determines the language in which messages are displayed.

LC_ALL determines the locale to use to override the values of all the other internationalization variables.

NLSPATH determines the location of message catalogs for the processing of **LC_MESSAGES**.

LANG provides a default value for the internationalization variables that are unset or null. If **LANG** is unset or null, the default value of "C" (see *lang*(5)) is used.

If any of the internationalization variables contains an invalid setting, **sort** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported.

EXAMPLES

Sort the contents of **infile** with the second field as the sort key:

```
sort -k 2,2 infile
```

Sort, in reverse order, the contents of **infile1** and **infile2**, placing the output in **outfile** and using the first two characters of the second field as the sort key:

```
sort -r -o outfile -k 2.1,2.2 infile1 infile2
```

Sort, in reverse order, the contents of **infile1** and **infile2**, using the first non-blank character of the fourth field as the sort key:

```
sort -r -k 4.1b,4.1b infile1 infile2
```

Print the password file (**/etc/passwd**) sorted by numeric user ID (the third colon-separated field):

```
sort -t: -k 3n,3 /etc/passwd
```

Print the lines of the presorted file **infile**, suppressing all but the first occurrence of lines having the same third field:

```
sort -mu -k 3,3 infile
```

DIAGNOSTICS

sort exits with one of the following values:

0 All input files were output successfully, or **-c** was specified and the input file was correctly presorted.

1 Under the **-c** option, the file was not ordered as specified, or if the **-c** and **-u** options were both specified, two input lines were found with equal keys. This exit status is not returned if the **-c** option is not used.

>1 An error occurred such as when one or more input lines are too long.

When the last line of an input file is missing a new-line character, **sort** appends one, prints a warning message, and continues.

If an error occurs when accessing the tables that contain the collation rules for the specified language, **sort** prints a warning message and defaults to the **POSIX** locale.

If a **-d**, **-f**, or **-i** option is specified for a language with multi-byte characters, **sort** prints a warning message and ignores the option.

WARNINGS

Numbering of fields and characters within fields (**-k** option) has changed to conform to the POSIX standard. Beginning at HP-UX Release 9.0, the **-k** option numbers fields and characters within fields, starting with **1**. Prior to HP-UX Release 9.0, numbering started at **0**.

A field separator specified by the **-t** option is recognized only if it is a single-byte character.

The character type classification categories **alpha**, **digit**, **space**, and **print** are not defined for multi-byte characters. For languages with multi-byte characters, all characters are significant in comparisons.

FILES

/var/tmp/stm???
/tmp/stm???

AUTHOR

sort was developed by OSF and HP.

SEE ALSO

comm(1), join(1), uniq(1), collate8(4), environ(5), hpnls(5), lang(5).

STANDARDS CONFORMANCE

sort: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

NAME

spell, hashmake, spellin, hashcheck - find spelling errors

SYNOPSIS

spell [**-v**] [**-b**] [**-x**] [**-l**] [**-i**] [**+***local_file*] [*files*]

/usr/lbin/spell/hashmake

/usr/lbin/spell/spellin *n*

/usr/lbin/spell/hashcheck *spelling_list*

DESCRIPTION

The **spell** command collects words from the named *files* and looks them up in a spelling list. Words that neither occur among nor are derivable (by applying certain inflections, prefixes, and/or suffixes) from words in the spelling list are printed on the standard output. If no *files* are named, words are collected from the standard input.

The **spell** command ignores most **troff**, **tbl**, and *eqn* constructions.

Options

The **spell** command recognizes the following options:

-v All words not literally in the spelling list are printed, and plausible derivations from the words in the spelling list are indicated.

-b British spelling is checked. Besides preferring **centre**, **colour**, **programme**, **speciality**, **travelled**, etc., this option insists upon **-ise** in certain words, such as in **standardise**.

-x Every plausible stem is printed with **=** for each word.

By default, **spell** follows chains of included files much like **deroff** (see *deroff*(1)) which recognizes the **troff**/**nroff** intrinsics **.so** and **.nx**, *unless* the names of such included files begin with **/usr/share/lib**. If the **-l** option is used, **spell** follows the chains of *all* included files. With the **-i** option, **spell** ignores all chains of included files.

If the **+***local_file* option is used, words found in *local_file* are removed from **spell**'s output. *local_file* is the name of a user-provided file containing a sorted list of words, one per line. With this option, the user can specify a set of words that are correct spellings (in addition to *spell*'s own spelling list) for each job.

The spelling list is based on many sources, and while more haphazard than an ordinary dictionary, is also more effective with respect to proper names and popular technical words. Coverage of the specialized vocabularies of biology, medicine, and chemistry is light.

Pertinent auxiliary files can be specified by name arguments, indicated below with their default settings (see FILES and VARIABLES). Copies of all output are accumulated in the history file. The stop list filters out misspellings (such as **thier=thy-y+ier**) that would otherwise pass.

Three routines help maintain and check the hash lists used by **spell**:

hashmake Reads a list of words from the standard input and writes the corresponding nine-digit hash code on the standard output.

spellin *n* Reads *n* hash codes from the standard input and writes a compressed spelling list on the standard output. Information about the hash coding is printed on standard error.

hashcheck Reads a compressed *spelling_list* and recreates the nine-digit hash codes for all the words in it; it writes these codes on the standard output.

EXAMPLES

To check spelling of a single *word*:

echo *word* **| spell**

If *word* is spelled correctly, a prompt is returned. If it is spelled incorrectly, *word* is printed before the prompt is returned. To check spelling of multiple words, they can also be typed as a group on the same command line:

echo *worda wordb wordc ...* **| spell**

To create a personal spelling list that incorporates the words already present in the default American spelling list file **/usr/share/dict/hlista**:

```
cat /usr/share/dict/hlista | /usr/lbin/spell/hashcheck >tmp1
/usr/lbin/spell/hashmake <addwds >>tmp1
sort -u -o tmp1 tmp1
/usr/lbin/spell/spellin ‘wc -l <tmp1‘ <tmp1 >hlista
```

To modify the default British spelling list file **/usr/share/dict/hlistb**, replace all occurrences of **hlista** with **hlistb** in the above example.

To add words to the default spelling list, change login to **root**, change the current working directory to **/usr/share/dict** and execute the commands listed in the above example.

WARNINGS

The spelling list's coverage is uneven. When undertaking the use of **spell** as a new tool, it may be advisable to monitor the output for several months to gather local additions. Typically, these are kept in a separate local file that is added to the hashed *spelling_list* via **spellin**, as shown above.

The British spelling feature was developed by an American.

Start-up versions of files **hlista**, **hlistb**, and **hstop** are available in directory **/usr/newconfig/usr/share/dict**. If these files or a suitable equivalent are not present in directory **/usr/share/dict**, **spell** complains:

```
spell: cannot initialize hash table
spell: cannot initialize hash table
```

The **spell** command is likely to be withdrawn from X/Open standards. Applications using this command might not be portable to other vendors' systems.

FILES

/usr/share/dict/hlist[**ab**]	Hashed spelling lists, American and British.
/usr/share/dict/hstop	Hashed stop list.
/var/adm/spellhist	History file.
/usr/lbin/spell/spellprog	Executable program file.

VARIABLES

D_SPELL	Your hashed spelling list (default is **D_SPELL=/usr/share/dict/hlist**[**ab**])
H_SPELL	Spelling history (default is **H_SPELL=/var/adm/spellhist**).
S_SPELL	Your hashed stop list (default is **S_SPELL=/usr/share/dict/hstop**).
TMPDIR	Directory for temporary files; overrides the default **/tmp**.

SEE ALSO

deroff(1), sed(1), sort(1), tbl(1), tee(1).

STANDARDS CONFORMANCE

spell: SVID2, SVID3, XPG2, XPG3

NAME

split - split a file into pieces

SYNOPSIS

split [**-l** *line_count*] [**-a** *suffix_length*] [*file* [*name*]]

split [**-b** *n*[**k**|**m**]] [**-a** *suffix_length*] [*file* [*name*]]

Obsolescent

split [**-***n*] [*file* [*name*]]

DESCRIPTION

split reads *file* and writes it in pieces (default 1000 lines) onto a set of output files. The name of the first output file is *name* with **aa** appended, and so on lexicographically, up to **zz** (only ASCII letters are used, a maximum of 676 files). If no output *name* is given, **x** is the default.

If no input *file* is given, or if **-** is given instead, the standard input file is used.

Options

split recognizes the following command-line options and arguments:

-l *line_count* The input file is split into pieces *line_count* lines in size.

-a *suffix_length*
suffix_length letters are used to form the suffix of the output filenames. This option allows creation of more than 676 output files. The output file names created cannot exceed the maximum file name length allowed in the directory containing the files.

-b *n* The input file is split into pieces *n* bytes in size.

-b *n***k** The input file is split into pieces $n \times 1024$ bytes in size. No space separates the *n* from the **k**.

-b *n***m** The input file is split into pieces $n \times 1\,048\,576$ bytes in size. No space separates the *n* from the **m**.

-*n* The input file is split into pieces *n* lines in size. This option is obsolescent and is equivalent to using the **-l** *line_count* option.

EXTERNAL INFLUENCES

Environment Variables

LC_CTYPE determines the locale for the interpretation of text as single- and/or multi-byte characters.

LC_MESSAGES determines the language in which messages are displayed.

If **LC_CTYPE** or **LC_MESSAGES** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**.

If any internationalization variable contains an invalid setting, **split** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported.

SEE ALSO

bfs(1), csplit(1).

STANDARDS CONFORMANCE

split: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

NAME

ssp - remove multiple line-feeds from output

SYNOPSIS

ssp

DESCRIPTION

ssp (single-space) removes redundant blank lines from the standard input and sends the result to the standard output. All blank lines at the beginning of a file are removed, and all multiple blank lines elsewhere in the file (including end-of-file) are reduced to a single blank line.

ssp is typically used in pipelines such as

```
nroff -ms file1 | ssp
```

ssp is equivalent to the 4.2BSD **cat -s** command.

To remove all blank lines from a file except at beginning of file, use **rmnl** (see *rmnl*(1)). To remove all blank lines from a file including beginning of file, use **rmnl** piped to **ssp**, or **ssp** piped to **rmnl**.

SEE ALSO

cat(1), rmnl(1).

NAME

strings - find the printable strings in an object or other binary file

SYNOPSIS

strings [**-a**] [**-t** *format*] [**-n** *number*] [*file*] ...

Obsolescent

strings [**-a**] [**-o**] [**-***number*] [*file*] ...

DESCRIPTION

strings looks for ASCII strings in a file. If no *file* is specified, standard input is used. A string is any sequence of four or more printing characters ending with a newline or null character.

strings is useful for identifying random object files and many other things.

Options

strings recognizes the following options:

-a By default, *strings* looks only in the initialized data space of object files (as recognized by their magic numbers). If this flag is used, the entire file is inspected. This flag is always set if standard input is being read or the file is not recognized as an object file. For backward compatibility, **-** is understood as a synonym for **-a**.

-t *format* Write each string preceded by its byte offset from the start of the file. The format is dependent on the single character used as the *format* option-argument:

d	The offset is written in decimal.
o	The offset is written in octal.
x	The offset is written in hexadecimal.

-n *number* Specify *number* as the minimum string length, rather than the default 4.

-o Each string is preceded by its offset in the file (in octal). This option is obsolescent and is equivalent to specifying the **-t** *o* option.

-*number* Specify *number* as the minimum string length, rather than the default 4. This option is obsolescent and is equivalent to using the **-n** *number* option.

EXTERNAL INFLUENCES

Environment Variables

LC_CTYPE determines the locale for the interpretation of text as single- and/or multi-byte characters.

LC_MESSAGES determines the language in which messages are displayed.

If **LC_CTYPE** or **LC_MESSAGES** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**.

NLSPATH Determine the location of message catalogues for the processing of **LC_MESSAGES.**

If any internationalization variable contains an invalid setting, **strings** behaves as if all internationalization variables are set to "C". See *environ*(5).

WARNINGS

The algorithm for identifying strings is extremely primitive.

AUTHOR

strings was developed by the University of California, Berkeley.

SEE ALSO

od(1).

STANDARDS CONFORMANCE

strings: XPG4, POSIX.2

NAME

strip - strip symbol and line number information from an object file

SYNOPSIS

strip [**-l**] [**-x**] [**-r**] [**-V**] [**-U**] *filename* ...

DESCRIPTION

strip removes the symbol table and line number information from object files, including archives. Thereafter, no symbolic debugging access is available for that file; thus, this command is normally run only on production modules that have been debugged and tested. The effect is nearly identical to using the **-s** option of *ld*.

Options

The amount of information stripped from the symbol table can be controlled by using any of the following options:

-l Strip line number information only; do not strip any symbol table information.

-x Do not strip static or external symbol information.

Note that the **-l** and **-x** options are synonymous because the symbol table contains only static and external symbols. Either option strips only symbolic debugging information and unloadable data.

-r Reset the relocation indexes into the symbol table (SOM only). Obsolete for ELF files. This option allows **strip** to be run on relocatable files, in which case the effect is also to strip only symbolic debugging information and unloadable data.

-V Print the version of the strip command on the standard error output.

-U Print the usage menu.

If there are any relocation entries in the object file and any symbol table information is to be stripped, **strip** complains and terminates without stripping *filename* unless the **-r** option is used.

If **strip** is executed on an archive file (see *ar*(4)), the archive symbol table is removed. The archive symbol table must be restored by executing **ar** with its **s** operator (see *ar*(1)) before the archive can be used by the **ld** command (se *ld*(1)). **strip** instructs the user with appropriate warning messages when this situation arises.

The purpose of this command is to reduce file storage overhead consumed by the object file.

EXTERNAL INFLUENCES

Environment Variables

The following internationalization variables affect the execution of **strip:**

LANG

Determines the locale category for native language, local customs and coded character set in the absence of **LC_ALL** and other **LC_*** environment variables. If **LANG** is not specified or is set to the empty string, a default of **C** (see *lang*(5)) is used instead of **LANG**.

LC_ALL

Determines the values for all locale categories and has precedence over **LANG** and other **LC_*** environment variables.

LC_MESSAGES

Determines the locale that should be used to affect the format and contents of diagnostic messages written to standard error.

LC_NUMERIC

Determines the locale category for numeric formatting.

LC_CTYPE

Determines the locale category for character handling functions.

ST_STRIPCAT
NLSPATH

Determines the location of message catalogues for the processing of **LC_MESSAGES**.

If any internationalization variable contains an invalid setting, **`strip`** behaves as if all internationalization variables are set to **`C`**. See *environ*(5).

In addition, the following environment variable affects **`strip`**:

`TMPDIR`
Specifies a directory for temporary files (see *tmpnam*(3S)).

International Code Set Support

Single- and multi-byte character code sets are supported.

DIAGNOSTICS

`strip:` *name*: **`cannot open`**
name cannot be read.

`strip:` *name*: **`bad magic`**
name is not an appropriate object file.

`strip:` *name*: **`relocation entries present; cannot strip`**
name contains relocation entries and the **`-r`** option was not specified. Symbol table information cannot be stripped.

EXAMPLES

Strip symbol table and debug information from the shared library **`libfoo.sl`** in the current directory to reduce its size. Symbol information required to use the library is preserved:

```
strip ./libfoo.sl
```

FILES

`/var/tmp/SGSstrp*`	temporary files

SEE ALSO

System Tools:

ar(1)	create archived libraries
as(1)	translate assembly code to machine code
cc(1)	invoke the HP-UX C compiler
ld(1)	invoke the link editor

Miscellaneous:

a.out(4)	assembler, compiler, and linker output
ar(4)	archive format

STANDARDS CONFORMANCE

`strip`: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

NAME

stty - set the options for a terminal port

SYNOPSIS

stty [**-a** | **-g** | *options*]

DESCRIPTION

stty sets or reports current settings of certain terminal I/O options for the device that is the current standard input. The command takes four forms:

stty	Report the settings of a system-defined set of options;
stty -a	Report all of current option settings;
stty -g	Report current settings in a form that can be used as an argument to another **stty** command.
stty *options*	Set terminal I/O options as defined by *options*.

For detailed information about the modes listed below from Control Modes through Local Modes as they relate to asynchronous lines, see *termio*(7). For detailed information about the modes listed under Hardware Flow Control Modes below, see *termiox*(7).

Options in the Combination Modes group are implemented using options in the previous groups. Note that many combinations of options make no sense, but no sanity checking is performed.

options are selected from the following:

Control Modes

rows *number*	Set the terminal window row size equal to *number*.
columns *number*	Set the terminal window column size (width) equal to *number*. **cols** can be used as an abbreviation for **columns**.
parenb (**-parenb**)	Enable (disable) parity generation and detection.
parodd (**-parodd**)	Select odd (even) parity.
cs5 cs6 cs7 cs8	Select character size (see *termio*(7)).
0	Hang up phone line immediately.

50 75 110 134.5 150 200 300 600 900 1200 1800 2400
3600 4800 7200 9600 19200 38400 57600 115200 230400 exta extb

Set terminal baud rate to the number given, if possible (some hardware interfaces do not support all of the speeds listed here). Speeds above 38400 are supported on Series 700 only.

ispeed *number*	Set terminal input baud rate to *number*. If *number* is zero, the input baud rate is set to the value of the output baud rate.
ospeed *number*	Set terminal output baud rate to *number*. If *number* is zero, the modem control lines are released, which in turn disconnects the line.
hupcl (**-hupcl**)	Hang up (do not hang up) modem connection on last close.
hup (**-hup**)	Same as **hupcl** (**-hupcl**).
cstopb (**-cstopb**)	Use two (one) stop bits per character.
cread (**-cread**)	Enable (disable) the receiver.
crts (**-crts**)	Enable (disable) request-to-send.
clocal (**-clocal**)	Assume a line without (with) modem control.
loblk (**-loblk**)	Block (do not block) output from a noncurrent layer.

Input Modes

ignbrk (**-ignbrk**)	Ignore (do not ignore) break on input.
ienqak (**-ienqak**)	Enable (disable) ENQ-ACK Handshaking.

brkint (**-brkint**)	Signal (do not signal) INTR on break.
ignpar (**-ignpar**)	Ignore (do not ignore) parity errors.
parmrk (**-parmrk**)	Mark (do not mark) parity errors (see *termio*(7)).
inpck (**-inpck**)	Enable (disable) input parity checking.
istrip (**-istrip**)	Strip (do not strip) input characters to seven bits.
inlcr (**-inlcr**)	Map (do not map) newline character to carriage return (CR) on input.
igncr (**-igncr**)	Ignore (do not ignore) CR on input.
icrnl (**-icrnl**)	Map (do not map) CR to a newline character on input.
iuclc (**-iuclc**)	Map (do not map) uppercase alphabetic characters to lowercase on input.
ixon (**-ixon**)	Enable (disable) START/STOP output control. Output is stopped by sending an ASCII DC3 and started by sending an ASCII DC1.
ixany (**-ixany**)	Allow any character (only DC1) to restart output.
ixoff (**-ixoff**)	Request that the system send (not send) START/STOP characters when the input queue is nearly empty/full.
imaxbel (**-imaxbel**)	Echo (do not echo) BEL when the input line is too long.

Output Modes

opost (**-opost**)	Post-process output (do not post-process output; ignore all other output modes).
olcuc (**-olcuc**)	Map (do not map) lowercase alphabetics to uppercase on output.
onlcr (**-onlcr**)	Map (do not map) newline character to a carriage-return/newline character sequence on output.
ocrnl (**-ocrnl**)	Map (do not map) CR to newline character on output.
onocr (**-onocr**)	Do not (do) output CRs at column zero.
onlret (**-onlret**)	On the terminal, a newline character performs (does not perform) the CR function.
ofill (**-ofill**)	Use fill characters (use timing) for delays.
ofdel (**-ofdel**)	Fill characters are DELs (NULs).
cr0 cr1 cr2 cr3	Select style of delay for carriage returns (see *termio*(7)).
nl0 nl1	Select style of delay for newline characters (see *termio*(7)).
tab0 tab1 tab2 tab3	Select style of delay for horizontal tabs (see *termio*(7).
bs0 bs1	Select style of delay for backspaces (see *termio*(7)).
ff0 ff1	Select style of delay for form-feeds (see *termio*(7)).
vt0 vt1	Select style of delay for vertical tabs (see *termio*(7)).

Local Modes

isig (**-isig**)	Enable (disable) the checking of characters against the special control characters INTR and QUIT.
icanon (**-icanon**)	Enable (disable) canonical input (ERASE and KILL processing).
iexten (**-iexten**)	Enable (disable) any implementation-defined special control characters not currently controlled by *icanon*, *isig*, or *ixon*.
xcase (**-xcase**)	Canonical (unprocessed) uppercase and lowercase presentation.
echo (**-echo**)	Echo back (do not echo back) every character typed.
echoe (**-echoe**)	Echo (do not echo) ERASE character as a backspace-space-backspace string. Note: this mode erases the ERASEed character on many CRT terminals. However, it does *not* keep track of column position and, as a result, may not correctly erase escaped characters, tabs, and backspaces.

echok (**-echok**)	Echo (do not echo) a newline character after a KILL character.
lfkc (**-lfkc**)	(obsolete) Same as **echok** (**-echok**).
echonl (**-echonl**)	Echo (do not echo) newline character.
noflsh (**-noflsh**)	Disable (enable) flush after INTR or QUIT.
echoctl (**-echoctl**)	Echo (do not echo) control characters as ^char, delete as ^?
echoprt (**-echoprt**)	Echo (do not echo) erase character as character is erased.
echoke (**-echoke**)	BS-SP-BS erase (do not BS-SP-BS erase) entire line on line kill.
flusho (**-flusho**)	Output is (is not) being flushed.
pendin (**-pendin**)	Retype (do not retype) pending output at next read or input character.
tostop (**-tostop**)	Enable (disable) generation of SIGTTOU signals when background jobs attempt output.

Hardware Flow Control Modes

The following options are reserved for use with those devices that support hardware flow control through the termiox interface. If the functionality is supported, this interface must be used.

rtsxoff (**-rtsxoff)**	enable (disable) RTS hardware flow control on input (see *termiox*(7))
ctsxon (**-ctsxon)**	enable (disable) CTS hardware flow control on output (see *termiox*(7))

Control Assignments

control-character c	Set *control-character* to *c*, where *control-character* is **erase**, **kill**, **intr**, **quit**, **eof**, **eol**, **eol2**, **werase**, **lnext**, **min**,**or time** (**min** and **time** are used with **-icanon**; see *termio*(7)). For systems that support job control, **susp** and **dsusp** characters can also be set. For systems that support shell layers (see *shl*(1)) **swtch** can also be set. If *c* is preceded by an (escaped from the shell) circumflex (**^**), the value used is the corresponding control character (for example, **^d** represents **Ctrl-d**); ^? is interpreted as DEL and ^- is interpreted as undefined.
line *i*	Set line discipline to *i* where the value of *i* ranges from zero through 127 decimal (See *termio*(7)).

Combination Modes

evenp or **parity**	Enable **parenb** and **cs7**.
oddp	Enable **parenb**, **cs7**, and **parodd**.
-parity, **-evenp**, or **-oddp**	Disable **parenb** and set **cs8**.
raw (**-raw** or **cooked**)	Enable (disable) raw input and output (no ERASE, KILL, INTR, QUIT, EOT, or output post processing). See WARNINGS.
nl (**-nl**)	Unset (set) **icrnl** and **onlcr** **.** In addition **-nl** unsets **inlcr**, **igncr**, **ocrnl**, and **onlret**.
lcase (**-lcase**)	Set (unset) **xcase**, **iuclc**, and **olcuc**.
LCASE (**-LCASE**)	Same as **lcase** (**-lcase**).
tabs (**-tabs** or **tab3**)	Preserve (expand to spaces) tabs when printing.
ek	Reset ERASE and KILL characters back to default **#** and **@**.
sane	Reset all modes to some reasonable values.
term	Set all modes suitable for the terminal type *term*, where *term* is one of **tty33**, **tty37**, **vt05**, **tn300**, **ti700**, **hp**, or **tek**.

Reporting Functions

size	Print terminal window size to standard output in a rows-and-columns format.

Control Character Default Assignments

The control characters are assigned default values when the terminal port is opened, see *termio*(7). The default values used are those specified by the System V Interface Definition, Third Edition (SVID3), except for the **werase** and **lnext** control characters, which are set to **_POSIX_VDISABLE** to maintain binary compatibility with previous releases of HP-UX.

The default values for the control characters may be changed by a user with root capability by using **stty** and redirecting stdin to the device **/dev/ttyconf**. Any of the four command forms specified in the Description section above may be used. However, only the control character defaults will be reported or altered. It will have no effect on the defaults for any of the other modes.

Note that these defaults will be used for all terminal ports in the system, except the system console, and the changes will not become effective for a particular port until it is (re)opened. The default control character assignment will not work with the system console because the system console is never closed while the system is running, and therefore cannot be reopened.

Care should be exercised when re-assigning the control character defaults. Control character values should be tested with applications before assigning them as a default value.

EXTERNAL INFLUENCES

Environment Variables

LC_CTYPE determines the valid control characters for printing.

If LC_CTYPE is not specified in the environment or is set to the empty string, the value of LANG is used as a default for each unspecified or empty variable. If LANG is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of LANG. If any internationalization variable contains an invalid setting, **stty** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single-byte character code sets are supported.

EXAMPLES

The command:

```
stty kill '^X' intr '^C'
```

sets the delete-line character to **^X** (Ctrl-X) and the interrupt character to **^C**. This command is usually found in the **.login** or **.profile** file so that **^X** and **^C** need not be set by the user at each login session.

The command:

```
stty kill '^X' intr '^C' werase '^W' </dev/ttyconf
```

sets the default values for the delete-line character to **^X** (Ctrl-X), the interrupt character to **^C**, and the word erase character to **^W**. Any terminal port opened after this command is issued will see these new default values for the **kill**, **intr**, and **werase** control characters.

WARNINGS

Use of **raw** mode produces certain side effects which have varied from release to release in the past and may vary in the future. Relying on these side effects in applications can lead to unreliable results in the future and is therefore discouraged.

DEPENDENCIES

Refer to the DEPENDENCIES section of *termio*(7) for a further description of capabilities that are not supported.

SEE ALSO

shl(1), tabs(1), ioctl(2), termio(7), termiox(7).

STANDARDS CONFORMANCE

stty: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

NAME

su - switch user

SYNOPSIS

su [**-**] [*username* [*arguments*]]

DESCRIPTION

The **su** (set user or superuser) command allows one user to become another user without logging out.

username is the name of a user defined in the **/etc/passwd** file (see *passwd*(4)). The default name is **root** (that is, superuser).

To use **su**, the appropriate password must be supplied unless the current user is superuser. If a valid password is entered, **su** executes a new shell with the real and effective user ID, real and effective group ID, and group access list set to that of the specified user. The new shell is the one specified in the shell field of the new user's entry in the password file, **/etc/passwd**.

The *arguments* are passed along to the new shell for execution, permitting the user to run shell procedures with the new user's privileges.

When exiting from the new shell, the previous *username* and environment are restored.

If the **-** option is specified, the new shell starts up as if the new user had initiated a new login session. Exceptions are as follows:

- The **HOME** variable is reset to the new user's home directory.
- If the new user name is **root**, the path and prompt variables are reset:

  ```
  PATH=/usr/bin:/usr/sbin:/sbin
  PS1=#
  ```

  For other user names:

  ```
  PATH=/usr/bin
  PS1=$
  ```

- The **TERM** variable is retained.
- The rest of the environment is deleted and reset to the login state. However, the login files are normally executed anyway, usually restoring the expected value of **PATH** and other variables.

If the **-** option is omitted, the new shell starts as if a subshell was invoked. Exceptions are as follows:

- If the new user name is **root**, the path and prompt variables are reset:

  ```
  PATH=/usr/bin:/usr/sbin:/sbin
  PS1=#
  ```

- The previously defined **HOME** and **ENV** environment variables are removed.
- The rest of the environment is retained.

If the shell specified in **/etc/passwd** is **/usr/bin/sh**, **su** sets the value of parameter **0** in the new shell (referenced as **$0**) to **su**. If the **-** option of the **su** command is specified, **su** sets parameter **0** to **-su**.

If the shell specified in **/etc/passwd** is not **/usr/bin/sh**, **su** sets the value of parameter **0** in the new shell to *shellname*. If the **-** option of the **su** command is specified, **su** sets parameter **0** to **-***shellname*. For example, if the Korn shell is invoked, the value of *shellname* will be either **ksh** or **-ksh**.

By comparison, the **login** command always sets parameter **0** to **-***shellname*.

All attempts to become another user are logged in **/var/adm/sulog**, including failures. Successful attempts are flagged with **+**; failures, with **-**. They are also logged with **syslog()** (see *syslog*(3C)).

HP-UX Smart Card Login

If the user account is configured to use a Smart Card, the user password is stored in the card. This password has characteristics identical to a normal password stored on the system.

In order to **su** using a Smart Card account, the Smart Card from the destination user account must be inserted into the Smart Card reader. The user is prompted for a PIN instead of a password during authentication.

```
Enter PIN:
```

The password is retrieved automatically from the Smart Card when a valid PIN is entered. Therefore, it is not necessary to know the password, only the PIN.

The card is locked if an incorrect PIN is entered three consecutive times. It may be unlocked only by the card issuer.

SECURITY FEATURES

Except for user **root**, users on a trusted system cannot use **su** to change to an account that has been locked because of expired passwords or other access restrictions.

EXTERNAL INFLUENCES

Environment Variables

HOME	User's home directory
LANG	The language in which messages are displayed. If **LANG** is not specified or is null, it defaults to **C** (see *lang*(5)). If any internationalization variable contains an invalid setting, all internationalization variables default to **C** (see *environ*(5)).
LOGNAME	User's login name
PATH	Command name search path
PS1	Default prompt
SHELL	Name of the user's shell

International Code Set Support

Characters in the 7-bit US-ASCII code sets are supported in login names (see *ascii*(5)).

EXAMPLES

Become user **bin** while retaining the previously exported environment:

```
su bin
```

Become user **bin** but change the environment to what would be expected if **bin** had originally logged in:

```
su - bin
```

Execute *command* and its *arguments* using the temporary environment and permissions of user **bin**:

```
su - bin -c command arguments
```

WARNINGS

After a valid password is supplied, **su** uses information from **/etc/passwd** and **/etc/logingroup** to determine the user's group ID and group access list. If **/etc/group** is linked to **/etc/logingroup**, and group membership for the user trying to log in is managed by the Network Information Service (NIS), and no NIS server is able to respond, **su** waits until a server does respond.

In normal operation, root is able to **su** to another user's account without being prompted for a password. However, DCE (Distributed Computing Environment) credentials for a user cannot be obtained without that user's password. Therefore, if DCE is being used as the authentication mechanism, and root wants to **su** to another user's account and get DCE credentials for that user, the **-d** flag must be specified. With this flag set, root will be prompted for the user's password and should supply that user's password at the prompt. For example:

```
su -d DCEPrincipalName
```

The **-d** flag cannot be used with **-c** flag.

DEPENDENCIES

Pluggable Authentication Modules (PAM)

PAM is an Open Group standard for user authentication, password modification, and account validation. In particular, **pam_authenticate()** is invoked to perform all functions related to **su**. This includes password retrieval, account validation, and error message displays.

FILES

$HOME/.profile	User's profile

`/etc/logingroup`	System's default group access list file
`/etc/passwd`	System's password file
`/etc/profile`	System's profile
`/var/adm/sulog`	Log of all attempts

SEE ALSO

env(1), login(1), sh(1), initgroups(3C), syslog(3C), group(4), passwd(4), profile(4), environ(5).

Pluggable Authentication Modules (PAM)

pam_acct_mgmt(3), pam_authenticate(3).

HP-UX Smart Card Login

scpin(1).

STANDARDS CONFORMANCE

`su`: SVID2, SVID3, XPG2

NAME

sum - print checksum and block or byte count of file(s)

SYNOPSIS

sum [**-r**] [**-p**] [*file* ...]

Remarks

sum is obsolescent and should not be used in new applications that are intended to be portable between systems. Use **cksum** instead (see *cksum*(1)).

DESCRIPTION

sum calculates and prints to standard output a checksum for each named file, and also prints the size of the file in 512 byte blocks, rounded up.

The default algorithm is a 16-bit sum of the bytes in which overflow is ignored. Alternate algorithms can be selected with the **-r** and **-p** options.

Standard input is used if no file names are given.

sum is typically used to verify data integrity when copying files between systems.

Options

sum recognizes the following options:

-r	Use an alternate algorithm in which the 16-bit sum is right rotated with each byte in computing the checksum.
-p	Use the 32-bit cyclical redundancy check (CRC) algorithm used by **cksum**.

RETURN VALUE

sum returns the following values upon completion:

0	All files were processed successfully.
>0	One or more files could not be read or some other error occurred.

If an inaccessible file is encountered, **sum** continues processing any remaining files, but the final exit status is affected.

DIAGNOSTICS

Read error conditions are indistinguishable from end of file on most devices; check the block or byte count.

WARNINGS

This command is likely to be withdrawn from X/Open standards. Applications using this command might not be portable to other vendors' platforms. The usage of *cksum*(1) is recommended.

SEE ALSO

cksum(1), wc(1), pdf(4).

STANDARDS CONFORMANCE

sum: SVID2, SVID3, XPG2, XPG3

NAME

tabs - set tabs on a terminal

SYNOPSIS

tabs [*tabspec*] [**+m** *n*] [**-T** *type*]

DESCRIPTION

tabs sets the tab stops on the user's terminal according to the tab specification *tabspec*, after clearing any previous settings. The user's terminal must have remotely-settable hardware tabs.

If you are using a non-HP terminal, you should keep in mind that behavior will vary for some tab settings.

Four types of tab specification are accepted for *tabspec*: "canned", repetitive, arbitrary, and file. If no **tabspec** is given, the default value is **-8**; i.e., UNIX "standard" tabs. The lowest column number is 1. Note that for *tabs*, column 1 always refers to the left-most column on a terminal, even one whose column markers begin at 0.

-*code* Gives the name of one of a set of "canned" tabs. Recognized *code*s and their meanings are as follows:

-a 1,10,16,36,72
Assembler, IBM S/370, first format

-a2 1,10,16,40,72
Assembler, IBM S/370, second format

-c 1,8,12,16,20,55
COBOL, normal format

-c2 1,6,10,14,49
COBOL compact format (columns 1-6 omitted). Using this code, the first typed character corresponds to card column 7, one space gets you to column 8, and a tab reaches column 12. Files using this tab setup should have **tabs** specify a format specification file as defined by **--***file* below. The *file* should have the following format specification:

```
<:t-c2 m6 s66 d:>
```

-c3 1,6,10,14,18,22,26,30,34,38,42,46,50,54,58,62,67
COBOL compact format (columns 1-6 omitted), with more tabs than **-c2.** This is the recommended format for COBOL. The appropriate format specification is:

```
<:t-c3 m6 s66 d:>
```

-f 1,7,11,15,19,23
FORTRAN

-p 1,5,9,13,17,21,25,29,33,37,41,45,49,53,57,61
PL/I

-s 1,10,55
SNOBOL

-u 1,12,20,44
UNIVAC 1100 Assembler

In addition to these "canned" formats, three other types exist:

-*n* A repetitive specification requests tabs at columns 1+*n*, 1+2×*n*, etc. Of particular importance is the value **-8**: this represents the UNIX "standard" tab setting, and is the most likely tab setting to be found at a terminal. Another special case is the value **-0**, implying no tabs at all.

*n1***,***n2***,**... The arbitrary format permits the user to type any chosen set of numbers, separated by commas, in ascending order. Up to 40 numbers are allowed. If any number (except the first one) is preceded by a plus sign, it is taken as an increment to be added to the previous value. Thus, the tab lists 1,10,20,30 and 1,10,+10,+10 are considered identical.

--*file* If the name of a file is given, **tabs** reads the first line of the file, searching for a format specification. If it finds one there, it sets the tab stops according to it, otherwise it sets them as **-8**. This type of specification can be used to ensure that a tabbed file is printed with correct tab settings, and is suitable for use with the **pr** command (see *pr*(1)):

```
tabs -- file; pr file
```

Any of the following can be used also; if a given option occurs more than once, the last value given takes effect:

-T*type* **tabs** usually needs to know the type of terminal in order to set tabs and always needs to know the type to set margins. *type* is a name listed in *term*(5). If no **-T** option is supplied, **tabs** searches for the **$TERM** value in the *environment* (see *environ*(5)). If **TERM** is not defined in the environment, **tabs** tries a sequence that will work for many terminals.

+m*n* The margin argument can be used for some terminals. It causes all tabs to be moved over *n* columns by making column *n+1* the left margin. If **+m** is given without a value of *n*, the value assumed is 10. The normal (left-most) margin on most terminals is obtained by **+m0**. The margin for most terminals is reset only when the **+m** option is given explicitly.

Tab and margin setting is performed via the standard output.

EXTERNAL INFLUENCES

Environment Variables

LC_CTYPE determines the interpretation of text within file as single- and/or multi-byte characters.

LC_MESSAGES determines the language in which messages are displayed.

If **LC_CTYPE** or **LC_MESSAGES** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**.

If any internationalization variable contains an invalid setting, **tabs** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported.

DIAGNOSTICS

illegal tabs
Arbitrary tabs are ordered incorrectly.

illegal increment
A zero or missing increment found in an arbitrary specification.

unknown tab code
A “canned” code cannot be found.

can’t open
--*file* option was used and file cannot be opened.

file indirection
--*file* option was used and the specification in that file points to yet another file. Indirection of this form is not permitted.

WARNINGS

There is no consistency among different terminals regarding ways of clearing tabs and setting the left margin.

It is generally impossible to usefully change the left margin without also setting tabs.

tabs clears only 20 tabs (on terminals requiring a long sequence), but is willing to set 64.

SEE ALSO

nroff(1), pr(1), tset(1), environ(5), term(5).

STANDARDS CONFORMANCE

tabs: SVID2, SVID3, XPG2, XPG3, XPG4

NAME

tail - deliver the last part of a file

SYNOPSIS

tail [**-f**] [**-b** *number*] [*file*]

tail [**-f**] [**-c** *number*] [*file*]

tail [**-f**] [**-n** *number*] [*file*]

Obsolescent:

tail [±[*number*]][**l**|**b**|**c**] [**-f**] [*file*]

DESCRIPTION

tail copies the named *file* to the standard output beginning at a designated place. If no *file* is named, standard input is used.

Command Forms

tail can be used in three forms as indicated above:

tail -b *number...*	Copy file starting at *number* blocks from end or beginning of file.
tail -c *number...*	Copy file starting at *number* bytes from end or beginning of file.
tail -n *number...* **tail** *number...*	Copy file starting at *number* lines from end or beginning of file.

tail with no options specified is equivalent to **tail -n 10**... .

Options and Command-Line Arguments

tail recognizes the following options and command-line arguments:

-f	Follow option. If the input file is a regular file or if *file* specifies a FIFO, do not terminate after the last line of the input file has been copied, but read and copy further bytes from the input file when they become available (**tail** enters an endless loop wherein it sleeps for one second then attempts to read and copy further records from the input file). This is useful when monitoring text being written to a file by another process. If no *file* argument is specified and the input is a pipe (FIFO), the **-f** option is ignored.
number	Decimal integer indicating quantity of output to be copied, measured in units specified by accompanying option. If *number* is preceded by a **+** character, copy operation starts *number* units from beginning of file. If *number* is preceded by a **-** character or the option name, copy operation starts *number* units from end of file. If *number* is not preceded by a **b**, **c**, or **n** option, **-n** is assumed. If both the option and *number* are not specified, **-n 10** is assumed.
-b *number*	Copy file beginning *number* 512-byte blocks from end or beginning of file. If *number* is not specified, **-b 10** is assumed. See *number* description above.
-c *number*	Copy file beginning *number* bytes from end or beginning of file. If *number* is not specified, **-c 10** is assumed. See *number* description above.
-n *number*	Copy file beginning *number* lines from end or beginning of file. If *number* is not specified, **-n 10** is assumed. See *number* description above.
file	Name of file to be copied. If not specified, the standard input is used.

If the **-c** option is specified, the input file can contain arbitrary data. Otherwise, the input file should be a text file.

Obsolescent Form

In the obsolescent form, option letters can be concatenated after the *number* argument to select blocks, bytes, or lines. If this syntax is used, ±*number* must be the first argument given. If *number* is not specified, −10 is assumed. This version is provided for backward compatibility only. The forms discussed previously are recommended for portability.

EXTERNAL INFLUENCES

Environment Variables

LC_CTYPE determines the locale for the interpretation of sequences of bytes of text data as characters (e.g., single- versus multibyte characters in arguments and input files).

LC_MESSAGES determines the language in which messages are displayed.

If **LC_CTYPE** or **LC_MESSAGES** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**.

If any internationalization variable contains an invalid setting, **tail** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported. However, the **b** and **c** options can break multi-byte characters and should be used with caution in a multi-byte locale environment.

EXAMPLES

Print the last three lines in file **file1** to the standard output, and leave **tail** in “follow” mode:

```
tail -fn 3 file1
tail -3 -f file1
```

Print the last 15 bytes of file **logfile** followed by any lines that are appended to **logfile** after **tail** is initiated until it is killed:

```
tail -fc15 logfile
tail -f -c 15 logfile
```

Three ways to print an entire file:

```
tail -b +1 file
tail -c +1 file
tail -n +1 file
```

WARNINGS

Tails relative to end-of-file are stored in a 20-Kbyte buffer, and thus are limited in length. Therefore, be wary of the results when piping output from other commands into **tail**.

Various kinds of anomalous behavior may occur with character special files.

SEE ALSO

dd(1), head(1).

STANDARDS CONFORMANCE

tail: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

NAME

talk - talk to another user

SYNOPSIS

talk *talk_party* [*ttyname*]

DESCRIPTION

The **talk** utility is a two-way, screen-oriented communication program.

The command argument *talk_party* can take one the following forms:

user
user@*host*
host!*user*
host:*user*
host.*user*

where *user* is a login name and *host* is a host name.

The optional command argument, *ttyname*, can be used to specify the terminal to be used when contacting a user who is logged in more than once. In absence of this argument, **talk** will try to contact the user on the user's most recently used terminal.

When first invoked, **talk** sends the following message to the party it tries to connect to (*callee*):

Message from Talk_Daemon@*callee_host*
...
talk: connection requested by *caller*@*caller_host*
talk: respond with: talk *caller*@*caller_host*

At this point, the recipient of the message can reply by typing:

talk *caller*@*caller_host*

Once communication is established, the two parties can type simultaneously, with their output displayed in separate regions of the screen. Characters are processed as follows:

- Typing characters from LC_CTYPE classifications **print** or **space** will cause those characters to be sent to the recipient's terminal.
- Typing <control>-L will cause the sender's screen to be refreshed.
- Typing the erase, kill or kill word character will delete the last character, line or word on the sender's terminal, with the action propagated to the recipient's terminal.
- Typing the interrupt character will terminate the local *talk* utility. Once the *talk* session has been terminated on one side, the other side of the *talk* session will be notified that the session has been terminated and will be able to do nothing except exit.
- Other non-printable characters typed on the sender's terminal are converted to printable characters before they are sent to the recipient's terminal.

Permission to be a recipient of a **talk** message can be denied or granted by using the **mesg** utility. However, a user may need other privileges to be able to access other users' terminals. The **talk** utility will fail when the user lacks the appropriate privileges.

SEE ALSO

mesg(1), who(1), write(1).

STANDARDS CONFORMANCE

talk: XPG4

NAME

tar - tape file archiver

SYNOPSIS

tar [**-**]*key* [*arg* ...] [*file* | **-C** *directory*] ...

DESCRIPTION

The **tar** command saves and restores archives of files on a magnetic tape, a flexible disk, or a regular file. The default archive file is **/dev/rmt/0m**. See the **-f** option below. Its actions are controlled by the *key* argument.

Arguments

key is a string of characters containing an optional version letter, exactly one function letter, and possibly one or more function modifiers, specified in any order. Whitespace is not permitted in *key*. The *key* string can be preceded by a hyphen (**-**), as when specifying options in other HP-UX commands, but it is not necessary.

arg ... The **b** and **f** function modifiers each require an *arg* argument (see below). If both **b** and **f** are specified, the order of the *arg* arguments must match the order of the modifiers. If specified, the *arg* arguments must be separated from the key and each other by whitespace.

file specifies a file being saved or restored. If *file* is a directory name, it refers to the files and (recursively) the subdirectories contained in that directory.

-C *directory* causes **tar** to perform a **chdir()** to *directory* (see *chdir*(2)). Subsequent *file* and **-C** *directory* arguments must be relative to *directory*. This allows multiple directories not related by a close or common parent to be archived using short relative path names.

The value of *file* is stored in the archive. The value of *directory* is not stored.

Version Keys

The version portion of the *key* determines the format in which **tar** writes the archive. **tar** can read either format regardless of the version. The version is specified by one of the following letters:

N Write a POSIX format archive. This format allows file names of up to 256 characters in length, and correctly archives and restores the following file types: regular files, character and block special devices, links, symbolic links, directories, and FIFO special files. It also stores the user and group name of each file and attempts to use these names to determine the user-ID and group-ID of a file when restoring it with the **p** function modifier. This is the default format.

O Write a pre-POSIX format archive.

Function Keys

The function portion of the *key* is specified by exactly one of the following letters:

c Create a new archive. Write from the beginning of the archive instead of appending after the last file. Any previous information in the archive is overwritten.

r Add the named *file* to the end of the archive. The same blocking factor used to create the archive must be used to append to it. This option cannot be used if the archive is a tape.

t List the names of all the files in the archive. Adding the **v** function modifier expands this listing to include the file modes and owner numbers. The names of all files are listed each time they occur on the tape.

u Add any named *file* to the archive if it is not already present or has been modified since it was last written in the archive. The same blocking factor used to create the archive must be used to update it.

x Extract the named *file* from the archive and restore it to the system. If a named *file* matches a directory whose contents were written to the archive, this directory is (recursively) extracted. If a named *file* on tape does not exist on the system, the *file* is created as follows:

- The user, group, and other protections are restored from the tape.
- The modification time is restored from the tape unless the **m** function modifier is specified.

- The file user ID and group ID are normally those of the restoring process.
- The set-user-ID, set-group-ID, and sticky bits are not set automatically. The **o** and **p** function modifiers control the restoration of protection; see below for more details.

If the files exist, their modes are not changed, but the set-user-id, set-group-id and sticky bits are cleared. If no *file* argument is given, the entire content of the archive is extracted. Note that if several files with the same name are on the archive, the last one overwrites all earlier ones.

Function Modifier Keys

The following function modifiers can be used in addition to the function letters listed above (note that some modifiers are incompatible with some functions):

A Suppress warning messages that **tar** did not archive a file's access control list. By default, **tar** writes a warning message for each file with optional ACL entries.

b Use the next *arg* argument as the blocking factor for archive records. The default is 20; the maximum is at least 20. However, if the **f -** modifier is used to specify standard input, the default blocking factor is 1.

The blocking factor is determined automatically when reading nine-track tapes (key letters **x** and **t**). On nine-track tapes, the physical tape record length is the same as the block size. The block size is defined as the logical record size times the blocking factor (number of logical records per block).

The blocking factor must be specified when reading flexible disks and cartridge tapes if they were written with a blocking factor other than the default.

If a **tar** file is read using a blocking factor not equal to the one used when the file was written, an error may occur at the end of the file but there may or may not be an actual error in the read. To prevent this problem, a blocking factor of **1** can be used, although performance may be reduced somewhat.

tar writes logical records of 512 bytes, independent of how logical records may be defined elsewhere by other programs (such as variable-length records (lines) within an ASCII text file).

e Fail if the extent attributes are present in the files to be archived. If **tar** fails for this reason, the partially created destination file is not be removed.

f Use the next *arg* argument as the name of the archive instead of the default, **/dev/rmt/0m**. If the name of the file is **-**, **tar** writes to standard output or reads from standard input, whichever is appropriate, and the default blocking factor becomes 1. Thus, **tar** can be used as the head or tail of a pipeline (see EXAMPLES).

h Force **tar** to follow symbolic links as if they were normal files or directories. Normally, **tar** does not follow symbolic links.

l Tell **tar** to complain if it cannot resolve all of the links to the files being saved. If **l** is not specified, no error messages are printed.

m Tell **tar** not to restore the modification time written on the archive. The modification time of the file will be the time of extraction.

o Suppress writing certain directory information that older versions of **tar** cannot handle on input. **tar** normally writes information specifying owners and modes of directories in the archive. Earlier versions of **tar**, when encountering this information, give error messages of the form:

name **- cannot create**

When **o** is used for reading, it causes the extracted *file* to take on the user and group IDs of the user running the program rather than those on the tape. This is the default for the ordinary user and can be overridden, to the extent that system protections allow, by using the **p** function modifier.

p Cause *file* to be restored to the original modes and ownerships written on the archive, if possible. This is the default for the superuser, and can be overridden by the **o** function modifier. If system protections prevent the ordinary user from executing **chown()**, the error is ignored, and the ownership is set to that of the restoring process (see *chown*(2)). The set-user-id, set-group-id, and sticky bit information are restored as allowed by the protections defined by **chmod()** if the **chown()** operation above succeeds.

n d Specify a particular nine-track tape drive and density, where *n* is a tape drive number: **0**–**7**, and *d* is the density: **l** = low (800 bpi); **m** = medium (1600 bpi); **h** = high (6250 bpi). This modifier selects the drive on which the nine-track tape is mounted. The default is **0m**.

v Normally, **tar** does its work silently. The **v** (verbose) function modifier causes **tar** to type the name of each file it treats, preceded by the function letter. With the **t** function, **v** gives more information about the archive entries than just the name.

V Same as the **v** function modifier except that, when using the **t** option, **tar** also prints out a letter indicating the type of the archived file.

w Cause **tar** to print the action being taken, followed by the name of the file, then wait for the user's confirmation. If the user answers **y**, the action is performed. Any other input means "no".

When end-of-tape is reached, **tar** prompts the user for a new special file and continues.

If a nine-track tape drive is used as the output device, it must be configured in Berkeley-compatibility mode (see *mt*(7)).

EXTERNAL INFLUENCES

Environment Variables

LC_TIME determines the format and contents of date and time strings output when listing the contents of an archive with the **-v** option.

LANG determines the language equivalent of **y** (for yes/no queries).

If **LC_TIME** is not specified in the environment or is set to the empty string, the value of **LANG** is used as the default.

If **LANG** is not specified or is set to the empty string, it defaults to "C" (see *lang*(5)).

If any internationalization variable contains an invalid setting, **tar** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multibyte character code sets are supported.

ERRORS

tar issues self-explanatory messages about bad key characters, tape read/write errors, and if not enough memory is available to hold the link tables.

EXAMPLES

Create a new archive on **/dev/rfd.0** and copy **file1** and **file2** onto it, using the default blocking factor of 20. The *key* is made up of one function letter (**c**) and two function modifiers (**v** and **f**):

```
tar cvf /dev/rfd.0 file1 file2
```

Archive files from **/usr/include** and **/etc**:

```
tar cv -C /usr/include -C /etc
```

Use **tar** in a pipeline to copy the entire file system hierarchy under *fromdir* to *todir*:

```
cd fromdir ; tar cf - . | ( cd todir ; tar xf -i )
```

Archive all files and directories in directory **my_project** in the current directory to a file called **my_project.TAR**, also in the current directory:

```
tar -cvf my_project.TAR my_project
```

WARNINGS

Because of industry standards and interoperability goals, **tar** does not support the archival of files larger than 2GB or files that have user/group IDs greater than 60K. Files with user/group IDs greater than 60K are archived and restored under the user/group ID of the current process.

The default version has changed from **O** to **N**, beginning with HP-UX Release 8.0.

Due to internal limitations in the header structure, not all file names of fewer than 256 characters fit when using the **N** version key. If a file name does not fit, **tar** prints a message and does not archive the file.

Link names are still limited to 100 characters when using the **N** version key.

There is no way to ask for the *n*-th occurrence of a file.

Tape errors are handled ungracefully.

The **u** function key can be slow.

If the archive is a file on disk, flexible disk, or cartridge tape, and if the blocking factor specified on output is not the default, the same blocking factor must be specified on input, because the blocking factor is not explicitly stored in the archive. Updating or appending to the archive without following this rule can destroy it.

Some previous versions of **tar** have claimed to support the selective listing of file names using the **t** function key with a list. This appears to be an error in the documentation because the capability does not appear in the original source code.

There is no way to restore an absolute path name to a relative position.

tar always pads information written to an archive up to the next multiple of the block size. Therefore, if you are creating a small archive and write out one block of information, **tar** reports that one block was written, but the actual size of the archive might be larger if the **b** function modifier is used.

Note that **tar c0m** is not the same as **tar cm0**.

Do not create archives on block special devices. Attempting to do so can causes excessive wear, leading to premature drive hardware failure.

DEPENDENCIES

Series 700/800

The **r** and **u** function keys are not supported on QIC or 8mm devices. If these options are used with QIC or 8mm devices, **tar** fails and displays the message:

```
tar: option not supported for this device
```

AUTHOR

tar was developed by AT&T, the University of California, Berkeley, HP, and POSIX.

FILES

/dev/rmt/*
/dev/rfd.*
/tmp/tar*

SEE ALSO

ar(1), cpio(1), acl(5), mt(7).

STANDARDS CONFORMANCE

tar: SVID2, SVID3, XPG2, XPG3

NAME

tbl - format tables for nroff

SYNOPSIS

tbl [**-TX**] [*file* ...]

DESCRIPTION

tbl is a preprocessor that formats tables for *nroff*(1). The input files are copied to the standard output, except for lines between **.TS** and **.TE** command lines, which are assumed to describe tables and are reformatted by **tbl**. (The **.TS** and **.TE** command lines are not altered by **tbl**).

.TS is followed by global options. The available global options are:

center	center the table (default is left-adjust);
expand	make the table as wide as the current line length;
box	enclose the table in a box;
doublebox	enclose the table in a double box;
allbox	enclose each item of the table in a box;
tab (*x***)**	use the character *x* instead of a tab to separate items in a line of input data.

The global options, if any, are terminated with a semi-colon (**;**).

Next come lines describing the format of each line of the table. Each such format line describes one line of the actual table, except that the last format line (which must end with a period) describes *all* remaining lines of the table. Each column of each line of the table is described by a single key-letter, optionally followed by specifiers that determine the font and point size of the corresponding item, indicate where vertical bars are to appear between columns, or determine column width, inter-column spacing, etc. The available key-letters are:

c	center item within the column;
r	right-adjust item within the column;
l	left-adjust item within the column;
n	numerically adjust item in the column: units positions of numbers are aligned vertically;
s	span previous item on the left into this column;
a	center longest line in this column, then left-adjust all other lines in this column with respect to that centered line;
^	span down previous entry in this column;
_	replace this entry with a horizontal line;
=	replace this entry with a double horizontal line.

The characters **B** and **I** stand for the bold (font position 3) and italic (font position 2) fonts, respectively; the character **|** indicates a vertical line between columns.

The format lines are followed by lines containing the actual data for the table, followed finally by **.TE**. Within such data lines, data items are normally separated by tab characters.

If a data line consists of only _ or **=**, a single or double line, respectively, is drawn across the table at that point; if a *single item* in a data line consists of only _ or **=**, then that item is replaced by a single or double line.

The **-TX** option forces **tbl** to use only full vertical line motions, making the output more suitable for devices that cannot generate partial vertical line motions (such as line printers).

If no file names are given as arguments (or if **-** is specified as the last argument), **tbl** reads the standard input, and thus can be used as a filter. When used with **neqn**, **tbl** should be used first to minimize the volume of data passed through pipes (see *neqn*(1)).

EXTERNAL INFLUENCES

Environment Variables

LC_CTYPE determines the interpretation of text as single- and/or multi-byte characters.

LC_NUMERIC determines the radix character used in numerical data.

LANG determines the language in which messages are displayed.

If **LC_CTYPE** or **LC_NUMERIC** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**. If any internationalization variable contains an invalid setting, **tbl** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported.

EXAMPLES

If we redefine the tab character to a semicolon, then the input:

```
.TS
center box tab(;) ;
cB s s
cI | cI s
^  | c  c
l  | n  n.
Household Population
_
Town;Households
;Number;Size
=
Bedminster;789;3.26
Bernards Twp.;3087;3.74
Bernardsville;2018;3.30
Bound Brook;3425;3.04
Bridgewater;7897;3.81
Far Hills;240;3.19
.TE
```

yields:

Household Population		
Town	*Households*	
	Number	Size
Bedminster	789	3.26
Bernards Twp.	3087	3.74
Bernardsville	2018	3.30
Bound Brook	3425	3.04
Bridgewater	7897	3.81
Far Hills	240	3.19

The **tbl** command is used most often with **nroff** and **col** (see *col*(1)). A common usage is:

tbl filename | nroff -m*macro_package_name* **| col**

t

WARNINGS

See WARNINGS under *nroff*(1).

SEE ALSO

col(1), mm(1), neqn(1), nroff(1), soelim(1), mm(5).

tbl tutorial in *Text Formatters Users Guide*.

NAME

tcio - Command Set 80 CS/80 Cartridge Tape Utility

SYNOPSIS

tcio -o[**dervVZ**] [**-l** *number* [**-n** *limit*]] [**-S** *buffersize*] [**-T** *tty*] *file*

tcio -i[**drvZ**] [**-l** *number* [**-n** *limit*]] [**-S** *buffersize*] [**-T** *tty*] *file*

tcio -u[**rvV**] [**-l** *number*] [**-m** *blocknumber*] *file*

DESCRIPTION

tcio is designed to optimize the data transfer rate between certain cartridge tape units and the controlling computer. When used in conjunction with other utilities (such as **cpio**) a significant improvement in throughput can be obtained, in addition to reducing the wear and tear on the tape cartridges and drives. With autochanger mechanisms, **tcio** provides the capability of loading a specified cartridge, or automatically switching to successive cartridges as needed. With the utility operation, **tcio** provides functions that are unique to cartridge tapes.

tcio commands take one of the following forms:

tcio -o (copy out) Reads the standard input and writes the data to the CS/80 Cartridge Tape Unit specified by *file*.

tcio -i (copy in) Reads the CS/80 Cartridge Tape Unit specified by *file* and writes the data to the standard output.

tcio -u (utility) Performs utility functions on the cartridge tape, such as unload, mark, and/or verify the cartridge.

In all cases, *file* must refer to a character special file associated with a CS/80 cartridge tape unit.

During input and output operations, **tcio** enables immediate report mode on cartridge tape units that support this mode (see DEPENDENCIES). During writing, this mode enables the drive to complete a write transaction with the host before the data has actually been written to the tape from the drive's buffer. This allows the host to start gathering data for the next write request while the data for the previous request is still in the process of being written. During reading, this mode enables the drive to read ahead after completing a host read request. This allows the drive to gather data for future read requests while the host is still processing data from the previous read request. Under favorable conditions, immediate report mode allows the drive to stream the tape continuously across multiple read/write requests, rather than having to reposition the tape between each read/write request. See *ct*(7) for more information.

By default, **tcio** writes a tape mark in the first block on each tape to prevent the tape from being image restored onto a disk. It also uses the last block on each tape to hold a flag indicating whether or not the tape is the last tape in a multi-tape sequence.

Options

Every **tcio** command must be followed by a **-o**, **-i**, or **-u** option to indicate the type of operation being performed. In addition, the following command options are recognized. They can be specified in any order, but all must precede the *file* name. Options without parameters can be listed individually (each preceded by a **-**) or grouped together. Options with parameters require the parameter, and must be listed individually.

-d Print a checksum to standard error. The checksum is a 32-bit unsigned addition of all bytes written to or read from the tape, providing an extra check of data validity (in addition to tape verification). The checksum value is only reported to the user, and is not written on the media; thus, the user must manually record and check it. The checksum is valid *only* if the same number of bytes are read from the tape as were written to it; in other words, the checksum as a data verification test is meaningless unless the **-e** option was used when writing the tape. This option is independent of the verbose modifier.

-e Cause a tape mark to be written on the nearest 1024-byte boundary following the end of the data. When a tape containing an end-of-data tape mark is read back, the read terminates upon encountering the tape mark. Thus, by using this option, the checksums generated by the input and output operations are guaranteed to agree.

-r Unload the tape from the drive. On autochanger units, the tape is returned to the magazine.

-v Verbose mode; prints information and error messages to standard error.

-V This option turns off tape verification. Some cartridge tape units (see DEPENDENCIES) provide hardware for verifying the data output to the tape (called "read-while-write"). For these units software-driven verification is somewhat redundant, and this option is suggested as a means of reducing wear on tape heads and transport mechanisms. However, read-while-write verification does not completely eliminate all risk of data loss, so software verification may still be desired in situations where data preservation is critically important.

For drives that do not have the read-while-write hardware, a separate verification operation is suggested. Thus, it is recommended that this option not be used with drives that do not support read-while-write.

-Z Prevents **tcio** from writing a file mark in the first and last blocks. This option should be used with care because a tape without a tape mark in block zero can be image-restored to a disk.

-l *number* This option is intended solely for autochanging tape drives. For input or output operations (**-i** or **-o**) the **-l** option selects the cartridge specified by *number* from the magazine as the first cartridge used in the transfer. For utility operations (**-u** option), **tcio** loads the cartridge specified by *number* into the drive. (Note: the autochanger must be in selective mode for the autochanger options to work properly.) Whitespace between **-l** and *number* is optional.

-m *blocknumber*
This option writes a tape mark on a tape at the specified block. A tape mark in block zero of the tape prevents it from being image-restored to a disk. Whitespace between **-m** and *blocknumber* is optional.

-n *limit* This option specifies the maximum number of cartridges to be allowed in a multitape transfer. It applies only to autochanger type units, and must be preceded by the **-l** option. Thus, **-l** starts the transfer by loading cartridge *number* and uses at most *limit* cartridges. If **-l** is specified without **-n**, **tcio** quietly assumes all remaining cartridges (in ascending order) in the magazine. Whitespace between **-n** and *limit* is optional.

-S *buffersize* Enable specification of buffer size. This option forces allocation of a block of memory to be used in reading or writing the tape. The size of the buffer in bytes is 1024 times the value specified for *buffersize*. If *buffersize* is less than 4, it is silently increased to 4. A *buffersize* greater than 64 is silently decreased to 64. If *buffersize* is not specified, **tcio** allocates a 64K-byte buffer. Whitespace between **-S** and *buffersize* is optional.

On tape units that support immediate report, a significant performance increase can often be obtained by using a smaller buffer. 8 Kbytes is the recommended buffer size for these units. On tape units that do not support the immediate report mode, or on tape units that share a controller with a disk (see DEPENDENCIES) that is simultaneously being accessed, an increase in performance can usually be obtained with a larger buffer. 64K bytes, the default, is the recommended buffer size for these units.

-T *tty* Specify *tty* as an alternative to **/dev/tty**. Normally **/dev/tty** is opened by **tcio** when terminal interaction is required. The specified file *tty* is opened instead of **/dev/tty**. Whitespace between **-T** and *tty* is optional. If no input device is available, use **/dev/null**.

t

EXAMPLES

Copy the contents of a directory into an archive:

```
ls | cpio -o | tcio -o /dev/rct/c4t1d0
```

Restore it:

```
tcio -i /dev/rct/c4t1d0 | cpio -i
```

Unload the cartridge from the drive (without verifying the tape):

```
tcio -urV /dev/rct/c4t1d0
```

Copy all files in the current directory to the tape specified by the device file > .CR /dev/rct/c4t1d0 . The device has a read-while-write head, so verify is turned off; a buffer size (option **-S)** of 8 blocks (8 Kbytes) is to be used:

```
ls | cpio -o | tcio -oV -S 8 /dev/rct/c4t1d0
```

Assume that the cartridge tape unit is an autochanger on controller 2, with 8 tapes in the magazine. Start writing with cartridge 3, and use at most 4 cartridges before prompting the user for additional media:

```
find usr -cpio | tcio -oV -S 8 -l 3 -n 4 /dev/rct/c2t0d0
```

DEPENDENCIES

HP 7941CT, HP 9144A, HP 9145A, and HP 35401

These cartridge tape devices contain read-while-write hardware and support immediate report mode.

HP 7942, HP 7946

These cartridge tape devices contain read-while-write hardware and support immediate report mode. Use of a small buffer size is not recommended with these shared-controller devices when simultaneous access to the disk is also required because the intervening disk accesses prevent proper tape streaming.

HP 7908, HP 7911, HP 7912, and HP 7914

These cartridge tape devices do not contain read-while-write hardware, and therefore do not support immediate report mode.

AUTHOR

tcio was developed by HP.

SEE ALSO

ct(7).

HP-UX System Administrator Manuals.

NAME
tee - pipe fitting

SYNOPSIS
tee [**-i**] [**-a**] [*file*] ...

DESCRIPTION
The **tee** command transcribes the standard input to the standard output and makes copies in the *files*.

OPTIONS
-i This option ignores interrupts.

-a This option appends the output to the *files* rather than overwriting the *files*.

EXTERNAL INFLUENCES
Environment Variables
LC_MESSAGES determines the language in which messages are displayed.

If **LC_MESSAGES** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**.

If any internationalization variable contains an invalid setting, **tee** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support
Single- and multi-byte character code sets are supported.

RETURN VALUE
The **tee** command returns zero upon successful completion, or nonzero if the command fails.

EXAMPLES
Write a list of users to the screen and also append the list to the file **hunt**:

```
who | tee -a hunt
```

STANDARDS CONFORMANCE
tee: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

NAME

telnet - user interface to the TELNET protocol

SYNOPSIS

telnet [[*options*]*host* [*port*]]

DESCRIPTION

telnet is used to communicate with another host using the TELNET protocol. If **telnet** is invoked without arguments, it enters command mode, indicated by its prompt (**telnet>**). In this mode, it accepts and executes the commands listed below. If **telnet** is invoked with arguments, it performs an **open** command (see below) with those arguments.

Once a connection has been opened, **telnet** enters an input mode. The input mode will be either "character at a time" or "line by line", depending on what the remote system supports.

In "character at a time" mode, most text typed is immediately sent to the remote host for processing.

In "line by line" mode, all text is echoed locally, and (normally) only completed lines are sent to the remote host. The "local echo character" (initially **^E**) can be used to turn off and on the local echo (this would mostly be used to enter passwords without the password being echoed).

In either mode, if the **localchars** toggle is TRUE (the default in line mode; see below), the user's **quit** and **intr** characters are trapped locally, and sent as TELNET protocol sequences to the remote side. There are options (see **toggle autoflush** and **toggle autosynch** below) which cause this action to flush subsequent output to the terminal (until the remote host acknowledges the TELNET sequence) and flush previous terminal input (in the case of **quit** and **intr**).

While connected to a remote host, **telnet** command mode can be entered by typing the **telnet** "escape character" (initially **^]**). When in command mode, the normal terminal editing conventions are available.

telnet supports eight-bit characters when communicating with the server on the remote host. To use eight-bit characters you may need to reconfigure your terminal or the remote host appropriately (see *stty*(1)). Furthermore, you may have to use the **binary** toggle to enable an 8-bit data stream between **telnet** and the remote host. Note that some remote hosts may not provide the necessary support for eight-bit characters.

If, at any time, **telnet** is unable to read from or write to the server over the connection, the message **Connection closed by foreign host.** is printed on standard error. **telnet** then exits with a value of 1.

telnet supports the TAC User ID (also known as the TAC Access Control System, or TACACS User ID) option. Enabling the option on a host server allows the user to **telnet** into that host without being prompted for a second login sequence. The TAC User ID option uses the same security mechanism as **rlogin** for authorizing access by remote hosts and users. The system administrator must enable the (telnetd) option only on systems which are designated as participating hosts. The system administrator must also assign to each user of TAC User ID the very same UID on every system for which he is allowed to use the feature. (See *telnetd*(1M) and the *Managing Systems and Workgroups* manual)

The following **telnet** options are available:

-8 Enable cs8 (8 bit transfer) on local tty.

-e c Set the **telnet** command mode escape character to be **^c** instead of its default value of **^]**.

-l Disable the TAC User ID option if enabled on the client, to cause the user to be prompted for login username and password. Omitting the -l option executes the default setting.

t

Commands

The following commands are available in command mode. You need only type enough of each command to uniquely identify it (this is also true for arguments to the **mode**, **set**, **toggle**, and **display** commands).

open *host* [*port*]

Open a connection to the named host at the indicated port. If no port is specified, **telnet** attempts to contact a TELNET server at the standard TELNET port. The hostname can be either the official name or an alias as understood by **gethostbyname()** (see *gethostent*(3N)), or an Internet address specified in the dot notation as described in *hosts*(4). If no hostname is given, **telnet** prompts for one.

close Close a TELNET session. If the session was started from command mode, **telnet** returns to command mode; otherwise **telnet** exits.

quit Close any open TELNET session and exit **telnet**. An end of file (in command mode) will also close a session and exit.

z Suspend **telnet**. If **telnet** is run from a shell that supports job control, (such as *csh*(1) or *ksh*(1)), the **z** command suspends the TELNET session and returns the user to the shell that invoked **telnet**. The job can then be resumed with the **fg** command (see *csh*(1) or *ksh*(1)).

mode *mode* Change **telnet**'s user input mode to *mode*, which can be **character** (for "character at a time" mode) or **line** (for "line by line" mode). The remote host is asked for permission to go into the requested mode. If the remote host is capable of entering that mode, the requested mode is entered. In **character** mode, **telnet** sends each character to the remote host as it is typed. In **line** mode, **telnet** gathers user input into lines and transmits each line to the remote host when the user types carriage return, linefeed, or EOF (normally **^D**; see *stty*(1)). Note that setting line-mode also sets local echo. Applications that expect to interpret user input character by character (such as **more**, **csh**, **ksh**, and **vi**) do not work correctly in line mode.

status Show current status of **telnet**. **telnet** reports the current escape character. If **telnet** is connected, it reports the host to which it is connected and the current **mode**. If **telnet** is not connected to a remote host, it reports **No connection.** Once **telnet** has been connected, it reports the local flow control toggle value.

display [*argument* ...]
Displays all or some of the **set** and **toggle** values (see below).

? [*command*] Get help. With no arguments, **telnet** prints a help summary. If a command is specified, **telnet** prints the help information available about that command only. Help information is limited to a one-line description of the command.

! [*shell_command*]
Shell escape. The **SHELL** environment variable is checked for the name of a shell to use to execute the command. If no *shell_command* is specified, a shell is started and connected to the user's terminal. If **SHELL** is undefined, **/usr/bin/sh** is used.

send *arguments*
Sends one or more special character sequences to the remote host. Each *argument* can have any of the following values (multiple *argument*s can be specified with each **send** command):

escape Sends the current **telnet** escape character (initially **^]**).

synch Sends the TELNET SYNCH sequence. This sequence causes the remote system to discard all previously typed (but not yet read) input. This sequence is sent as TCP urgent data (and may not work to some systems -- if it doesn't work, a lower case "r" may be echoed on the terminal).

brk Sends the TELNET BRK (Break) sequence, which may have significance to the remote system.

ip Sends the TELNET IP (Interrupt Process) sequence, which should cause the remote system to abort the currently running process.

ao Sends the TELNET AO (Abort Output) sequence, which should cause the remote system to flush all output *from* the remote system *to* the user's terminal.

ayt Sends the TELNET AYT (Are You There) sequence, to which the remote system may or may not choose to respond.

ec Sends the TELNET EC (Erase Character) sequence, which should cause the remote system to erase the last character entered.

el Sends the TELNET EL (Erase Line) sequence, which should cause the remote system to erase the line currently being entered.

ga Sends the TELNET GA (Go Ahead) sequence, which likely has no significance to the remote system.

nop Sends the TELNET NOP (No OPeration) sequence.

? Prints out help information for the **send** command.

set *variable_name value*
Set any one of a number of **telnet** variables to a specific value. The special value **off** turns off the function associated with the variable. The values of variables can be shown by using the **display** command. The following *variable_name*s can be specified:

echo
This is the value (initially **^E**) which, when in line-by-line mode, toggles between doing local echoing of entered characters (for normal processing), and suppressing echoing of entered characters (for entering, for example, a password).

escape
This is the **telnet** escape character (initially **^]**) which causes entry into **telnet** command mode (when connected to a remote system).

interrupt
If **telnet** is in **localchars** mode (see **toggle localchars** below) and the *interrupt* character is typed, a TELNET IP sequence (see **send ip** above) is sent to the remote host. The initial value for the interrupt character is taken to be the terminal's **intr** character.

quit
If **telnet** is in **localchars** mode (see **toggle localchars** below) and the **quit** character is typed, a TELNET BRK sequence (see **send brk** above) is sent to the remote host. The initial value for the quit character is taken to be the terminal's **quit** character.

flushoutput
If **telnet** is in **localchars** mode (see **toggle localchars** below) and the **flushoutput** character is typed, a TELNET AO sequence (see **send ao** above) is sent to the remote host. The initial value for the flush character is **^O**.

erase
If **telnet** is in **localchars** mode (see **toggle localchars** below), *and* if **telnet** is operating in character-at-a-time mode, then when this character is typed, a TELNET EC sequence (see **send ec** above) is sent to the remote system. The initial value for the erase character is taken to be the terminal's **erase** character.

kill
If **telnet** is in **localchars** mode (see **toggle localchars** below), *and* if **telnet** is operating in character-at-a-time mode, then when this character is typed, a TELNET EL sequence (see **send el** above) is sent to the remote system. The initial value for the kill character is taken to be the terminal's **kill** character.

eof If **telnet** is operating in line-by-line mode, entering this character as the first character on a line causes this character to be sent to the remote system. The initial value of the **eof** character is taken to be the terminal's **eof** character.

toggle *arguments ...*
Toggle (between TRUE and FALSE) various flags that control how **telnet** responds to events. More than one argument can be specified. The state of these flags can be shown by using the **display** command. Valid arguments are:

localchars
If TRUE, the **flush**, **interrupt**, **quit**, **erase**, and **kill** characters (see **set** above) are recognized locally, and transformed into appropriate TELNET control sequences (respectively **ao**, **ip**, **brk**, **ec**, and **el**; see **send** above). The initial value for this toggle is **TRUE** in line-by-line mode, and **FALSE** in character-at-a-time mode.

autoflush
If **autoflush** and **localchars** are both TRUE, whenever the **ao**, **intr**, or **quit** characters are recognized (and transformed into TELNET sequences –

t

see **set** above for details), **telnet** refuses to display any data on the user's terminal until the remote system acknowledges (via a TELNET *Timing Mark* option) that it has processed those TELNET sequences. The initial value for this toggle is TRUE.

autosynch
If **autosynch** and **localchars** are both TRUE, when either the **intr** or **quit** character is typed (see **set** above for descriptions of the **intr** and **quit** characters), the resulting TELNET sequence sent is followed by the TELNET SYNCH sequence. This procedure *should* cause the remote system to begin discarding all previously typed input until both of the TELNET sequences have been read and acted upon. The initial value of this toggle is FALSE.

binary
Enable or disable the TELNET BINARY option on both input and output. This option should be enabled in order to send and receive 8-bit characters to and from the TELNET server.

crlf
If TRUE, end-of-line sequences are sent as an ASCII carriage-return and line-feed pair. If FALSE, end-of-line sequences are sent as an ASCII carriage-return and NUL character pair. The initial value for this toggle is FALSE.

crmod
Toggle carriage return mode. When this mode is enabled, any carriage return characters received from the remote host are mapped into a carriage return and a line feed. This mode does not affect those characters typed by the user; only those received. This mode is only required for some hosts that require the client to do local echoing, but output "naked" carriage returns. The initial value for this toggle is FALSE.

echo
Toggle local echo mode or remote echo mode. In local echo mode, user input is echoed to the terminal by the local **telnet** before being transmitted to the remote host. In remote echo, any echoing of user input is done by the remote host. Applications that handle echoing of user input themselves, such as C shell, Korn shell, and **vi** (see *csh*(1), *ksh*(1), and *vi*(1)), do not work correctly with local echo.

options
Toggle viewing of TELNET options processing. When options viewing is enabled, all TELNET option negotiations are displayed. Options sent by **telnet** are displayed as **''SENT''**, while options received from the TELNET server are displayed as **''RCVD''**. The initial value for this toggle is FALSE.

netdata
Toggles the display of all network data (in hexadecimal format). The initial value for this toggle is FALSE.

? Displays the legal **toggle** commands.

t

RETURN VALUE

In the event of an error, or if the TELNET connection is closed by the remote host, **telnet** returns a value of 1. Otherwise it returns zero (0).

DIAGNOSTICS

The following diagnostic messages are displayed by **telnet**:

Error! Could not retrieve authentication type.
There are two authentication mechanisms used by TELNET. One authentication mechanism is based on Kerberos and the other is not. The type of authentication mechanism is obtained from a system file which is updated by **inetsvcs_sec**. If the system file on either the local host or the remote host does not contain known authentication types, the above error is displayed.

telnet/tcp: Unknown service
telnet was unable to find the TELNET service entry in the *services*(4) database.

hostname`: Unknown host`
: `telnet` was unable to map the host name to an Internet address. Your next step should be to contact the system administrator to check whether there is an entry for the remote host in the `hosts` database (see *hosts*(4)).

`?Invalid command`
: An invalid command was typed in `telnet` command mode.

system call`>:` ...
: An error occurred in the specified system call. See the appropriate manual entry for a description of the error.

AUTHOR

`telnet` was developed by the University of California, Berkeley.

SEE ALSO

csh(1), ksh(1), login(1), rlogin(1), stty(1), telnetd(1M), inetsvcs_sec(1M), hosts(4), services(4), termio(7).

t

NAME

telnet - user interface to the TELNET protocol

SYNOPSIS

telnet [[*options*] *host* [*port*]]

DESCRIPTION

telnet is used to communicate with another host using the TELNET protocol. If **telnet** is invoked without arguments, it enters command mode, indicated by its prompt (**telnet>**). In this mode, it accepts and executes the commands listed below. If **telnet** is invoked with arguments, it performs an **open** command (see below) with those arguments.

Once a connection has been opened, **telnet** enters an input mode. The input mode will be either "character at a time" or "line by line", depending on what the remote system supports.

In "character at a time" mode, most text typed is immediately sent to the remote host for processing.

In "line by line" mode, all text is echoed locally, and (normally) only completed lines are sent to the remote host. The "local echo character" (initially **^E**) can be used to turn off and on the local echo (this would mostly be used to enter passwords without the password being echoed).

In either mode, if the **localchars** toggle is TRUE (the default in line mode; see below), the user's **quit** and **intr** characters are trapped locally, and sent as TELNET protocol sequences to the remote side. There are options (see **toggle autoflush** and **toggle autosynch** below) which cause this action to flush subsequent output to the terminal (until the remote host acknowledges the TELNET sequence) and flush previous terminal input (in the case of **quit** and **intr**).

While connected to a remote host, **telnet** command mode can be entered by typing the **telnet** "escape character" (initially **^]**). When in command mode, the normal terminal editing conventions are available.

telnet supports eight-bit characters when communicating with the server on the remote host. To use eight-bit characters you may need to reconfigure your terminal or the remote host appropriately (see *stty*(1)). Furthermore, you may have to use the **binary** toggle to enable an 8-bit data stream between **telnet** and the remote host. Note that some remote hosts may not provide the necessary support for eight-bit characters.

If, at any time, **telnet** is unable to read from or write to the server over the connection, the message **Connection closed by foreign host.** is printed on standard error. **telnet** then exits with a value of 1.

By default (or by use of the **-a** option or the **-l** option), this Kerberos version of **telnet** behaves as a client which supports authentication based on Kerberos V5. As a Kerberos client, **telnet** will authenticate and authorize the user to access the remote system. (See *sis*(5) for details on Kerberos authentication and authorization.) However, it will not support integrity-checked or encrypted sessions.

telnet supports the TAC User ID (also known as the TAC Access Control System, or TACACS User ID) option. Enabling the option on a host server allows the user to **telnet** into that host without being prompted for a second login sequence. The TAC User ID option uses the same security mechanism as **rlogin** for authorizing access by remote hosts and users. The system administrator must enable the (telnetd) option only on systems which are designated as participating hosts. The system administrator must also assign to each user of TAC User ID the very same UID on every system for which he is allowed to use the feature. (See *telnetd*(1M) and the *Managing Systems and Workgroups* manual)

The following **telnet** options are available:

-8 Enable cs8 (8 bit transfer) on local tty.

-a Attempt automatic login into the Kerberos realm and disable the TAC User ID option. (Note: this is the default login mode.)

Sends the user name via the NAME subnegotiation of the Authentication option. The name used is that of the current user as returned by the USER environment variable. If this variable is not defined, the name used is that returned by *getpwnam*(3) if it agrees with the current user ID. Otherwise, it is the name associated with the user ID.

-e c Set the **telnet** command mode escape character to be **^c** instead of its default value of **^]**.

-l *user* Attempt automatic login into the Kerberos realm as the specified user and disable the TAC User ID option. The user name specified is sent via the NAME subnegotiation of the Authentication option. Omitting the -l option executes the default setting. Only one -l option is allowed.

-P Disable use of Kerberos authentication and authorization. When this option is specified, a password is required which is sent across the network in a readable form. (See *sis*(5).)

-f Allows local credentials to be forwarded to the remote system. Only one of -f or -F is allowed.

-F Allows local credentials to be forwarded to the remote system including any credentials that have already been forwarded into the local environment. Only one of -f or -F is allowed.

Commands

The following commands are available in command mode. You need only type enough of each command to uniquely identify it (this is also true for arguments to the **mode**, **set**, **toggle**, and **display** commands).

open [-l *user*] *host* [*port*]
Open a connection to the named host at the indicated port. If no port is specified, **telnet** attempts to contact a TELNET server at the standard TELNET port. The hostname can be either the official name or an alias as understood by **gethostbyname()** (see *gethostent*(3N)), or an Internet address specified in the dot notation as described in *hosts*(4). If no hostname is given, **telnet** prompts for one. The -l option can be used to specify the user name to use when automatically logging in to the remote system. Using this option disables the TAC User ID option.

close Close a TELNET session. If the session was started from command mode, **telnet** returns to command mode; otherwise **telnet** exits.

quit Close any open TELNET session and exit **telnet**. An end of file (in command mode) will also close a session and exit.

z Suspend **telnet**. If **telnet** is run from a shell that supports job control, (such as *csh*(1) or *ksh*(1)), the **z** command suspends the TELNET session and returns the user to the shell that invoked **telnet**. The job can then be resumed with the **fg** command (see *csh*(1) or *ksh*(1)).

mode *mode* Change **telnet**'s user input mode to *mode*, which can be **character** (for "character at a time" mode) or **line** (for "line by line" mode). The remote host is asked for permission to go into the requested mode. If the remote host is capable of entering that mode, the requested mode is entered. In **character** mode, **telnet** sends each character to the remote host as it is typed. In **line** mode, **telnet** gathers user input into lines and transmits each line to the remote host when the user types carriage return, linefeed, or EOF (normally **^D**; see *stty*(1)). Note that setting line-mode also sets local echo. Applications that expect to interpret user input character by character (such as **more**, **csh**, **ksh**, and **vi**) do not work correctly in line mode.

status Show current status of **telnet**. **telnet** reports the current escape character. If **telnet** is connected, it reports the host to which it is connected and the current **mode.** If **telnet** is not connected to a remote host, it reports **No connection.** Once **telnet** has been connected, it reports the local flow control toggle value.

display [*argument* ...]
Displays all or some of the **set** and **toggle** values (see below).

? [*command*] Get help. With no arguments, **telnet** prints a help summary. If a command is specified, **telnet** prints the help information available about that command only. Help information is limited to a one-line description of the command.

! [*shell_command*]
Shell escape. The **SHELL** environment variable is checked for the name of a shell to use to execute the command. If no *shell_command* is specified, a shell is started and connected to the user's terminal. If **SHELL** is undefined, **/usr/bin/sh** is used.

send *arguments*
Sends one or more special character sequences to the remote host. Each *argument* can have any of the following values (multiple *arguments* can be specified with each **send** command):

escape Sends the current **telnet** escape character (initially **^]**).

t

synch Sends the TELNET SYNCH sequence. This sequence causes the remote system to discard all previously typed (but not yet read) input. This sequence is sent as TCP urgent data (and may not work to some systems -- if it doesn't work, a lower case "r" may be echoed on the terminal).

brk Sends the TELNET BRK (Break) sequence, which may have significance to the remote system.

ip Sends the TELNET IP (Interrupt Process) sequence, which should cause the remote system to abort the currently running process.

ao Sends the TELNET AO (Abort Output) sequence, which should cause the remote system to flush all output *from* the remote system *to* the user's terminal.

ayt Sends the TELNET AYT (Are You There) sequence, to which the remote system may or may not choose to respond.

ec Sends the TELNET EC (Erase Character) sequence, which should cause the remote system to erase the last character entered.

el Sends the TELNET EL (Erase Line) sequence, which should cause the remote system to erase the line currently being entered.

ga Sends the TELNET GA (Go Ahead) sequence, which likely has no significance to the remote system.

nop Sends the TELNET NOP (No OPeration) sequence.

? Prints out help information for the **send** command.

set *variable_name value*

Set any one of a number of **telnet** variables to a specific value. The special value **off** turns off the function associated with the variable. The values of variables can be shown by using the **display** command. The following *variable_name*s can be specified:

echo

This is the value (initially **^E**) which, when in line-by-line mode, toggles between doing local echoing of entered characters (for normal processing), and suppressing echoing of entered characters (for entering, for example, a password).

escape

This is the **telnet** escape character (initially **^]**) which causes entry into **telnet** command mode (when connected to a remote system).

interrupt

If **telnet** is in **localchars** mode (see **toggle localchars** below) and the *interrupt* character is typed, a TELNET IP sequence (see **send ip** above) is sent to the remote host. The initial value for the interrupt character is taken to be the terminal's **intr** character.

t

quit

If **telnet** is in **localchars** mode (see **toggle localchars** below) and the **quit** character is typed, a TELNET BRK sequence (see **send brk** above) is sent to the remote host. The initial value for the quit character is taken to be the terminal's **quit** character.

flushoutput

If **telnet** is in **localchars** mode (see **toggle localchars** below) and the **flushoutput** character is typed, a TELNET AO sequence (see **send ao** above) is sent to the remote host. The initial value for the flush character is **^O**.

erase

If **telnet** is in **localchars** mode (see **toggle localchars** below), *and* if **telnet** is operating in character-at-a-time mode, then when this character is typed, a TELNET EC sequence (see **send ec** above) is sent to the remote system. The initial value for the erase character is taken to be the terminal's **erase** character.

kill

If **telnet** is in **localchars** mode (see **toggle localchars** below), *and* if **telnet** is operating in character-at-a-time mode, then when this character is typed, a TELNET EL sequence (see **send el** above) is sent to the remote system. The initial value for the kill character is taken to be the terminal's **kill** character.

eof If **telnet** is operating in line-by-line mode, entering this character as the first character on a line causes this character to be sent to the remote system. The initial value of the **eof** character is taken to be the terminal's **eof** character.

toggle *arguments ...*

Toggle (between TRUE and FALSE) various flags that control how **telnet** responds to events. More than one argument can be specified. The state of these flags can be shown by using the **display** command. Valid arguments are:

localchars

If TRUE, the **flush**, **interrupt**, **quit**, **erase**, and **kill** characters (see **set** above) are recognized locally, and transformed into appropriate TELNET control sequences (respectively **ao**, **ip**, **brk**, **ec**, and **el**; see **send** above). The initial value for this toggle is **TRUE** in line-by-line mode, and **FALSE** in character-at-a-time mode.

autoflush

If **autoflush** and **localchars** are both TRUE, whenever the **ao**, **intr**, or **quit** characters are recognized (and transformed into TELNET sequences – see **set** above for details), **telnet** refuses to display any data on the user's terminal until the remote system acknowledges (via a TELNET *Timing Mark* option) that it has processed those TELNET sequences. The initial value for this toggle is TRUE.

autologin

Enable or disable automatic login into the Kerberos realm. Using this option yields the same results as using the **-a** option. The initial value for this toggle is TRUE.

autosynch

If **autosynch** and **localchars** are both TRUE, when either the **intr** or **quit** character is typed (see **set** above for descriptions of the **intr** and **quit** characters), the resulting TELNET sequence sent is followed by the TELNET SYNCH sequence. This procedure *should* cause the remote system to begin discarding all previously typed input until both of the TELNET sequences have been read and acted upon. The initial value of this toggle is FALSE.

binary

Enable or disable the TELNET BINARY option on both input and output. This option should be enabled in order to send and receive 8-bit characters to and from the TELNET server.

crlf

If TRUE, end-of-line sequences are sent as an ASCII carriage-return and line-feed pair. If FALSE, end-of-line sequences are sent as an ASCII carriage-return and NUL character pair. The initial value for this toggle is FALSE.

crmod

Toggle carriage return mode. When this mode is enabled, any carriage return characters received from the remote host are mapped into a carriage return and a line feed. This mode does not affect those characters typed by the user; only those received. This mode is only required for some hosts that require the client to do local echoing, but output "naked" carriage returns. The initial value for this toggle is FALSE.

echo

Toggle local echo mode or remote echo mode. In local echo mode, user input is echoed to the terminal by the local **telnet** before being transmitted to the remote host. In remote echo, any echoing of user input is done by the remote host. Applications that handle echoing of user input themselves, such as C shell, Korn shell, and **vi** (see *csh*(1), *ksh*(1), and *vi*(1)), do not work correctly with local echo.

options

Toggle viewing of TELNET options processing. When options viewing is enabled, all TELNET option negotiations are displayed. Options sent by **telnet** are displayed as **''SENT''**, while options received from the TELNET server are displayed as **''RCVD''**. The initial value for this toggle is FALSE.

netdata
Toggles the display of all network data (in hexadecimal format). The initial value for this toggle is FALSE.

? Displays the legal **toggle** commands.

RETURN VALUE

In the event of an error, or if the TELNET connection is closed by the remote host, **telnet** returns a value of 1. Otherwise it returns zero (0).

DIAGNOSTICS

Diagnostic messages displayed by **telnet** are displayed below. Kerberos specific errors are listed in *sis*(5).

Error! Could not retrieve authentication type.
There are two authentication mechanisms used by TELNET. One authentication mechanism is based on Kerberos and the other is not. The type of authentication mechanism is obtained from a system file which is updated by **inetsvcs_sec** (see *inetsvcs_sec*(1M)). If the system file on either the local host or the remote host does not contain known authentication types, the above error is displayed.

telnet/tcp: Unknown service
telnet was unable to find the TELNET service entry in the *services*(4) database.

*hostname***: Unknown host**
telnet was unable to map the host name to an Internet address. Your next step should be to contact the system administrator to check whether there is an entry for the remote host in the **hosts** database (see *hosts*(4)).

?Invalid command
An invalid command was typed in **telnet** command mode.

*system call***>:** ...
An error occurred in the specified system call. See the appropriate manual entry for a description of the error.

AUTHOR

telnet was developed by the University of California, Berkeley.

SEE ALSO

csh(1), ksh(1), login(1), rlogin(1), stty(1), telnetd(1M), inetsvcs_sec(1M), hosts(4), services(4), termio(7), sis(5).

NAME

test - condition evaluation command

SYNOPSIS

test *expr*

[*expr* **]**

DESCRIPTION

The **test** command evaluates the expression *expr* and, if its value is True, returns a zero (true) exit status; otherwise, a nonzero (false) exit status is returned. **test** also returns a nonzero exit status if there are no arguments. The following primitives are used to construct *expr*:

-r *file*	True if *file* exists and is readable.
-w *file*	True if *file* exists and is writable.
-x *file*	True if *file* exists and is executable.
-f *file*	True if *file* exists and is a regular file.
-d *file*	True if *file* exists and is a directory.
-c *file*	True if *file* exists and is a character special file.
-b *file*	True if *file* exists and is a block special file.
-p *file*	True if *file* exists and is a named pipe (fifo).
-u *file*	True if *file* exists and its set-user-ID bit is set.
-g *file*	True if *file* exists and its set-group-ID bit is set.
-k *file*	True if *file* exists and its sticky bit is set.
-s *file*	True if *file* exists and has a size greater than zero.
-h *file*	True if *file* exists and is a symbolic link.
-t [*fildes*]	True if the open file whose file descriptor number is *fildes* (1 by default) is associated with a terminal device.
-z *s1*	True if the length of string *s1* is zero.
-n *s1*	True if the length of the string *s1* is non-zero.
s1 **=** *s2*	True if strings *s1* and *s2* are identical.
s1 **!=** *s2*	True if strings *s1* and *s2* are *not* identical.
s1	True if *s1* is *not* the null string.
n1 **-eq** *n2*	True if the integers *n1* and *n2* are algebraically equal. Any of the comparisons **-ne**, **-gt**, **-ge**, **-lt**, and **-le** can be used in place of **-eq**.

These primaries can be combined with the following operators:

!	Unary negation operator.
-a	Binary AND operator.
-o	Binary OR operator (**-a** has higher precedence than **-o**).
(*expr* **)**	Parentheses for grouping.

Note that all the operators and flags are separate arguments to **test**. Note also that parentheses are significant to the shell and therefore must be escaped.

test is interpreted directly by the shell, and therefore does not exist as a separate executable program.

EXTERNAL INFLUENCES

International Code Set Support

Single byte and multibyte character code sets are supported.

EXAMPLES

Exit if there are not two or three arguments:

```
if [ $# -l2 2 -o $# -gt 3 ]; then exit 1; fi
```

Create a new file containing the text string `default` if the file does not already exist:

```
[ ! -f thisfile ] && echo default > thisfile
```

Wait for myfile to become non-readable:

```
while test -r myfile
do
    sleep 30
done
echo '"myfile" is no longer readable'
```

WARNINGS

When the **[** form of this command is used, the matching **]** must be the final argument, and both must be separate arguments from the arguments they enclose (white space delimiters required.

Parentheses and other special shell metacharacters intended to be handled by test must be escaped or quoted when invoking **test** from a shell.

Avoid such problems when comparing strings by inserting a non-operator character at the beginning of both operands:

```
test "X$response" = "Xexpected string"
```

This approach does not work with numeric comparisons or the unary operators because it would affect the operand being checked.

AUTHOR

test was developed by the University of California, Berkeley and HP.

SEE ALSO

find(1), sh(1).

STANDARDS CONFORMANCE

test: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

[: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

NAME

tftp - trivial file transfer program

SYNOPSIS

tftp [*host*]

DESCRIPTION

tftp is the user interface to the Internet TFTP (Trivial File Transfer Protocol), that allows users to transfer files to and from a remote machine. The remote *host* can be specified on the command line, in which case **tftp** uses *host* as the default host for future transfers (see the **connect** command below).

Commands

Once **tftp** is running, it issues the prompt **tftp>** and recognizes the following commands:

connect *host-name* [*port*]
: Set the *host* (and optionally *port*) for transfers. Note that the TFTP protocol, unlike the FTP protocol, does not maintain connections between transfers; thus, the **connect** command does not actually create a connection, but merely remembers what host is to be used for transfers. You do not have to use the **connect** command; the remote host can be specified as part of the **get** or **put** commands.

mode *transfer-mode*
: Set the mode for transfers; *transfer-mode* can be one of **ascii** or **binary** (default is **ascii**).

put *file*
put *localfile remotefile*
put *file1 file2 ... fileN remotedirectory*
: Put a file or set of files to the specified remote file or directory. The destination can be in one of two forms: a filename on the remote host if the host has already been specified, or a string of the form *host***:***filename* to specify both a host and filename at the same time. If the latter form is used, the hostname specified becomes the default for future transfers. If the remote-directory form is used, the remote host is assumed to be a UNIX-like machine.

get *filename*
get *remotename localname*
get *file1 file2 ... fileN*
: Get a file or set of files from the specified *source*s. *source* can be in one of two forms: a filename on the remote host if the host has already been specified, or a string of the form *host***:***filename* to specify both a host and filename at the same time. If the latter form is used, the last hostname specified becomes the default for future transfers.

quit Exit **tftp**. Typing the end-of-file character also causes an exit.

verbose Toggle verbose mode.

trace Toggle packet tracing.

status Show current status.

rexmt *retransmission-timeout*
: Set the per-packet retransmission timeout, in seconds.

timeout *total-transmission-timeout*
: Set the total transmission timeout, in seconds.

ascii Shorthand for "mode ascii"

binary Shorthand for "mode binary"

? [*command-name ...*]
: Print help information.

WARNINGS

Since there is no user-login or validation within the TFTP protocol, the remote site probably has some sort of file-access restrictions in place. The exact methods are specific to each site and are therefore difficult to document here.

AUTHOR

tftp was developed by the University of California, Berkeley.

SEE ALSO

tftpd(1M).

NAME

time - time a command

SYNOPSIS

time *command*

XPG4 only

time [**-p**] *utility* [*argument* ...]

DESCRIPTION

command is executed. Upon completion, **time** prints the elapsed time during the command, the time spent in the system, and the time spent executing the command. Times are reported in seconds.

Execution time can depend on the performance of the memory in which the program is running.

The times are printed on standard error.

Options

time recognizes the following option:

-p (XPG4 only.) Writes the timing statistics to standard error.

SEE ALSO

sh(1), timex(1), times(2).

STANDARDS CONFORMANCE

time: SVID2, XPG2, XPG3, XPG4, POSIX.2

NAME

timex - time a command; report process data and system activity

SYNOPSIS

timex [**-o**] [**-p**[**fhkmrt**]] [**-s**] *command*

DESCRIPTION

timex reports in seconds the elapsed time, user time, and system time spent in execution of the given *command*. Optionally, process accounting data for *command* and all its children can be listed or summarized, and total system activity during the execution interval can be reported.

The output of **timex** is written on the standard error. Timex returns an exit status of 1 if it is used incorrectly, if it is unable to fork, or if it could not exec command . Otherwise, timex returns the exit status of command.

Options

-o Report the total number of blocks read or written and total characters transferred by *command* and all its children. This option works only if the process accounting software is installed.

-p[**fhkmrt**] List process accounting records for *command* and all its children. The suboptions **f**, **h**, **k**, **m**, **r**, and **t** modify the data items reported. They behave as defined in *acctcom*(1M). The number of blocks read or written and the number of characters transferred are always reported.

This option works only if the process accounting software is installed and **/usr/lib/acct/turnacct** has been invoked to create **/var/adm/pacct**.

-s Report total system activity (not just that due to *command*) that occurred during the execution interval of *command*. All the data items listed in *sar*(1M) are reported.

EXAMPLES

A simple example:

```
timex -ops sleep 60
```

A terminal session of arbitrary complexity can be measured by timing a sub-shell:

```
timex -opskmt sh
    session commands
EOT
```

WARNINGS

Process records associated with *command* are selected from the accounting file **/var/adm/pacct** by inference, since process genealogy is not available. Background processes having the same user-ID, terminal-ID, and execution time window are spuriously included.

SEE ALSO

sar(1M), acctcom(1M).

STANDARDS CONFORMANCE

timex: SVID2, SVID3

NAME

top - display and update information about the top processes on the system

SYNOPSIS

top [**-s** *time*] [**-d** *count*] [**-q**] [**-u**] [**-n** *number*]

DESCRIPTION

top displays the top processes on the system and periodically updates the information. Raw CPU percentage is used to rank the processes.

Options

top recognizes the following command-line options:

-s *time* Set the delay between screen updates to *time* seconds. The default delay between updates is 5 seconds.

-d *count* Show only *count* displays, then exit. A display is considered to be one update of the screen. This option is used to select the number of displays to be shown before the program exits.

-q This option runs the **top** program at the same priority as if it is executed via a **nice -20** command so that it will execute faster (see *nice*(1)). This can be very useful in discovering any system problem when the system is very sluggish. This option is accessibly only to users who have appropriate privileges.

-u User ID (uid) numbers are displayed instead of usernames. This improves execution speed by eliminating the additional time required to map uid numbers to user names.

-n *number* Show only *number* processes per screen. Note that this option is ignored if *number* is greater than the maximum number of processes that can be displayed per screen.

Screen-Control Commands

When displaying multiple-screen data, **top** recognizes the following keyboard screen-control commands:

j Display next screen if the current screen is not the last screen.

k Display previous screen if the current screen is not the first screen.

t Display the first (top) screen.

Program Termination

To exit the program and resume normal user activities, type **q** at any time.

Display Description

Three general classes of information are displayed by **top**:

System Data:

The first few lines at the top of the display show general information about the state of the system, including:

- System name and current time.
- Load averages in the last one, five, and fifteen minutes.
- Number of existing processes and the number of processes in each state (sleeping, waiting, running, starting, zombie, and stopped).
- Percentage of time spent in each of the processor states (user, nice, system, idle, interrupt and swapper) per processor on the system.
- Average value for each of the processor states (only on multi-processor systems).

Memory Data

Includes virtual and real memory in use (with the amount of memory considered "active" in parentheses) and the amount of free memory.

Process Data

Information about individual processes on the system. When process data cannot fit on a single screen, **top** divides the data into two or more screens. To view multiple-screen data, use the **j**, **k**, and **t** commands described previously. Note that the system- and memory-data displays are present in each screen of multiple-screen process data.

Process data is displayed in a format similar to that used by *ps*(1):

CPU	Processor number on which the process is executing (only on multi-processor systems).
TTY	Terminal interface used by the process.
PID	Process ID number.
USERNAME	Name of the owner of the process. When the **-u** option is specified, the user ID (uid) is displayed instead of **USERNAME**.
PRI	Current priority of the process.
NI	Nice value ranging from −20 to +20.
SIZE	Total size of the process in kilobytes. This includes text, data, and stack.
RES	Resident size of the process in kilobytes. The resident size information is, at best, an approximate value.
STATE	Current state of the process. The various states are **sleep**, **wait**, **run**, **idl**, **zomb**, or **stop**.
TIME	Number of system and CPU seconds the process has consumed.
%WCPU	Weighted CPU (central processing unit) percentage.
%CPU	Raw CPU percentage. This field is used to sort the top processes.
COMMAND	Name of the command the process is currently running.

EXAMPLES

top can be executed with or without command-line options. To display five screens of data at two-second intervals then automatically exit, use:

```
top -s2 -d5
```

AUTHOR

top was developed by HP and William LeFebvre of Rice University.

t

NAME

touch - update access, modification, and/or change times of file

SYNOPSIS

touch [**-amc**] [**-r** *ref_file* | **-t** *time*] *file_name* ...

Obsolescent:

touch *time_str file_name* ...

DESCRIPTION

touch updates the access, modification, and last-change times of each argument. The file name is created if it does not exist. If no time is specified (see *date*(1)) the current time is used.

The **-r** and **-t** options are mutually exclusive.

Options

The following options are available:

-a Change the access time of *file_name* to *time*, or to the current time if *time* is not specified. Do not change the modification time unless **-m** is also specified.

-m Change the modification time of *file_name* to *time*, or to the current time if *time* is not specified. Do not change the access time unless **-a** is also specified.

-c Silently prevent **touch** from creating the file if it did not previously exist. Do not write any diagnostic messages concerning this condition.

-r *ref_file*
Use the corresponding time of *ref_file* instead of the current time.

-t *time* Use the specified *time* instead of the current time. The option argument is a decimal number of the form:

[[*CC*]*YY*]*MMDD*hhmm[.*SS*]

where each two digits represents the following:

CC	The first two digits of the year.
YY	The second two digits of the year.
MM	The month of the year (01-12).
DD	The day of the month (01-31).
hh	The hour of the day (00-23).
mm	The minute of the hour (00-59).
SS	The second of the minute (00-61).

If neither *CC* nor *YY* is given, the current year is assumed. If *YY* is specified, but *CC* is not, *CC* is derived as follows: (taken into account the local time factor)

If *YY* is:	*CC* becomes:
69-99	19
00-68	20

If the resulting time value precedes the Epoch (00:00:00 January 1, 1970 Greenwich Mean Time), **touch** exits immediately with an error status.

The range for SS is 00 through 61 rather than 00 through 59 to accommodate leap seconds. If SS is 60 or 61, and the resulting time, as affected by the **TZ** environment variable, does not refer to a leap second, the resulting time is one second after a time where SS is 59. If SS is not given a value, it is assumed to be 0.

The syntax shown by the second SYNOPSIS line is recognized when neither the **-r** option, the **-t** option, nor the **--** option delimiter is specified, and the first operand consists of all decimal digits. This operand is interpreted as the *time* argument instead of as a file name. However, in this case, *time_str* is assumed to be of the form:

MMDDhhmm[YY]

t

This is for backward compatibility. The **-t** form given above is recommended for future portability. The **--** option delimiter can be used before the first *file_name* if there is a possibility that *file_name* consists of all digits, in order to ensure that the first syntax is used.

touch succeeds *only* when invoked by the *owner* of the file if any of the following are true:

- A time is specified,
- Only the access time of the file is being updated, or
- Only the modification time of the file is being updated.

In addition, **touch** succeeds when invoked by a user with write permission on the file if both of the following are true:

- No time is specified, *and*
- *Both* the access time and modification time of the file are being updated.

EXTERNAL INFLUENCES

Environment Variables

TZ If the time is specified via the **-t** option, **TZ** is used to interpret the time for the specified time zone.

LC_MESSAGES determines the language in which messages are displayed.

If **LC_MESSAGES** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**.

If any internationalization variable contains an invalid setting, **touch** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported.

RETURN VALUE

touch returns zero if all *file_name* arguments were successfully changed.

touch returns non-zero and prints out a diagnostic message if an invalid time or a time earlier than the Epoch was specified with the **-t** option, or if the **-r** and **-t** options were both specified, or if one or more of the *file_name* arguments could not be accessed.

EXAMPLES

The following command sets the modification and access times of the file named "bastille" to midnight, July 14, 1989, creating the file if it does not already exist.

```
touch -t 8907140000 bastille
```

The following command does the same thing using the backward-compatible syntax:

```
touch 0714000089 bastille
```

t

The following command sets the time of the two files named "0714000089" and "bastille" to the current time, creating them if they do not exist:

```
touch -- 0714000089 bastille
```

To create a zero-length file, use any of the following:

```
touch file
cat /dev/null >file
cp /dev/null file
```

DEPENDENCIES

NFS:

An attempt to touch a file owned by the super-user on a remote server might fail, even if the invoking user has write permission on the file.

SEE ALSO

date(1), utime(2).

STANDARDS CONFORMANCE

touch: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

NAME

tput - query terminfo database

SYNOPSIS

tput [**-T** *type*] *capname* ...

tput [**-T** *type*] *capname* [*parms* ...]

tput -S

DESCRIPTION

The **tput** command uses the **terminfo** database to make terminal-dependent capabilities and information available to the shell (see *terminfo*(4)). The **tput** command outputs a string if the attribute (*capname*) is of type string, or an integer if the attribute is of type integer. If the attribute is of type boolean, **tput** simply sets the exit code (**0** for TRUE, **1** for FALSE), and produces no output.

Command-line Arguments

The **tput** command recognizes the following command-line arguments:

-T*type* Indicates the type of terminal. Normally this flag is unnecessary because the default is taken from the environment variable **TERM**.

capname Indicates the attribute from the **terminfo** database. See *terminfo*(4). In addition, the following *capnames* are supported:

clear	Echo the clear-screen sequence for the current terminal.
init	Echo the initialize sequence for the current terminal.
reset	Echo the sequence that will reset the current terminal.

parms If the *capname* takes optional numeric parameters, the *parms* will be placed in the string output by **tput**.

-S The *capnames* are read from stdin and multiple *capnames* are allowed. Only one *capname* is allowed per line when reading from stdin.

EXTERNAL INFLUENCES

Environment Variables

LC_ALL determines the locale to use. This overrides settings of other environment variables.

LC_MESSAGES determines the language to use for messages.

TERM determines the terminal type if the **-T** option is not specified.

EXAMPLES

Echo clear-screen sequence for the current terminal.

```
tput clear
```

Print the number of columns for the current terminal.

```
tput cols
```

Print the number of columns for the 70092 terminal.

```
tput -T70092 cols
```

Set shell variable **bold** to stand-out-mode sequence for current terminal.

```
bold='tput smso'
```

This might be followed by a prompt:

```
echo "${bold}Please type in your name: \c"
```

Set exit code to indicate if current terminal is a hard copy terminal.

```
tput hc
```

Clear the screen, move the cursor to line 10, column 20 and turn on bold.

```
tput -S <<EOF
clear
cup 10 20
bold
EOF
```

RETURN VALUE

If *capname* is of type boolean, then the exit code is set to **0** for true and **1** for false.

If *capname* is not of type boolean and **tput** fails, an error message is printed, and exit code is set to one of the following depending on the failure:

0	The capability name is of type integer and does not exist.
2	Usage error.
3	Unknown terminal type.
4	Unknown capability name.
>4	An error occurred.

If the exit code is **0**, a **−1** is printed if a capability name of type integer is requested for a terminal that has no entry for that capability name in the **terminfo** database (such as **tput -Thp70092 vt**).

FILES

/usr/share/lib/terminfo/?/*	Terminfo data base
/usr/include/curses.h	Definition files
/usr/include/term.h	

SEE ALSO

stty(1), untic(1M), terminfo(4).

STANDARDS CONFORMANCE

tput: SVID2, SVID3, XPG4

NAME

tr - translate characters

SYNOPSIS

tr [**-Acs**] *string1 string2*

tr -s [**-Ac**] *string1*

tr -d [**-Ac**] *string1*

tr -ds [**-Ac**] *string1 string1*

DESCRIPTION

tr copies the standard input to the standard output with substitution or deletion of selected characters. Input characters from *string1* are replaced with the corresponding characters in *string2*. If necessary, *string1* and *string2* can be quoted to avoid pattern matching by the shell.

tr recognizes the following command line options:

-A Translates on a byte-by-byte basis. When this flag is specified **tr** does not support extended characters.

-c Complements the set of characters in *string1*, which is the set of all characters in the current character set, as defined by the current setting of **LC_CTYPE**, except for those actually specified in the *string1* argument. These characters are placed in the array in ascending collation sequence, as defined by the current setting of **LC_COLLATE**.

-d Deletes all occurrences of input characters or collating elements found in the array specified in *string1*.

If **-c** and **-d** are both specified, all characters except those specified by *string1* are deleted. The contents of *string2* are ignored, unless **-s** is also specified. Note, however, that the same string cannot be used for both the **-d** and the **-s** flags; when both flags are specified, both *string1* (used for deletion) and *string2* (used for squeezing) are required.

If **-d** is not specified, each input character or collating element found in the array specified by *string1* is replaced by the character or collating element in the same relative position specified by *string2*.

-s Replaces any character specified in *string1* that occurs as a string of two or more repeating characters as a single instance of the character in *string2*.

If the *string2* contains a character class, the argument's array contains all of the characters in that character class. For example:

```
tr -s'[:space:]'
```

In a case conversion, however, the *string2* array contains only those characters defined as the second characters in each of the **toupper** or **tolower** character pairs, as appropriate. For example:

```
tr -s'[:upper:]''[:lower:]'
```

The following abbreviation conventions can be used to introduce ranges of characters, repeated characters or single-character collating elements into the strings:

c1-*c2* or **[***c1*-*c2***]** Stands for the range of collating elements *c1* through *c2*, inclusive, as defined by the current setting of the **LC_COLLATE** locale category.

[:*class***:]**or **[[:***class***:]]** Stands for all the characters belonging to the defined character class, as defined by the current setting of **LC_CTYPE** locale category. The following character class names will be accepted when specified in *string1*: **alnum**, **alpha**, **blank**, **cntrl**. **digit**, **graph**, **lower**, **print**, **punct**, **space**, **upper**, or **xdigit**, Character classes are expanded in collation order.

When the **-d** and **-s** flags are specified together, any of the character class names are accepted in *string2*; otherwise, only character class names **lower** or **upper** are accepted in *string2* and then only if the corresponding character class (**upper** and **lower**, respectively) is specified in the same relative position in *string1*. Such a specification is interpreted as a request for case conversion.

When **[:lower:]** appears in *string1* and **[:upper:]** appears in *string2*, the arrays contain the characters from the **toupper** mapping in the **LC_CTYPE** category of the current locale. When **[:upper:]** appears in *string1* and **[:lower:]** appears in *string2*, the arrays contain the characters from the **tolower** mapping in the **LC_CTYPE** category of the current locale.

[=*c***=]** or **[[=**c**=]]** Stands for all the characters or collating elements belonging to the same equivalence class as *c*, as defined by the current setting of **LC_COLLATE** locale category. An equivalence class expression is allowed only in *string1*, or in *string2* when it is being used by the combined **-d** and **-s** options.

[*a*******n***]** Stands for *n* repetitions of *a*. If the first digit of *n* is **0**, *n* is considered octal; otherwise, *n* is treated as a decimal value. A zero or missing *n* is interpreted as large enough to extend *string2*-based sequence to the length of the *string1*-based sequence.

The escape character \ can be used as in the shell to remove special meaning from any character in a string. In addition, \ followed by 1, 2, or 3 octal digits represents the character whose ASCII code is given by those digits.

An ASCII NUL character in *string1* or *string2* can be represented only as an escaped character; i.e. as **\000**, but is treated like other characters and translated correctly if so specified. NUL characters in the input are not stripped out unless the option **-d "\000"** is given.

EXTERNAL INFLUENCES

Environment Variables

LANG provides a default value for the internationalization variables that are unset or null. If **LANG** is unset or null, the default value of "C" (see *lang*(5)) is used. If any of the internationalization variables contains an invalid setting, **tr** will behave as if all internationalization variables are set to "C". See *environ*(5).

LC_ALL If set to a non-empty string value, overrides the values of all the other internationalization variables.

LC_CTYPE determines the interpretation of text as single and/or multi-byte characters, the classification of characters as printable, and the characters matched by character class expressions in regular expressions.

LC_MESSAGES determines the locale that should be used to affect the format and contents of diagnostic messages written to standard error and informative messages written to standard output.

NLSPATH determines the location of message catalogues for the processing of **LC_MESSAGES.**

RETURN VALUE

tr exits with one of the following values:

0	All input was processed successfully.
>0	An error occurred.

t

EXAMPLES

For the ASCII character set and default collation sequence, create a list of all the words in *file1*, one per line in *file2*, where a word is taken to be a maximal string of alphabetics. Quote the strings to protect the special characters from interpretation by the shell (012 is the ASCII code for a new-line (line feed) character):

```
tr -cs "[A-Z][a-z]" "[\012*]" <file1 >file2
```

Same as above, but for all character sets and collation sequences:

```
tr -cs "[:alpha:]" "[\012*]" <file1 >file2
```

Translate all lower case characters in *file1* to upper case and write the result to standard output.

```
tr "[:lower:]" "[:upper:]" <file1
```

Use an equivalence class to identify accented variants of the base character **e** in *file1*, strip them of diacritical marks and write the result to *file2*:

```
tr "[=e=]" "[e*]" <file1 >file2
```

Translate each digit in *file1* to a **#** (number sign), and write the result to *file2*.

```
tr "0-9" "[#*]" <file1 >file2
```

The ***** (asterisk) tells **tr** to repeat the **#** (number sign) enough times to make the second string as long as the first one.

AUTHOR

tr was developed by OSF and HP.

SEE ALSO

ed(1), sh(1), ascii(5), environ(5), lang(5), regexp(5).

STANDARDS CONFORMANCE

tr: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

NAME

true, false - return exit status zero or one respectively

SYNOPSIS

true

false

DESCRIPTION

The command **true** does nothing, and returns exit code zero. The command **false** does nothing, and returns exit code one. They are typically used to construct command procedures.

RETURN VALUE

Exit values are:

0	always from *true*.
1	always from *false*.

EXAMPLES

This command loop repeats without end:

```
while true
    do
         command
    done
```

WARNINGS

true is typically used in shell scripts. Some shells provide a built-in version of **true** (and **false**) that is more efficient than the standalone versions.

SEE ALSO

csh(1), ksh(1), sh(1), sh-bourne(1), sh-posix(1).

STANDARDS CONFORMANCE

true: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

false: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

NAME

tset, reset - terminal-dependent initialization

SYNOPSIS

tset [*options*] [**-m** [*ident*] [*test baudrate*] **:***type*] ... [*type*]

reset

DESCRIPTION

tset sets up the terminal when logging in on an HP-UX system. It does terminal-dependent processing, such as setting erase and kill characters, setting or resetting delays, and sending any sequences needed to properly initialize the terminal. It first determines the *type* of terminal involved, then does the necessary initializations and mode settings. The type of terminal attached to each HP-UX port is specified in the **/etc/ttytype** data base. Type names for terminals can be found in the files under the **/usr/share/lib/terminfo** directory (see *terminfo*(4)). If a port is not wired permanently to a specific terminal (not hardwired), it is given an appropriate generic identifier, such as **dialup**.

reset performs a similar function, setting the terminal to a sensible default state.

In the case where no arguments are specified, **tset** simply reads the terminal type out of the environment variable TERM and re-initializes the terminal. The rest of this manual entry concerns itself with mode and environment initialization, typically done once at login, and options used at initialization time to determine the terminal type and set up terminal modes.

When used in a startup script (**.profile** for *sh*(1), or **.login** for *csh*(1) users) it is desirable to give information about the type of terminal that will normally be used on ports that are not hardwired. These ports are identified in **/etc/ttytype** as **dialup** or **plugboard**, etc. To specify what terminal type you usually use on these ports, the **-m** (map) option flag is followed by the appropriate port type identifier, an optional baud rate specification, and the terminal type. (The effect is to "map" from some conditions to a terminal type; that is, to tell **tset** that "If I am on this kind of port, I will probably be on this kind of terminal".) If more than one mapping is specified, the first applicable mapping prevails. A missing port type identifier matches all identifiers. A *baudrate* is specified as with **stty** (see *stty*(1)), and is compared with the speed of the diagnostic output (which should be the control terminal). The baud rate *test* can be any combination of **>**, **=**, **<**, **@**, and **!.** **@** is a synonym for **=**, and **!** inverts the sense of the test. To avoid problems with metacharacters, it is best to place the entire argument to **-m** within single quotes; users of *csh*(1) must also put a **** before any **!** used.

Thus,

```
    tset -m 'dialup>300:2622' -m 'dialup:2624' -m 'plugboard:?2623'
```

causes the terminal type to be set to an HP 2622 if the port in use is a dialup at a speed greater than 300 baud, or to an HP 2624 if the port is otherwise a dialup (i.e. at 300 baud or less). If the *type* finally determined by **tset** begins with a question mark, the user is asked for verification that the type indicated is really the one desired. A null response means to use that type; otherwise, another type can be entered. Thus, in the above case, if the user is on a plugboard port, he or she will be asked whether or not he or she is actually using an HP 2623.

t

If no mapping applies and a final *type* option, not preceded by a **-m**, is given on the command line, that type is used. Otherwise, the identifier found in the **/etc/ttytype** data base is taken to be the terminal type. The latter should always be the case for hardwired ports.

It is usually desirable to return the terminal type, as finally determined by *tset*, and information about the terminal's capabilities to a shell's environment. This can be done using the **-s** option. From *sh*(1), the command:

```
    eval `tset -s options...`
```

or using the C shell, (*csh*(1)):

```
    set noglob; eval `tset -s options...`
```

These commands cause **tset** to generate as output a sequence of shell commands which place the variable **TERM** in the environment; see *environ*(5).

Once the terminal type is known, **tset** engages in terminal mode setting. This normally involves sending an initialization sequence to the terminal, setting the single character erase (and optionally the full line erase or line-kill) characters, and setting special character delays. Tab and new-line expansion are turned off during transmission of the terminal initialization sequence.

On terminals that can backspace but not overstrike (such as a CRT), and when the erase character is the default erase character (**#** on standard systems), the erase character is changed to Back space (**^H**).

Options

tset recognizes the following options:

-e*c* Set the erase character to be the named character *c*; *c* defaults to **^H** (BACKSPACE). The character *c* can either be typed directly, or entered using circumflex notation used here (e.g., the circumflex notation for control-H is **^H**; in *sh*(1) the **^** character should be escaped (**\^**)).

-k*c* Set the kill character to *c*. The default *c* is **^X**. If *c* is not specified, the kill character remains unchanged unless the original value of the kill character is null, In which case the kill character is set to **@**.

- Report terminal type. Whatever type is decided on is reported. If no other flags are given, the only effect is to write the terminal type on the standard output. Has no effect if used with **-s**.

-s Generate appropriate commands (depending on current **SHELL** environment variable) to set **TERM**.

-I Suppress transmitting terminal initialization strings.

-Q Suppress printing the **Erase set to** and **Kill set to** messages.

-A Ask the user for the **TERM** type.

-S Output the strings that would be assigned to **TERM** in the environment rather than generating commands for a shell. In *sh*(1), the following is an alternate way of setting **TERM**:

```
set -- ‘tset -S ...‘
TERM=$1
```

-h Force a read of **/etc/ttytype**. When **-h** is not specified, the terminal type is determined by reading the environment unless some mapping is specified.

For compatibility with earlier versions of **tset**, the following flags are accepted, but their use is discouraged:

-r Report to the user in addition to other flags.

-E*c* Set the erase character to *c* only if the terminal can backspace. *c* defaults to **^H**.

In addition to capabilities described in **terminfo** (see *termio*(7) and *terminfo*(4)), the following boolean terminfo capabilities are understood by **tset** and **reset**, and can be included in the terminfo database for the purpose of terminal setup:

UC “Uppercase” mode sets character mapping for terminals that support only uppercase characters. Equivalent to **stty lcase**.

LC “Lowercase” mode permits input and output of lowercase characters. Equivalent to **stty -lcase**.

EP Set “even parity”. Equivalent to **stty parenb -parodd**

OP Set “odd parity”. Equivalent to **stty parenb parodd**.

NL Set “new line” mode. Equivalent to **stty onlret**.

HD Set “half-duplex” mode. Equivalent to **stty -echo**.

pt Set “print tabs” mode. Equivalent to **stty tabs**.

EXAMPLES

These examples all assume the *sh*(1). Note that a typical use of **tset** in a **.profile** also uses the **-e** and **-k** options, and often the **-m** or **-Q** options as well. These options have been omitted here to keep the examples small.

Assume, for the moment, that you are on an HP 2622. This is suitable for typing by hand but not for a **.profile** unless you are *always* on a 2622.

```
export TERM; TERM=‘tset - 2622‘
```

t

Assume you have an HP 2623 at home which you dial up on, but your office terminal is hardwired and known in **/etc/ttytype**.

```
export TERM; TERM=‘tset - -m dialup:2623‘
```

Suppose you are accessing the system through a switching network that can connect any system to any incoming modem line in an arbitrary combination, making it nearly impossible to key on what port you are coming in on. Your office terminal is an HP 2622, and your home terminal is an HP 2623 running at 1200 baud on dial-up switch ports. Sometimes you use someone else's terminal at work, so you want it to verify what terminal type you have at high speeds, but at 1200 baud you are always on a 2623. Note the placement of the question mark and the quotes to protect the **>** and **?** from interpretation by the shell.

```
export TERM; TERM=‘tset - -m ’switch>1200:?2622’ -m ’switch<=1200:2623’‘
```

All of the above entries fall back on the terminal type specified in **/etc/ttytype** if none of the conditions hold. The following entry is appropriate if you always dial up, always at the same baud rate, on many different kinds of terminals. Your most common terminal is an HP 2622. It always asks you what kind of terminal you are on, defaulting to 2622.

```
export TERM; TERM=‘tset - ?2622‘
```

If the file **/etc/ttytype** is not properly installed and you want to key entirely on the baud rate, the following can be used:

```
export TERM; TERM=‘tset - -m ’>1200:2624’ 2622‘
```

VARIABLES

SHELL	if **csh**, generate **csh** commands; otherwise generate *sh*(1) commands.
TERM	the (canonical) terminal name.

FILES

/etc/ttytype	port-name to terminal-type mapping data base;
/usr/share/lib/terminfo/?/*	terminal information data base.

AUTHOR

tset was developed by the University of California, Berkeley.

SEE ALSO

csh(1), sh(1), stty(1), ttytype(4), environ(5).

NAME

tsm - Terminal Session Manager

SYNOPSIS

`tsm`

DESCRIPTION

`tsm` allows a user to interact with more than one shell or application (session) from a single terminal. Each session is bound to a virtual device emulating the physical terminal. The emulation includes maintaining display state, softkeys, and terminal modes for each session. The virtual device can be manipulated like the actual terminal by using `stty` and `ioctl` (see *stty*(1) and *ioctl*(2)). Additionally `tsm` supports cut and paste between sessions, and provides an interface for a local lp device. Each session has its own process group ID.

Definitions

A session is **current** if it is being displayed and is the recipient of keyboard input.

The **standard search path** is:

```
./
$HOME
$TSMPATH
/usr/tsm/
```

Configuration files and such are searched for in the order indicated and defined by these paths.

Commands

There are two methods of interacting with `tsm`: a pull-down menu, and a command line interface. The pull-down menu (when configured) can be activated from a session by pressing the **tsm menu hot key** (default is `^T`) and should be self explanatory. The command line interface can be activated by pressing the **tsm hot key** (default is `^W`) in a session. Pressing a “hot key” twice passes the “hot key” character to the session instead of activating `tsm` command or menu mode.

Commands to `tsm` generally have single character invocation, in some cases the user is prompted for more input. The following commands can be issued from the `tsm` prompt level:

`0`-`9` Pressing a number at the command prompt selects the session of the same number to become the current session.

`+` Select the next higher numbered session.

`-` Select the next lower numbered session.

`l` Select the last session.

`?` Display a help screen describing `tsm` commands.

`c` Copy (cut): Three types:

- Text (Lines including new-lines). This is the default. Select with `T` when cut prompt is displayed.
- String (Lines strung together with white space in place of new-lines). Select with `T` when cut prompt is displayed.
- Block (A rectangle). Select with `T` when cut prompt is displayed.

The user is prompted for the "cut extents". The extents are defined by using arrow keys or the keys `u`, `d`, `l`, and `r` to move the cursor as desired. Pressing the space bar aborts the cut operation. The selected text is placed in the **cut buffer**. Trailing whitespace and character attribute information are ignored.

`p` Paste: the contents of the **cut buffer** is echoed to the current session as if it were typed from the keyboard.

`r` Run a program as a new session. The user is prompted for the program name.

`s` Start a new session containing a shell.

`o` Output the current display to a printer (screen dump). The print mechanism is specified in a file named `.tsmprint` searched for in the standard way. Character attribute information

t

is ignored.

k	Load the softkeys of the current session from a file. To load **tsm** defaults, specify "file" **+**. To load terminal defaults, specify "file" **-**.
g	Same as **k** above but softkeys are loaded "globally" into all sessions.
x	Access extended **tsm** commands as described in the **tsm** reference manual or on the **tsm** help screen.
q	Quit **tsm**: **SIGHUP** is sent to all processes started under **tsm**, and **tsm** exits.

EXTERNAL INFLUENCES

In general **tsm** environment variables must be set prior to **tsm** invocation. **TSMLP** is the *lp*(1) name of a printer that gets its output redirected to the printer port of the terminal.

TSMTPATH specifies an alternate search path for tsm files.

TSMTERM specifies an alternate terminal information file to be used by **tsm** instead of that specified by **TERM**. **TSMHOTKEY** specifies an alternate **tsm** hotkey for invocation ot the **tsm** command line.

WARNINGS

Some operations are not supported on certain terminals.

AUTHOR

tsm was developed by Structured Software Solutions, Inc.

FILES

/usr/tsm/.tsm	**tsm** main configuration file (default). Copy to **$HOME** for user customization.
/usr/tsm/.tsmkeys	**tsm** softkey configuration file (default). Copy to **$HOME** for user customization.
/usr/tsm/term/*	terminal description files

SEE ALSO

tsm.info(1), tsm.command(1), tsm.lpadmin(3M), shl(1).

NAME

tsm.command - send commands to the Terminal Session Manager (TSM)

SYNOPSIS

/usr/tsm/bin/tsm.command *command*

DESCRIPTION

tsm.command is used to send a command string programmaticly to the Terminal Session Manager (TSM), as if the string were typed on the TSM command line. **tsm.command** fails unless it is run from inside a TSM session. Actions caused by **tsm.command** affect only the instance of TSM that **tsm.command** is run under. *command* can have any value that is a valid key sequence for the TSM command line. The sequence should not include the "hotkey" character that normally initiates the command line mode of TSM. The sequence should end at the point where TSM exits command mode. If it ends prematurely TSM behaves as though escape was pressed, which exits command mode, usually canceling the command. **\r** should be used to indicate a return key. If no arguments are given on the command line, the program prompts for input from the user. If a **^C** terminates the sequence, the remainder of the sequence is accepted from the user.

AUTHOR

tsm.command was developed by Structured Software Solutions, Inc.

SEE ALSO

tsm(1).

NAME

tsm.info - get Terminal Session Manager state information

SYNOPSIS

/usr/tsm/bin/tsm.info *request*

DESCRIPTION

tsm.info is used to obtain information about TSM. When run from inside a TSM session it returns valid information; otherwise it fails with a nonzero error code. Information returned is written to standard output. *request* can have any of the following values:

is_a_window	Successful (returns zero) if executed from a TSM session, nonzero error code otherwise.
session_number	Writes the session number of the session the **tsm.info** command is executed in.
current_session_number	Writes the session_number of the TSM session the user currently has active.
active_session_numbers	Writes the session numbers (separated by whitespace) of all active sessions (sessions not idle).
idle_session_numbers	Writes the session numbers (separated by whitespace) of all idle sessions.
program_name	Writes the program name (as assigned in *.tsm* or with **tsm.command**) of the session the **tsm.info** command is executed in.
program_name_n	Writes the program name of session **n**.

AUTHOR

tsm.info was developed by Structured Software Solutions, Inc.

SEE ALSO

tsm(1)

t

NAME

tsort - topological sort

SYNOPSIS

tsort [*file*]

DESCRIPTION

tsort produces on the standard output a totally ordered list of items consistent with a partial ordering of items mentioned in the input text *file*. If no *file* is specified, the standard input is understood. **tsort** is generally used in conjunction with the **lorder** command to sort the objects to be installed in a library by **ar** (see *lorder*(1) and *ar*(1)).

The input consists of pairs of text items (nonempty strings) separated by blanks. Pairs of different items indicate ordering. Pairs of identical items indicate presence, but not ordering.

EXTERNAL INFLUENCES

Environment Variables

LC_CTYPE determines the locale for the interpretation of text as single- and/or multi-byte characters.

LC_MESSAGES determines the language in which messages are displayed.

If **LC_CTYPE** or **LC_MESSAGES** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**.

If any internationalization variable contains an invalid setting, **tsort** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported.

DIAGNOSTICS

Odd data

There is an odd number of fields in the input file.

WARNINGS

Libraries and object files cannot be **tsort**ed directly.

tsort uses a quadratic algorithm that is not considered worth fixing given its typical use of ordering a library archive file.

SEE ALSO

lorder(1).

STANDARDS CONFORMANCE

tsort: SVID2, SVID3, XPG2, XPG3, XPG4

NAME

tty, pty - get the name of the terminal

SYNOPSIS

tty [**-s**]

pty [**-s**]

DESCRIPTION

tty and **pty** print the path name of the user's terminal. The **-s** option inhibits printing of the terminal path name and any diagnostics, providing a means to test only the exit code.

RETURN VALUE

Exit status codes for **tty** are:

2	Invalid options were specified,
1	The standard input is not a terminal or pseudo-terminal,
0	The standard input is a terminal or pseudo-terminal.

Exit status codes for **pty** are:

2	Invalid options were specified,
1	The standard input is not a pseudo-terminal,
0	The standard input is a pseudo-terminal.

DIAGNOSTICS

not a tty
standard input is not a terminal or pseudo-terminal for **tty**.

not a pty
standard input is not a pseudo-terminal for **pty**.

AUTHOR

tty was developed by AT&T. **pty** was developed by HP.

STANDARDS CONFORMANCE

tty: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

NAME

ttytype - terminal identification program

SYNOPSIS

ttytype [**-apsv**] [**-t** *type*]

DESCRIPTION

ttytype automatically identifies the current terminal type by sending an identification request sequence to the terminal. This method works for local, modem, and remote terminal connections, as well as for the **hpterm** and **xterm** terminal emulators.

Once the terminal has been identified, **ttytype** prints the terminal's type to the standard output (see *terminfo*(4)). This string is usually used as the value for the **TERM** environment variable.

If **ttytype** is unable to determine the correct terminal type, it prompts the user for the correct terminal identification string.

Options

ttytype recognizes the following options:

-a Causes **ttytype** to return an ID of "unknown" instead of prompting for the terminal type if auto-identification fails. If this option is not present, **ttytype** interactively prompts the user for the terminal type if it is unable to determine the correct type automatically.

-p Causes **ttytype** to prompt for the terminal type before it sends the terminal identification request sequence. If the user responds with only a carriage return, **ttytype** proceeds with the automatic terminal identification process. Any other response is taken as the correct terminal type. Note that the **LINES** and **COLUMNS** variables are not set if the user manually enters a terminal type.

The **-p** option is normally used only for terminals that do not behave well when presented with **ttytype**'s terminal identification request sequence. It gives the user a chance to respond with the correct terminal type before any escape sequences are sent that could have an adverse effect on the terminal.

The **-a** option can be used in conjunction with the **-p** option. The **-a** option only inhibits interactive prompting after **ttytype** has failed to identify the terminal by other means.

-s Tells **ttytype** to print a series of shell commands to set the **TERM**, **LINES**, and **COLUMNS** environment variables to appropriate values. In addition, the variable **ERASE** is set to the two-character sequence representing the appropriate erase character for the terminal (DEL for ANSI terminals, backspace for all others). This two-character sequence can then be used as an argument to **stty** or **tset** (see *stty*(1) and *tset*(1)).

The **SHELL** environment variable is consulted to see which shell syntax to use for setting the environment variables. This output is normally used with a command of the form:

```
eval ‘ttytype -s‘
```

-t *type* **ttytype** normally attempts identification of Wyse, ANSI and HP terminals. The **-t** *type* argument can be used to restrict the inquiry to that required for terminals of the specified type. The accepted types are **ansi**, **hp**, and **wyse**. Multiple **-t** options can be specified.

-v Enable verbose messages to standard error.

EXAMPLES

ttytype is most commonly used as part of the login sequence. The following shell script fragment can be used during login shell initialization:

```
#
# If TERM is not set, see if our port is listed in /etc/ttytype.
# If /etc/ttytype doesn’t have information for our port, run
# ttytype(1) to try to determine the type of terminal we have.
#
```

t

```
# To have ttytype(1) prompt for the terminal type before trying
# to automatically identify the terminal, add the "-p" option
# to the "ttytype -s" command below.
#
if [ -z "$TERM" -o "$TERM" = network ]; then
    unset TERM
    eval ‘tset -s -Q‘
    if [ -z "$TERM" -o "$TERM" = unknown ]; then
      eval ‘ttytype -s‘
      tset -Q -e ${ERASE:-\^h} $TERM
    fi
fi
```

NOTES

Use of the **-s** option is highly recommended because many terminals support variable-size displays. This option provides the only means for automatically configuring the user environment in such a manner that applications can handle these terminals correctly. Note that **LINES** and **COLUMNS** are not set if the **-p** option is used and the user manually enters a terminal type.

The following steps are performed in the order indicated when identifying a terminal:

1. **ttytype** tries the Wyse 30/50/60 id request sequence.
2. **ttytype** tries the standard ANSI id request sequence. If a response is received, it is converted to a string according to an internal table.
3. **ttytype** tries the HP id request sequence.
4. If none of the above steps succeed, **ttytype** prompts interactively for the correct terminal type unless the **-a** option has been given.

ttytype may skip one or more of the first three steps, depending on the presence of **-t** options.

The HP ID-request sequence can switch some ANSI terminals into an unexpected operating mode. Recovery from such a condition sometimes requires cycling power on the terminal. To avoid this problem, **ttytype** always sends the HP identification sequence last.

WARNINGS

The terminal identification sequences sent by **ttytype** can cause unexpected behavior on terminals other than the Wyse 30/50/60, standard ANSI or HP terminals. If you have such terminals in your configuration, use the **-t** or **-p** options to prevent **ttytype** from sending sequences that cause unexpected behavior.

AUTHOR

ttytype was developed by HP.

SEE ALSO

csh(1), ksh(1), sh(1), stty(1), ttytype(4), environ(5).

NAME
ul - do underlining

SYNOPSIS
ul [**-t** *terminal*] [**-i**] [*name* ...]

DESCRIPTION
ul reads the named files (or standard input if none are given) and translates occurrences of underscores to the sequence which indicates underlining for the terminal in use, as specified by the environment variable **TERM**. The **-t** option overrides the terminal type specified in the environment. The *terminfo*(4) file corresponding to **TERM** is read to determine the appropriate sequences for underlining. If the terminal is incapable of underlining, but is capable of a standout mode, the standout mode is used instead. If the terminal can overstrike, or handles underlining automatically, **ul** degenerates to **cat**. If the terminal cannot underline, underlining is ignored.

The **-i** option causes **ul** to indicate underlining onto by a separate line containing appropriate dashes **-**; this is useful when you want to look at the underlining present in an **nroff** output stream on a CRT terminal.

WARNINGS
nroff usually outputs a series of backspaces and underlines intermixed with the text to indicate underlining. No attempt is made to optimize the backward motion.

EXTERNAL INFLUENCES
International Code Set Support
Single- and multi-byte character code sets are supported with the exception that multi-byte-character file names are not supported.

FILES
/usr/share/lib/terminfo/?/*	terminal capability files

AUTHOR
ul was developed by the University of California, Berkeley.

SEE ALSO
col(1), man(1), nroff(1).

NAME

umask - set or display the file mode creation mask

SYNOPSIS

Set Mask

umask *mask*

Display Mask

umask [**-S**]

DESCRIPTION

The **umask** command sets the value of the file mode creation mask or displays the current one. The mask affects the initial value of the file mode (permission) bits for subsequently created files.

Setting the File Mode Creation Mask

The **umask** *mask* command sets a new file mode creation mask for the current shell execution environment. *mask* can be a symbolic or numeric (obsolescent) value.

A symbolic mask provides a flexible way of modifying the mask permission bits individually or as a group. A numeric mask specifies all the permission bits at one time.

When a mask is specified, no output is written to standard output.

Symbolic Mask Value

A symbolic *mask* replaces or modifies the current file mode creation mask. It is specified as specified as a comma-separated list of operations in the following format. Whitespace is not permitted.

[*who*][*operator*][*permissions*][, ...]

The fields can have the following values:

who One or more of the following letters:

- **u** Modify permissions for user (owner).
- **g** Modify permissions for group.
- **o** Modify permissions for others.

Or:

- **a** Modify permissions for all (**a** = **ugo**).

operator One of the following symbols:

- **+** Add *permissions* to the existing mask for *who*.
- **-** Delete *permissions* from the existing mask for *who*.
- **=** Replace the existing mask for *who* with *permissions*.

permissions One or more of the following letters:

- **r** The read permission.
- **w** The write permission.
- **x** The execute/search permission.

u

If one or two of the fields are omitted, the following table applies:

Format Entered	Effect	Input	Equals
who	Delete current permissions for *who*	**g**	**g=**
operator	No action	**-**	(none)
permissions	Equal to: **a+***permissions*	**rw**	**a+rw**
*who***=**	Delete current permissions for *who*	**u=**	**u=**
*who***+**	No action	**u+**	(none)
*who***-**	No action	**u-**	(none)
whopermissions	Equal to: *who*= *permissions*	**ux**	**u=x**
operatorpermissions	Equal to: **a***operatorpermissions*	**-rw**	**a-rw**

Numeric Mask Value (Obsolescent)

A numeric *mask* replaces the current file mode creation mask. It is specified as an unsigned octal integer, constructed from the logical OR (sum) of the following mode bits (leading zeros can be omitted):

0400	(**a=rwx,u-r**)	Read by owner
0200	(**a=rwx,u-w**)	Write by owner
0100	(**a=rwx,u-x**)	Execute (search in directory) by owner
0040	(**a=rwx,g-r**)	Read by group
0020	(**a=rwx,g-w**)	Write by group
0010	(**a=rwx,g-x**)	Execute/search by group
0004	(**a=rwx,o-r**)	Read by others
0002	(**a=rwx,o-w**)	Write by others
0001	(**a=rwx,o-x**)	Execute/search by others

Displaying the Current Mask Value

To display the current file mode creation mask value, use one of the commands:

umask -S Print the current file mode creation mask in a symbolic format:

u=[**r**][**w**][**x**]**,g=**[**r**][**w**][**x**]**,o=**[**r**][**w**][**x**]

The characters **r** (read), **w** (write), and **x** (execute/search) represent the bits that are clear in the mask for **u** (user/owner), **g** (group), and **o** (other). All other bits are set.

umask Print the current file mode creation mask as an octal value.

The zero bits in the numeric value correspond to the displayed **r**, **w**, and **x** permission characters in the symbolic value. The one bits in the numeric value correspond to the missing permission characters in the symbolic value.

Depending on implementation, the display consists of one to four octal digits; the first digit is always zero (see DEPENDENCIES). The rightmost three digits (leading zeros implied as needed) represent the bits that are set or clear in the mask.

Both forms produce output that can be used as the *mask* argument to set the mask in a subsequent **umask** command.

General Operation

When a new file is created (see *creat*(2)), each bit that is set in the file mode creation mask causes the corresponding permission bit in the the file mode to be cleared (disabled). Conversely, bits that are clear in the mask allow the corresponding file mode bits to be enabled in newly created files.

For example, the mask **u=rwx,g=rx,o=rx** (**022**) disables group and other write permissions. As a result, files normally created with a file mode shown by the **ls -l** command as **-rwxrwxrwx** (**777**) become mode **-rwxr-xr-x** (**755**); while files created with file mode **-rw-rw-rw-** (**666**) become mode **-rw-r--r--** (**644**).

Note that the file creation mode mask does not affect the set-user-id, set-group-id, or "sticky" bits.

The file creation mode mask is also used by the **chmod** command (see *chmod*(1)).

Since **umask** affects the current shell execution environment, it is generally provided as a shell regular built-in (see DEPENDENCIES.

If **umask** is called in a subshell or separate utility execution environment, such as one of the following:

```
(umask 002)
nohup umask ...
find . -exec umask ...
```

it does not affect the file mode creation mask of the calling environment.

The default mask is **u=rwx,g=rwx,o=rwx** (**000**).

RETURN VALUE

umask exits with one of the following values:

0	The file mode creation mask was successfully changed or no *mask* operand was supplied.
>0	An error occurred.

EXAMPLES

In these examples, each line show an alternate way of accomplishing the same task.

Set the **umask** value to produce read and write permissions for the file's owner and read permissions for all others (**ls -l** displays **-rw-r--r--** on newly created files):

```
umask u=rwx,g=rx,o=rx      symbolic mode
umask a=rx,u+w             symbolic mode
umask 022                  numeric mode
```

Set the **umask** value to produce read, and write permissions for the file's owner, read-only for others users in the same group, and no access to others (**-rw-r-----**):

```
umask a-rwx,u+rw,g+r       symbolic mode
umask 137                  numeric mode
```

Set the **umask** value to deny read, write, and execute permissions to everyone (**----------**):

```
umask a=                   symbolic mode
umask 777                  numeric mode
```

Add the write permission to the current mask for everyone (there is no equivalent numeric mode):

```
umask a+w                  symbolic mode
```

WARNINGS

If you set a mask that prevents read or write access for the user (owner), many programs, such as editors, that create temporary files will fail because they cannot access the file data.

DEPENDENCIES

The **umask** command is implemented both as a separate executable file (**/usr/bin/umask**) and as built-in shell commands.

POSIX Shell and Separate File

All features are supported (see *sh-posix*(1). The numeric mask display uses a minimum of two digits.

Korn Shell

The **-S** option is not supported in the Korn shell built-in command (see *ksh*(1). The numeric mask display uses a minimum of two digits.

C Shell

The **-S** option and symbolic mask values are not supported in the C shell built-in command (see *csh*(1). The numeric mask display uses a minimum of one digit.

Bourne Shell

The **-S** option and symbolic mask values are not supported in the Bourne shell built-in command (see *sh-bourne*(1). The numeric mask display always consists of four digits.

SEE ALSO

chmod(1), csh(1), ksh(1), sh-posix(1), sh(1), chmod(2), creat(2), umask(2).

STANDARDS CONFORMANCE

umask: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

NAME

umodem - XMODEM-protocol file transfer program

SYNOPSIS

umodem [*options*] *files* ...

umodem -c

DESCRIPTION

umodem is a file transfer program that incorporates the well-known XMODEM protocol used on CP/M systems and on the HP 110 portable computer.

Options

umodem recognizes the following options and command-line arguments:

-1	(one) Employ TERM II FTP 1.
-3	Enable TERM FTP 3 (CP/M UG).
-7	Enable 7-bit transfer mask.
-a	Turn on ARPA Net flag.
-c	Enter command mode.
-d	Do not delete **umodem.log** before starting.
-l	(ell) Turn on entry logging.
-m	Allow overwriting of files.
-p	Print all messages.
-r[**t**|**b**]	Receive file. Specify **t** for text, or **b** for binary.
-s[**t**|**b**]	Send file. Specify **t** for text, or **b** for binary.
-y	Display file status only.
files	Name of file or files to be transferred.

WARNINGS

When a binary file is transferred, the target file may have some extra bytes added at the end. This is due to the limitation of the underlying **XMODEM** protocol.

EXAMPLES

Receive a text file:

umodem -rt7 *file*

Receive a binary file:

umodem -rb *file*

Send a text file:

umodem -st7 *file*

Send a binary file:

umodem -sb *file*

AUTHOR

umodem is in the public domain.

SEE ALSO

cu(1), kermit(1), uucp(1).

NAME

uname - display information about computer system; set node name (system name)

SYNOPSIS

uname [**-ailmnrsv**]

uname [**-S** *nodename*]

DESCRIPTION

In the first form above, the **uname** command displays selected information about the current computer system, derived from the **utsname** structure (see *uname*(2)).

In the second form, **uname** sets the node name (system name) that is used in the **utsname** structure.

Options

uname recognizes the options listed below. If you enter several options, the output is always in the order shown for the **-a** option.

none	Equivalent to **-s**.
-a	Display the options below in the following order, separated by blanks. **-s -n -r -v -m -i -l**
-i	Display the machine identification number (or the node name, if the machine identification number cannot be determined).
-l	Display the license level of the operating system. 128-, 256-, and unlimited-user licenses are shown as **unlimited-user license**.
-m	Display the machine hardware and model names. See *WARNINGS*.
-n	Display the node name (system name) by which the system is usually known in a UUCP network. See *WARNINGS*.
-r	Display the current release level of the operating system.
-s	Display the name of the operating system. On standard HP-UX systems, this option always displays **HP-UX**.
-v	Display the current version level of the operating system.
-S *nodename*	Change the node name (system name) to *nodename*. *nodename* is restricted to **UTSLEN-1** characters (see *uname*(2)). See *WARNINGS*. Only users with appropriate privileges can use the **-S** option.

EXAMPLES

When you execute the command **uname -a**, it produces output like the following:

```
HP-UX myhost A.09.01 C 9000/750 2015986034 32-user license
```

The displayed fields are interpreted as follows:

HP-UX	The operating system name (option **-s**).
myhost	The UUCP network system name by which the system is known (**-n**).
A.09.01	The operating system release identifier (**-r**).
C	The operating system version identifier (**-v**).
9000/750	The machine and model numbers (**-m**).
2015986034	The machine identification number (**-i**).
32-user license	The operating system license level (**-l**).

WARNINGS

It is recommended that the *model*(1) command or *getconf*(1) command be used to obtain the model name, since future model names may not be compatible with **uname**.

Many types of networking services are supported on HP-UX, each of which uses a separately assigned system name and naming convention. To ensure predictable system behavior, it is essential that system names (also called host names or node names) be assigned in such a manner that they do not create conflicts when the various networking facilities interact with each other.

The system does not rely on a single system name in a specific location, partly because different services use dissimilar name formats as explained below. The **hostname** and **uname** commands assign system names as follows:

Node Name	**Command**	*name* **Format**	**Used By**
Internet name	**hostname** *name*	*sys*[.*x*.*y*.*z*...]	ARPA and NFS Services
UUCP name	**uname -S** *name*	*sys*	**uucp** and related programs

where *sys* represents the assigned system name. It is *strongly* recommended that *sys* be identical for all commands and locations and that the optional .*x*.*y*.*z*... follow the specified notation for the particular ARPA/NFS environment.

Internet names are also frequently called host names or domain names (which are different from NFS domain names). Refer to *hostname*(5) for more information about Internet naming conventions.

Whenever the system name is changed in any file or by the use of any of the above commands, it should also be changed in all other locations as well. Other files or commands in addition to those above (such as **/etc/uucp/Permissions** if used to circumvent **uname**, for example) may contain or alter system names. To ensure correct operation, they should also use the same system name.

System names are normally assigned by the **/sbin/init.d/hostname** script at start-up, and should not be altered elsewhere.

SEE ALSO

hostname(1), setuname(1M), gethostname(2), sethostname(2), uname(2), hostname(5).

STANDARDS CONFORMANCE

uname: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

NAME

unget - undo a previous get of an SCCS file

SYNOPSIS

unget [**-r** *SID*] [**-s**] [**-n**] *file* ...

DESCRIPTION

The **unget** command undoes the effect of a **get -e** done prior to creating the intended new delta. If *file* is a directory name, **unget** treats each file in the directory as a file to be processed, except that non-SCCS files and unreadable files are silently ignored. If **-** is specified for *file*, the standard input is read with each line being taken as the name of an SCCS file to be processed. Refer to *sact*(1), which describes how to determine what deltas are currently binding for an s-file.

Options

unget recognizes the following options and command-line arguments. Options and arguments apply independently to each named *file*.

-r *SID* Uniquely identifies which delta is no longer wanted (this would have been specified by **get** as the "new delta"). This option is necessary only if two or more outstanding **get**s for editing on the same SCCS file were done by the same person (login name). **unget** prints a diagnostic message if the specified *SID* is ambiguous, or if it is required but not present on the command line (see *sact*(1)).

-s Silent option. Suppress printing the intended delta's *SID* on the standard output.

-n Retain the file deposited in the current directory by the previous **get**. Normally this file is removed by **unget**.

EXTERNAL INFLUENCES

Environment Variables

LC_MESSAGES determines the language in which messages are displayed.

If **LC_MESSAGES** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**.

If any internationalization variable contains an invalid setting, **unget** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single-byte and multi-byte character code sets are supported.

DIAGNOSTICS

Use *sccshelp*(1) for explanations.

WARNINGS

Only the user who did the corresponding **get -e** can execute **unget**. Any other user must either use **su** to change user ID to that user (see *su*(1)), or edit the *p-file* directly (which can be done either by the *s-file* owner or a user who has appropriate privileges).

FILES

p-file	See *delta*(1).
g-file	See *delta*(1).

SEE ALSO

delta(1), get(1), sccshelp(1), sact(1).

STANDARDS CONFORMANCE

unget: SVID2, SVID3, XPG2, XPG3, XPG4

NAME

unifdef - remove preprocessor lines

SYNOPSIS

unifdef [**-clt**] [[**-D** *sym*] [**-U** *sym*] [**-iD** *sym*] [**-iU** *sym*]] ... [*file*]

DESCRIPTION

unifdef simulates some of the actions of **cpp** in interpreting C language preprocessor command lines (see *cpp*(1)). For **unifdef**, a valid preprocessor command line contains as its first character a **#** and one of the following keywords: **ifdef**, **ifndef**, **if**, **else**, or **endif**. The **#** character and its associated keyword must appear on the same line, but they can be separated by spaces, tabs, and commented text. When appropriate, the portions of code surrounded by and including the targeted preprocessor directives are removed, and the resultant text is written to the standard output.

Unlike **cpp**, **unifdef** does not insert included files, interpret macros, or strip comment lines. This means, among other things, that **#define** and **#undef** macros occurring within the input text are not interpreted.

Since **unifdef** is language-independent, it can be used for processing source files for languages other than the C language. For example, **unifdef** can be used on FORTRAN language source files, provided the C language preprocessor commands are used.

Options

unifdef recognizes the following command-line options:

-c	Complement the normal behavior by printing only the rejected lines.
-iD*sym*	Ignore text delimited by **#ifdef** *sym*. In other words, text that would otherwise be affected by some action is not touched when found within the context of a preprocessor command using *sym*.
-iU*sym*	Ignore text delimited by **#ifndef** *sym*.
-l	Replace rejected lines with blank lines in the text written to the standard output.
-t	Treat the input source as plain text. C-language comment and quoting constructs are not recognized.
-D*sym*	Define symbol *sym*.
-U*sym*	Cause symbol *sym* to be undefined.

RETURN VALUE

The **unifdef** command returns the following exit values:

0	Output is an exact copy of the input.
1	Output is not an exact copy of the input.
2	The **unifdef** command fails. The failure might be due to a premature EOF or to an inappropriate **else**, **elif**, or **endif**.

u

EXAMPLES

Assume file **foo.f** contains the following:

```
    PROGRAM TEST1
    INTEGER I, J
#ifdef ANSI77
    DO I=1,10
#else
    DO 100 I=1,10
#endif
    J=J+1
#if defined (DEBUG) || defined (TEST)
    PRINT *,J
#endif
#ifdef ANSI77
    ENDDO
#else
```

```
        100   CONTINUE
#endif
    END
```

The command sequence:

```
unifdef -DANSI77 -UDEBUG -DTEST foo.f > /tmp/foo.f
```

produces the following result in file **/tmp/foo.f**:

```
PROGRAM TEST1
INTEGER I, J
DO I=1,10
J=J+1
PRINT *,J
ENDDO
END
```

WARNINGS

Any symbol name defined in the file must be specified in the **unifdef** command line; otherwise, **unifdef** will ignore the line.

AUTHOR

unifdef was developed in the public domain.

SEE ALSO

cpp(1).

u

NAME

uniq - report repeated lines in a file

SYNOPSIS

uniq [**-udc** [**-f** *fields*] [**-s** *chars*] [*input_file* [*output_file*]]

DESCRIPTION

uniq reads the input text file *input_file*, comparing adjacent lines, and copies the result to *output_file*. If *input_file* is not specified, the standard input and standard output are used. If *input_file* is specified, but *output_file* is not, results are printed to standard output. *input_file* and *output_file* must not be the same file.

Line-Comparison Options

uniq recognizes the following options when comparing adjacent lines:

-u	Print *only* those lines that are *not* repeated in the original file.
-d	Print *one* copy only of each repeated line in the input file.
-c	Generate an output report in default style except that each line is preceded by a count of the number of times it occurred. If this option is specified, the **-u** and **-d** options are ignored if either or both are also present.

If none of the options **u**, **d**, or **c** are present, **uniq** prints the results of the union of the **-u** and **-d** options, producing a copy of the original input file with the second and succeeding copies of any repeated lines removed. (Note that repeated lines must be adjacent in order to be found — see *sort*(1)).

Field-Skip Options

Two options are provided for skipping an initial portion of each line when making comparisons:

-f *fields*	Ignore the first *fields* fields, together with any blanks before each. *fields* is a positive decimal integer. A field is defined as a string of non-space, non-tab characters separated by tabs and/or spaces from its neighbors.
-s *chars*	Ignore the first *chars* characters. *chars* is a positive decimal integer. Each line in the input is assumed to be terminated with a new line character for purposes of comparison. Fields are skipped before characters.

EXTERNAL INFLUENCES

Environment Variables

LC_COLLATE must be equal to the value it had when the input files were sorted.

LC_CTYPE determines the interpretation of text within files as single- and/or multi-byte characters, and defines a space character when the **-f** or **-s** option is used.

LC_MESSAGES determines the language in which messages are displayed.

If **LC_COLLATE**, **LC_CTYPE** or **LC_MESSAGES** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**. If any internationalization variable contains an invalid setting, **uniq** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported.

RETURN VALUE

Exit values are:

0	Successful completion.
>0	Error condition occurred.

AUTHOR

uniq was developed by OSF and HP.

SEE ALSO

comm(1), sort(1).

STANDARDS CONFORMANCE

uniq: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

NAME

units - conversion program

SYNOPSIS

units [**-** *file*]

DESCRIPTION

units converts quantities expressed in various standard scales to their equivalents in other scales. It works interactively as follows:

System Prompt	**User Response**
`You have:`	`inch`
`You want:`	`cm`

The system responds with two factors; one used if multiplying (preceded by *****), the other if dividing (preceded by **/**):

```
* 2.540000e+00
/ 3.937008e-01
```

After providing the conversion factors, **units** prompts for the next set of values. To terminate **units**, press the interrupt character as defined for your login.

A quantity is specified as a multiplicative combination of units optionally preceded by a numeric multiplier. Powers are indicated by suffixed positive integers, and division by the usual sign:

System Prompt	**User Response**
`You have:`	`15 lbs force/in2`
`You want:`	`atm`

The system responds with:

```
* 1.020689e+00
/ 9.797299e-01
```

units only does multiplicative scale changes; thus it can convert Kelvin to Rankine, but not Celsius to Fahrenheit. Most familiar units, abbreviations, and metric prefixes are recognized, together with a generous leavening of exotica and a few constants of nature including:

pi	ratio of circumference to diameter
c	speed of light
e	charge on an electron
g	acceleration of gravity
force	same as **g**,
mole	Avogadro's number,
water	pressure head per unit height of water,
au	astronomical unit.

Units must be provided in lowercase only. **pound** is not recognized as a unit of mass; **lb** is. Compound names are run together, (e.g., **lightyear**). British units that differ from their U.S. counterparts are prefixed thus: **brgallon**. For a complete list of units, examine the file:

/usr/share/lib/unittab

An alternate unit database file can be specified for use with the **-** *file* option. **units** looks in this file rather than the default **/usr/share/lib/unittab** for the table of conversions. This must be in the same format as **/usr/share/lib/unittab**. This is useful in defining your own units and conversions.

WARNINGS

The monetary exchange rates are out of date.

FILES

/usr/share/lib/unittab

NAME

uptime, w - show how long system has been up, and/or who is logged in and what they are doing

SYNOPSIS

uptime [**-hlsuw**] [*user*]

w [**-hlsuw**] [*user*]

DESCRIPTION

uptime prints the current time, the length of time the system has been up, the number of users logged on to the system, and the average number of jobs in the run queue over the last 1, 5, and 15 minutes.

w is linked to **uptime** and prints the same output as **uptime -w**, displaying a summary of the current activity on the system.

Options

uptime and **w** recognize the following options:

- **-h** Suppress the first line and the heading line. This option should not be used with the **-u** option. This option assumes the use of the **-w** option to **uptime**.
- **-l** Use long output. This option assumes the use of the **-w** option to **uptime**.
- **-s** Use the short form of output for displaying terminal information. The terminal name is abbreviated; the login time and CPU times are suppressed.
- **-u** Print only the first line describing the overall state of the system. This is the default for the **uptime** command.
- **-w** Print a summary of the current activity on the system for each user. This is the default for the **w** command.

EXAMPLES

The command:

```
uptime
```

produces text resembling the following:

```
2:30pm  up 14days, 2:39, 33 users,  load average: 1.71, 1.88, 1.80
```

depending upon the current status of the system.

AUTHOR

uptime was developed by the University of California, Berkeley.

u

NAME
users - compact list of users who are on the system

SYNOPSIS
users

DESCRIPTION
users lists the login names of the users currently on the system in a compact, one-line format.

The login names are sorted in ascending collation order (see Environment Variables below).

EXTERNAL INFLUENCES
Environment Variables
LC_COLLATE determines the order in which the output is sorted.

If **LC_COLLATE** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**. If any internationalization variable contains an invalid setting, *users* behaves as if all internationalization variables are set to "C" (see *environ*(5)).

AUTHOR
users was developed by the University of California, Berkeley and HP.

FILES
/etc/utmp

SEE ALSO
who(1).

NAME

uucp, uulog, uuname - UNIX system to UNIX system copy

SYNOPSIS

uucp [*options*] *source_files destination_file*

uulog -f *system* [**-x**] [**-***number*]

uulog [**-s** *system*] ... [**-x**] [**-***number*]

uuname [**-l**]

DESCRIPTION

uucp

uucp copies files named by the *source_files* argument to the destination identified by the *destination_file* argument. When copying files to or from a remote system, *source_files* and *destination_file* can be a path name on the local system, or have the form:

system_name!*path_name*

where *system_name* is the name of a remote system in a list of system names known to **uucp**. When copying files to (but not from) a remote system, *system_name* can also be a chained list of remote system names such as:

system_name!*system_name*!...!*system_name*!*path_name*

in which case an attempt is made to send the file, via the specified route, to the destination. Care should be taken to ensure that intermediate nodes in the route are configured to forward information (see WARNINGS below for restrictions).

The shell metacharacters **?**, * and **[...]** appearing in *path_name* are expanded on the appropriate system.

path_name can be one of:

- A full path name.
- A path name preceded by ~*user* where *user* is a login name on the specified system and ~*user* is replaced by that user's login directory. (If an invalid login is specified, the default public directory (**/var/spool/uucppublic**) is used instead.
- A path name preceded by **~/***destination* where *destination* is appended to **/var/spool/uucppublic**.

  NOTE: This destination is treated as a file name unless more than one file is being transferred by this request or the destination is already a directory. To ensure that *destination* is a directory, append a **/** to the destination argument. For example, **~/dan/** as the destination argument causes directory **/var/spool/uucppublic/dan** to be created if it does not already exist, and places the requested file or files in that directory.
- Anything else is prefixed by the current directory.

u

If an erroneous path name is specified for the remote system, the copy fails. If *destination_file* is a directory, the file-name part of the *source_file* argument is used.

uucp preserves execute permissions across the transmission and sets read and write permissions to 0666 (see *chmod*(2) and Access Control Lists below).

Options

uucp recognizes the following options:

-c	Do not copy local file to the spool directory for transfer to the remote machine (default).
-C	Force the copy of local files to the spool directory for transfer.
-d	Make all necessary directories for the file copy (default).
-f	Do not make intermediate directories for the file copy.
-g*grade*	*grade* is a single letter or number. A lower ASCII sequence value for *grade* causes the job to be transmitted earlier in a given conversation between systems.

-j	Output the ASCII job identification string on standard output. This job identification can be used by **uustat** to obtain the status or terminate a job (see *uustat*(1)).
-m*file*	Send mail to the requester when the copy is completed.
-n*user*	Notify *user* on the remote system that a file was sent.
-r	Do not start the file transfer; just queue the job.
-s*file*	Report status of the transfer to *file*. Note that *file* must be a full path name.
-x*debug_level*	Produce debugging on standard output. *debug_level* is a number between 0 and 9; higher numbers give more information.

uulog

uulog queries a log file of **uucp** transactions in a file **/var/uucp/.Log/uucico/***system*.

The following options cause **uulog** to print logging information:

-s *system*	Print information about work involving *system*.
-f *system*	Do a **tail -f** (see *tail*(1)) of the file transfer log for *system*.

Other options used in conjunction with the **-s** and **-f** options above are:

-x	Search for the given system in the **/var/uucp/.Log/uuxqt/***system* file instead of in the *uucico* log file.
-*number*	Do a *tail*(1) command of *number* lines.

uuname

uuname lists the **uucp** names of known systems. **uuname -l** returns the local system's default name.

Access Control Lists (ACLs)

A file's optional ACL entries are not preserved across **uucp** transmission. Instead, new files have a summary of the access modes (as returned in **st_mode** by **stat()**; see *stat*(2)).

EXTERNAL INFLUENCES

Environment Variables

LC_TIME determines the format and contents of date and time strings displayed by **uucp** and **uulog** commands.

LANG determines the language in which messages are displayed by **uucp** and **uuname** commands.

If **LC_TIME** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**. If any internationalization variable contains an invalid setting, **uucp**, **uulog**, and **uuname** behave as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported with the exception that multi-byte-character file names are not supported.

WARNINGS

The domain of remotely accessible files can (and for obvious security reasons, usually should) be severely restricted. In most cases, you cannot fetch files by path name from a remote system. Ask a responsible person on the remote system to send them to you. For the same reasons, you probably cannot send files to arbitrary path names. As distributed, remotely accessible files are those whose names begin **/var/spool/uucppublic** (equivalent to **~/**).

All files received by **uucp** are owned by **uucp**.

The **-m** option only works when sending files or when receiving a single file. Receiving multiple files specified by special shell characters **? * [** ... **]** does not activate the **-m** option.

Protected files and files in protected directories owned by the requester can be sent by **uucp**. However, if the requester is root and the directory is not searchable by **other** or the file is not readable by **other**, the request fails.

FILES

/etc/uucp	configuration files
/var/uucp	log and error files
/var/spool/uucp	spool directories
/var/spool/locks	lock files
/var/spool/uucppublic	public directory for receiving and sending

SEE ALSO

mail(1), uux(1), chmod(2), stat(2), acl(5).

Tim O'Reilly and Grace Todino,
Managing UUCP and Usenet, O'Reilly & Associates, Inc. USA.

Grace Todino and Dale Dougherty,
Using UUCP and Usenet, O'Reilly & Associates, Inc. USA.

STANDARDS CONFORMANCE

uucp: SVID2, SVID3, XPG2, XPG3

uulog: SVID2, SVID3, XPG2, XPG3

uuname: SVID2, SVID3, XPG2, XPG3

NAME

uuencode, uudecode - encode/decode a binary file for transmission by mailer

SYNOPSIS

uuencode [*source*] *remotedest*

uudecode [*file*]

DESCRIPTION

uuencode and **uudecode** can be used to send a binary file to another machine by means of such services as *elm*(1), *mailx*(1), or *uucp*(1) (see *elm*(1), *mailx*(1), and *uucp*(1)).

uuencode takes the named source file (default standard input) and produces an encoded version on the standard output. The encoding uses only printing ASCII characters, includes the original mode of the input file, and preserves the value of the remotedest argument which is the intended name for the file when it is restored later on the remote system.

uudecode reads an encoded file, ignores any leading and trailing lines added by mailers, and recreates the original file with the specified mode and name.

The encoded file is an ordinary ASCII text file and can be edited with any text editor to change the mode or remote name.

EXAMPLES

To encode and send a compiled program **foo** to user **friend**:

```
uuencode foo foo | mailx -s 'new program' friend
```

After receiving the mail message, user **friend** can decode the program by first saving the message in a file **foo.mail** and executing the command:

```
uudecode foo.mail
```

WARNINGS

The file is expanded by 35% (three bytes become four plus control information) causing it to take longer to transmit.

The user on the remote system who is invoking **uudecode** (often **uucp**) must have write permission for the specified file.

If an encoded file has the same name as the destination name specified in it, **uudecode** starts overwriting the encoded file before decoding is completed.

SEE ALSO

elm(1), mail(1), mailx(1), shar(1), uucp(1), uux(1), uuencode(4).

STANDARDS CONFORMANCE

uuencode, uudecode: XPG4, POSIX.2

NAME

uupath, mkuupath - access and manage the pathalias database

SYNOPSIS

uupath [**-f** *pathsfile*] *mailaddress*

mkuupath [**-v**] *pathsfile*

DESCRIPTION

uupath provides electronic message routing by expanding a simple UUCP address into a full UUCP path (see *uucp*(1)). For example, *host***!***user* could be expanded into *hostA***!***hostB***!***host***!***user*.

uupath expands an address by parsing *mailaddress* for the dominant host (see below) and looking up the host in the appropriate **pathalias** database (see *pathalias*(1)). If the host is found in the database, the expanded address is written to the standard output. If the host is not found, **uupath** writes the original address to the standard output and returns an exit status of 1. **uupath** expects *mailaddress* to be in UUCP format (*host***!**...**!***hostZ***!***user*) or ARPANET format (*user***@***host*).

The **-f** option opens the **pathalias** database based on *pathsfile* rather than the default database based on **/usr/lib/mail/paths**. This database must be a database created by **mkuupath**, consisting of the two files *pathsfile***.dir** and *pathsfile***.pag**.

The dominant host is the left-most UUCP host in *mailaddress*. If no UUCP host is found (no **!** is in the address), **uupath** assumes that the address is in the simple ARPANET format *user***@***host*. If the address does not match either format, **uupath** writes the original address to the standard output and returns an exit status of 1.

mkuupath constructs a mail routing database by using the *pathsfile* data file obtained from **pathalias** (see *pathalias*(1)). as input. The recommended *pathsfile* location is **/usr/lib/mail/paths**, because this is the default database used by **uupath**. The database files *pathsfile***.dir** and *pathsfile***.pag** are created by **mkuupath**. If these files already exist, they must be removed prior to running **mkuupath**.

The **-v** option specifies verbose mode, which writes a line to the standard output for each entry written to the database.

DIAGNOSTICS

uupath returns an exit status of 1 and writes the original *mailaddress* to the standard output if the address is not found or is incorrectly formatted. **uupath** returns an exit status of 2 and prints a diagnostic message if the database files are not accessible, or if improper parameters are given. Otherwise, **uupath** returns an exit status of 0.

If the database files *pathsfile***.dir** and *pathsfile***.pag** already exist prior to running **mkuupath**, the message **mkuupath:** *pathsfile***.dir: File exists** is displayed. These files must be removed before running **mkuupath**.

AUTHOR

uupath was developed by University of California, Berkeley.

FILES

/usr/lib/mail/paths
/usr/lib/mail/paths.dir
/usr/lib/mail/paths.pag

SEE ALSO

pathalias(1), uucp(1).

NAME

uustat - uucp status inquiry and job control

SYNOPSIS

uustat -a
uustat -m
uustat -p
uustat -q
uustat -k *jobid*]
uustat -r *jobid*]
uustat [**-s***sys*] [**-u***user*]

DESCRIPTION

uustat displays the status of, or cancels, previously specified **uucp** commands, or provide general status on **uucp** connections to other systems (see *uucp*(1)). Only one of the following options can be specified with **uustat** per command execution:

-a Output all jobs in queue.

-m Report the status of accessibility of all machines.

-p Execute a **ps -flp** for all the process IDs that are in the lock files.

-q List the jobs queued for each machine. If a status file exists for the machine, its date, time and status information are reported. In addition, if a number appears in **( )** next to the number of **C** or **X** files it is the age in days of the oldest **C.** or **X.** file for that system. The Retry field is the number of hours until the next possible call. The Count field is the number of failure attempts. Note that for systems with a moderate number of outstanding jobs, this could take 30 seconds or more of real time to execute. As an example of the output produced by **uustat -q :**

```
eagle  3C 04/07-11:07 NO DEVICES AVAILABLE
mh3bs3 2C 07/07-10:42 SUCCESSFUL
```

The above output tells how many command files are waiting for each system. Each command file can have zero or more files to be sent (a command file with no files to be sent causes the **uucp** system to call the remote system and see if work is waiting). The date and time refer to the previous interaction with the system followed by the status of interaction.

-k *jobid* Kill the **uucp** request whose job identification is *jobid*. The killed **uucp** request must belong to the person issuing the **uustat** command unless the command is executed by the super-user.

-r *jobid* Rejuvenate *jobid*. The files associated with *jobid* are touched so that their modification time is set to the current time. This prevents the cleanup daemon from deleting the job until the jobs modification time reaches the limit imposed by the cleanup daemon.

The following options can be used singly or together but cannot be used with the options listed above:

-s *sys* Report the status of all **uucp** requests for remote system *sys*.

-u *user* Report the status of all **uucp** requests issued by *user*.

Output for both the **-s** and **-u** options has the following format:

```
eaglen0000   4/07-11:01:03   (POLL)
eagleN1bd7   4/07-11:07      S       eagle dan 522 /usr/dan/A
eagleC1bd8   4/07-11:07      S       eagle dan 59 D.3b2a12cd4924
             4/07-11:07      S       eagle dan rmail mike
```

With the **-s** and **-u** options, the first field is the *jobid* of the job. This is followed by the date and time. The next field is either an **S** or **R**, depending on whether the job is to send or request a file. The next field is the destination system name. This is followed by the user ID of the user who queued the job. The next field contains the size of the file, or in the case of a remote execution the name of the command (such as **rmail** which is the command used for remote mail). When the size appears in this field, the file name is also given. This can either be the name given by the user or an internal name (such as **D.3b2alce4924**) that is created for data files associated with remote execution (**rmail** in this example).

u

When no options are given, **uustat** outputs the status of all **uucp** requests issued by the current user. The format used is the same as with the **-s** or **-u** options.

EXTERNAL INFLUENCES

Environment Variables

LC_TIME determines the format and contents of date and time strings.

LANG determines the language in which messages are displayed.

If **LC_TIME** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**. If any internationalization variable contains an invalid setting, **uustat** behaves as if all internationalization variables are set to "C". See *environ*(5).

FILES

/var/spool/uucp/*	spool directories

SEE ALSO

uucp(1).

Tim O'Reilly and Grace Todino,
Managing UUCP and Usenet, O'Reilly & Associates, Inc. USA.

Grace Todino and Dale Dougherty,
Using UUCP and Usenet, O'Reilly & Associates, Inc. USA.

STANDARDS CONFORMANCE

uustat: SVID2, SVID3, XPG2, XPG3, XPG4

NAME

uuto, uupick - public UNIX system to UNIX system file copy

SYNOPSIS

uuto [*options*] *source-files destination*

uupick [**-s** *system*]

DESCRIPTION

uuto sends *source-files* to *destination*. **uuto** uses the **uucp** facility to send files (see *uucp*(1)), while allowing the local system to control the file access. A source-file name is a path name on your machine. Destination has the form:

system **!** *user*

where *system* is taken from a list of system names that **uucp** knows about (see **uuname** in *uucp*(1) manual entry). *user* is the login name of someone on the specified system.

uuto recognizes the following options:

-p	Copy the source file into the spool directory immediately, and send the copy.
-m	Send mail to the requester when the copy is complete.

The files (or sub-trees if directories are specified) are sent to *PUBDIR* on *system*, where *PUBDIR* is the UUCP public directory (**/var/spool/uucppublic**). Specifically the files are sent to

PUBDIR/**receive/** *user* **/** *mysystem* **/files**.

The recipient is notified by electronic mail when the files arrive.

uupick accepts or rejects the files transmitted to the recipient. Specifically, **uupick** searches *PUBDIR* for files destined for the user. For each entry (file or directory) found, the following message is printed on the standard output:

from *system* **:** [**file** *file-name*] [**dir** *dirname*] **?**

uupick then reads a line from the standard input to determine the disposition of the file:

<new-line>	Go on to next entry.
d	Delete the entry.
m [*dir*]	Move the entry to named directory *dir* (current directory is default). Note that, if the current working directory is desired for *dir*, do *not* specify any parameter with **m**. A construction such as **m.** is erroneous, and results in loss of data.
a [*dir*]	Same as **m** except move all the files sent from *system*.
p	Print the contents of the file.
q	Stop.
EOT	(control-D) Same as **q**.
!*command*	Escape to the shell to execute *command*.
*****	Print a command summary.

uupick invoked with the **-s***system* option searches only the *PUBDIR* for files sent from *system*.

WARNINGS

To send files that begin with a dot (such as **.profile**) the filename must contain a corresponding dot. For example: **.profile**, **.prof***, and **.profil?** are correct, whereas ***prof*** and **?profile** are incorrect.

FILES

PUBDIR **/var/spool/uucppublic** public directory

SEE ALSO

mail(1), uuclean(1M), uucp(1), uustat(1), uux(1).

Tim O'Reilly and Grace Todino,
Managing UUCP and Usenet, O'Reilly & Associates, Inc. USA.

u

Grace Todino and Dale Dougherty,
Using UUCP and Usenet, O'Reilly & Associates, Inc. USA.

STANDARDS CONFORMANCE

`uuto`: SVID2, SVID3, XPG2, XPG3, XPG4

`uupick`: SVID2, SVID3, XPG2, XPG3, XPG4

NAME

uux - UNIX system to UNIX system command execution

SYNOPSIS

uux [*options*] *command-string*

DESCRIPTION

uux gathers zero or more files from various systems, executes a command on a specified system, then sends standard output to a file on a specified system. Note that, for security reasons, many installations limit the list of commands executable on behalf of an incoming request from **uux**. Many sites will permit little more than the receipt of mail (see *mail*(1), *mailx*(1), and *elm*(1)) via **uux**.

The *command-string* is made up of one or more arguments that look like a shell command line, except that the command and file names may be prefixed by *system-name* **!**. A null *system-name* is interpreted as the local system.

File names can be one of the following:

- A full path name;
- A path name preceded by **~***xxx* where *xxx* is a login name on the specified system and is replaced by that user's login directory. Note that if an invalid login is specified, the default will be to the public directory (**/var/spool/uucppublic**);
- A path name preceded by **~/***destination* where *destination* is appended to **/var/spool/uucppublic**.
- A simple file name (which is prefixed by the current directory). See *uucp*(1) for details.

For example, the command

```
uux "!diff usg!/usr/dan/file1 pwba!/a4/dan/file2 > !~/dan/file.diff"
```

gets files **file1** and **file2** from machines **usg** and **pwba**, and executes a *diff*(1) command, placing the results in **file.diff** in the local directory **/var/spool/uucppublic**.

Any special shell characters such as **<**, **>**, **;**, or **|** should be quoted, either by quoting the entire *command-string*, or quoting the special characters as individual arguments.

uux attempts to get all files to the execution system. For files that are output files, the file name must be escaped using parentheses. For example, the command

```
uux a!cut -f1 b!/usr/file \(c!/usr/file\)
```

gets **/usr/file** from system **b** and sends it to system **a**, performs a **cut** command on the file, and sends the result of the cut command to system **c**.

uux notifies you if the requested command on the remote system was disallowed. The list of commands allowed is specified in the **Permissions** file in **/etc/uucp**. The response comes by remote mail from the remote machine.

uux recognizes the following *options*:

-	The standard input to **uux** is made the standard input to the *command-string*.
-a*name*	Use *name* as the user identification replacing the initiator user-ID (notification is returned to the user).
-b	Return whatever standard input was provided to the **uux** command if the exit status is non-zero.
-c	Do not copy the local file to the spool directory for transfer to the remote machine (default).
-C	Force the copy of local files to the spool directory for transfer.
-g*grade*	*grade* is a single letter/number; lower ASCII sequence characters cause the job to be transmitted earlier during a particular conversation.
-j	Output the *jobid* (the job identification ASCII string) on the standard output. This job identification can be used by **uustat** to obtain the status or terminate a job (see *uustat*(1)).

-n	Do not notify the user if the command fails.
-r	Do not start the file transfer, just queue the job.
-s*file*	Report status of the transfer in *file*.
-x*debug_level*	Produce debugging output on standard output. The *debug_level* is a number between 0 and 9. The higher the number, the more detailed the information returned.
-z	Send success notification to user.

WARNINGS

Only the first command of a shell pipeline can have a *system-name* **!**. All other commands are executed on the system of the first command.

The use of the shell metacharacter ***** will probably not do what you want it to do. The shell tokens **<<** and **>>** are not implemented.

The execution of commands on remote systems takes place in an execution directory known to the UUCP subsystem. All files required for the execution are put into this directory unless they already reside on that machine. Therefore, the simple file name (without path or machine reference) must be unique within the **uux** request. The following command does *not* work:

```
uux "a!diff b!/usr/dan/xyz c!/usr/dan/xyz > !xyz.diff"
```

but the command:

```
uux "a!diff a!/usr/dan/xyz c!/usr/dan/xyz > !xyz.diff"
```

works (if **diff** is a permitted command).

Protected files and files that are in protected directories that are owned by the requester can be sent in commands using **uux**. However, if the requester is **root**, and the directory is not searchable by **other**, the request fails.

FILES

/etc/uucp	configuration files
/var/uucp	log and error files
/var/spool/uucp	spool directories
/var/spool/locks	lock files
/var/spool/uucppublic	public directory

SEE ALSO

mail(1), uuclean(1M), uucp(1).

Tim O'Reilly and Grace Todino,
Managing UUCP and Usenet, O'Reilly & Associates, Inc. USA.

Grace Todino and Dale Dougherty,
Using UUCP and Usenet, O'Reilly & Associates, Inc. USA.

STANDARDS CONFORMANCE

uux: SVID2, SVID3, XPG2, XPG3, XPG4

NAME

vacation - return “I am not here” indication

SYNOPSIS

vacation -i

vacation [[**-a** *alias*] ...] *login*

DESCRIPTION

The **vacation** program returns a message to the sender of a message telling them that you are currently not reading your mail. The intended use is in a **.forward** file in **$HOME**. For example, your **.forward** file might contain:

```
\eric, "|/usr/bin/vacation -a allman eric"
```

which would send messages to you (assuming your login name was **eric**) and reply to any messages for **eric** or **allman**. The \ preceding **eric** is required to force direct delivery to **eric**'s mailbox and prevent an infinite loop through the **.forward** file. The double quotes are needed to tell *sendmail*(1M) to treat the enclosed as a unit, rather than separate recipients. It is also important to specify the full path for the vacation program, and there must be no white space between the | character and the start of the path name.

No message is sent unless *login* or an *alias* supplied using the **-a** option is a substring of either the **To:** or **Cc:** headers of the mail. No messages from **???-REQUEST**, **Postmaster**, **UUCP**, **MAILER**, or **MAILER-DAEMON** are replied to, nor is a notification sent if a **Precedence: bulk** or **Precedence: junk** line is included in the mail headers. Only one message per week is sent to each unique sender (at each unique host system). The people who have sent you messages are recorded in a database in the files **.vacation.pag** and **.vacation.dir** in your home directory.

The **vacation** program expects a file **.vacation.msg**, in your home directory, containing a message to be sent back to each sender. It should be an entire message (including headers). For example, it might say:

```
>From: eric@ucbmonet.Berkeley.EDU (Eric Allman)
Subject: I am on vacation
X-Delivered-By-The-Graces-Of: The vacation program
Precedence: bulk

I am on vacation until July 22. If you have something urgent,
please contact Joe Kalash <kalash@ucbingres.Berkeley.EDU>.
        --eric
```

Header lines in this file must be left justified and must not be preceded by any other lines, including blank lines (see *sendmail*(1M)). If there is no **.vacation.msg** file, **vacation** uses the following file (if it exists):

```
/usr/share/lib/vacation.def
```

Otherwise, it logs an error.

vacation reads the first line from the standard input (the incoming mail message in the example **.forward** file above) for a UNIX style **From** line to determine the sender. *sendmail*(1M) includes this **From** line automatically, and its absence indicates non-mail input.

Options

The **vacation** program supports the following options:

-i Initializes the vacation database files. This option should be used before modifying a **.forward** file.

-a *alias* Identifies another name that can legitimately appear in the **To:** line of the mail header instead of your login name. More than one **-a** option can be specified.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which error messages are printed.

DIAGNOSTICS

On error, **vacation** exits with a value from **<sysexits.h>** and causes **sendmail** to report an error back to the sender of the original message. Errors such as the absence of **.vacation.msg** or calling **vacation** with incorrect arguments, are logged using **syslogd** on the system where **vacation** actually runs (see *syslogd*(1M)). The **syslog** file (**/var/adm/syslog/mail.log** by default – see **/etc/syslog.conf** and *syslogd*(1M) for customizations) should be inspected when **vacation** generates mailer error messages.

Remember that if the machine is configured for shared mail, inbound mail is handled at the mail server rather than on mail client nodes. This means that **syslog** diagnostics appear in the mail server's **syslog**; not the client's **syslog**.

WARNINGS

Errors in the **.forward** file can lead to loss of mail and infinite mail loops.

Always send test mail to yourself after configuring **vacation** to be sure that it is working properly. This is akin to checking telephone forwarding before leaving for an extended period, and can prevent loss of messages.

AUTHOR

vacation was developed by Eric Allman and the University of California, Berkeley.

FILES

$HOME/.vacation.dir	Database file.
$HOME/.vacation.msg	Message to send.
$HOME/.vacation.pag	Database file.
/usr/share/lib/vacation.def	System-wide default header and message.
/etc/syslog.conf	Dictates where error messages are recorded.

SEE ALSO

sendmail(1M), syslogd(1M), ndbm(3X).

NAME

val - validate SCCS file

SYNOPSIS

val -

val [**-s**] [**-r** *SID*] [**-m** *name*] [**-y** *type*] [**-v**] *files*

DESCRIPTION

The **val** command reads one or more *file*s to determine whether each file read is an SCCS file meeting the characteristics specified by the optional argument list. Command-line options may appear in any order, and are described below.

Options

The **val** command recognizes the following options and command-line arguments. The effects of each option apply independently to each specified *file*.

-s	Silent option. Suppress diagnostic messages normally generated on the standard output when an error is encountered while processing any specified *file*.
-r *SID*	Check existence of revision *SID* in *file* where *SID* (*S*CCS *ID*entification string) is an SCCS delta number. *SID* is first checked to ensure that it is unambiguous and valid before checking *file*. For example, ***-r1** is ambiguous because it physically does not exist but implies 1.1, 1.2, etc., which may exist; ***-r1.0** and ***-r1.1.0** are invalid because they have a zero suffix which never appears in a valid delta number.
-m *name*	*name* is compared with the SCCS **%M%** keyword in *file*.
-y *type*	*type* is compared with the SCCS **%Y%** keyword in *file*.
-v	Verbose option. Prints additional detailed diagnostic messages on the standard output for any corruption detected while processing each named *file*. The messages are intended for use with the information contained in *sccsfile*(4) when fixing the file.
file	One or more SCCS files to be processed. If **-** is used as a *file* argument, **val** reads the standard input until an end-of-file condition is encountered. Each line read is independently processed as if it were a command-line argument list.

EXTERNAL INFLUENCES

Environment Variables

LC_CTYPE determines the interpretation of text within file as single- and/or multi-byte characters.

LC_MESSAGES determines the language in which messages are displayed.

If **LC_CTYPE** or **LC_MESSAGES** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**.

If any internationalization variable contains an invalid setting, **val** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single-byte and multi-byte character code sets are supported.

RETURN VALUE

The 8-bit code returned by **val** is a disjunction of the possible errors; i. e., can be interpreted as a bit string where (moving from left to right) set bits are interpreted as follows:

Bit	**Interpretation**
0	Missing *file* argument;
1	Unknown or duplicate option argument;
2	Corrupt SCCS file;
3	Cannot open *file* or *file* is not an SCCS file;
4	*SID* is invalid or ambiguous;
5	*SID* does not exist;
6	**%***Y***%** does not match **-y** *type* argument;
7	**%***M***%** does not match **-m** *name* argument;

Note that **val** can process two or more files on a given command line, and in turn can process multiple command lines (when reading the standard input). In these cases an aggregate code is returned; a logical **OR** of the codes generated for each command line and file processed.

DIAGNOSTICS

val generates diagnostic messages on the standard output for each command line and file processed, and also returns a single 8-bit code upon exit as described earlier under RETURN VALUE. Use the *sccshelp*(1) command for explanations.

SEE ALSO

admin(1), delta(1), get(1), sccshelp(1), prs(1), sccsfile(4).

STANDARDS CONFORMANCE

val: SVID2, SVID3, XPG2, XPG3, XPG4

NAME

vc - substitutes assigned values in place of identification keywords.

SYNOPSIS

vc [**-a**] [**-t**] [**-c** *char*] [**-s**] [*keyword=value* ... *keyword=value*]

DESCRIPTION

The **vc**, or *version control* command copies lines from the standard input to the standard output under control of command line *arguments* and *control statements* encountered in the standard input. In the process of performing the copy operation, user declared *keywords* can be replaced by their string *value* when they appear in plain text and/or control statements. The copying of lines from the standard input to the standard output is conditional, based on tests of keyword values specified in control statements or on **vc** command arguments.

Replacement of keywords by values is done whenever a keyword surrounded by control characters is encountered on a version control statement. The **-a** option forces replacement of keywords in *all* lines of text. An uninterpreted control character can be included in a value by preceding it with \. If a literal \ is desired, it too must be preceded by \.

The **vc** command is part of the SCCS (Source Code Control System) command suite.

Options

vc recognizes the following options and arguments:

- **-a** Replace keywords surrounded by control characters with their assigned value in *all* text lines and not just in **vc** statements.
- **-t** Ignore all characters from the beginning of a line up to and including the first *tab* character for the purpose of detecting a control statement. If one is found, all characters up to and including the *tab* are discarded.
- **-c***char* Specify a control character to be used in place of **:**.
- **-s** Silence warning messages (not errors) that are normally printed on the diagnostic output.

Control Statements

A control statement is a single line beginning with a control character, and the default control character is colon (**:**) (Unless the **-t** and **-c** options are used [See above]). Input lines beginning with a backslash (\) followed by the control character are not control lines, and are copied to the standard output with the backslash removed. Lines beginning with a backslash followed by a non-control character are copied in their entirety.

A keyword is composed of 9 or fewer alphanumeric characters of which the first character is alphabetic. A value is any ASCII string that can be created using **ed** (see *ed*(1)); a numeric value is an unsigned string of digits. Keyword values must not contain spaces or tabs.

Version control statements occur in the following forms:

:dcl keyword[, ..., keyword]
: Used to declare keywords. All keywords must be declared.

:asg keyword=value
: Used to assign values to keywords. An **asg** statement overrides the assignment for the corresponding keyword on the **vc** command line and all previous **asg**s for that keyword. Keywords declared, but not assigned values have null values.

:if condition
. . .
:end
: Used to skip lines of the standard input. If the condition is true, all lines between the *if* statement and the matching *end* statement are copied to the standard output. If the condition is false, all intervening lines are discarded, including control statements. Note that intervening *if* statements and matching *end* statements are recognized solely for the purpose of maintaining the proper *if-end* matching.

The syntax of a condition may include the following:

```
<cond>    ::= [ "not" ] <or>
<or>      ::= <and> | <and> "|" <or>
<and>     ::= <exp> | <exp> "&" <and>
<exp>     ::= "(" <or> ")" | <value> <op> <value>
<op>      ::= "=" | "!=" | "<" | ">"
<value>   ::= <arbitrary ASCII string> | <numeric string>
```

The following are available operators and their meanings:

=	equal
!=	not equal
&	and
|	or
>	greater than
<	less than
( )	used for logical groupings
not	allowed only immediately after the *if*, and when present, inverts the value of the entire condition

The **>** and **<** operate only on unsigned integer values (such as **: 012 > 12** is false). All other operators take strings as arguments (for example, **: 012 != 12** is true). The precedence of the operators (from highest to lowest) is as follows:

= != > <	all of equal precedence
&	
|	

Parentheses can be used to alter the order of precedence. Values must be separated from operators or parentheses by at least one space or tab.

::text — Used for keyword replacement on lines that are copied to the standard output. The two leading control characters are removed, and keywords surrounded by control characters in text are replaced by their value before the line is copied to the output file. This action is independent of the **-a** option.

:on
:off — Turn on or off keyword replacement on all lines.

:ctl**char** — Change the control character to char.

:msg**message** — Prints the given message on the diagnostic output.

:err**message** — Prints the given message followed by:

```
ERROR: err statement on line ... (915)
```

on the diagnostic output. **vc** halts execution and returns an exit code of 1.

EXTERNAL INFLUENCES

Environment Variables

LC_CTYPE determines the interpretation of keywords, values, the control character assigned through **ctl** and within text as single- and/or multi-byte characters.

V

LANG determines the language in which messages are displayed.

If **LC_CTYPE** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**. If any internationalization variable contains an invalid setting, **vc** behaves as if all internationalization variables are set to "C". See *environ*(5).

RETURN VALUE

vc returns 0 on normal completion; 1 if an error occurs.

DIAGNOSTICS

Use *sccshelp*(1) for explanations.

SEE ALSO

ed(1), sccshelp(1).

NAME

vi, view, vedit - screen-oriented (visual) text editor

SYNOPSIS

vi [**-**] [**-l**] [**-r**] [**-R**] [**-t** *tag*] [**-v**] [**-V**] [**-w***size*] [**-x**] [**-C**] [**+***command*] [*file* ...]

XPG4 Synopsis

vi [**-rR**] [**-c** *command*] [**-t** *tag*] [**-w** *size*] [*file* ...]

Obsolescent Options

vi [**-rR**] [+*command*] [**-t** *tag*] [**-w** *size*] [*file* ...]

view [**-**] [**-l**] [**-r**] [**-R**] [**-t** *tag*] [**-v**] [**-V**] [**-w***size*] [**-x**] [**-C**] [**+***command*] [*file* ...]

vedit [**-**] [**-r**] [**-R**] [**-l**] [**-t** *tag*] [**-v**] [**-V**] [**-w***size*] [**-x**] [**-C**] [**+***command*] [*file* ...]

Remarks

The program names **ex**, **edit**, **vi**, **view**, and **vedit** are separate personalities of the same program. This manual entry describes the behavior of the **vi**/**view**/**vedit** personality.

DESCRIPTION

The **vi** (visual) program is a display-oriented text editor that is based on the underlying **ex** line editor (see *ex*(1)). It is possible to switch back and forth between the two and to execute **ex** commands from within **vi**. The line-editor commands and the editor options are described in *ex*(1). Only the visual mode commands are described here.

The **view** program is identical to **vi** except that the **readonly** editor option is set (see *ex*(1)).

The **vedit** program is somewhat friendlier for beginners and casual users. The **report** editor option is set to **1**, and the **nomagic**, **novice**, and **showmode** editor options are set.

In **vi**, the terminal screen acts as a window into a memory copy of the file being edited. Changes made to the file copy are reflected in the screen display. The position of the cursor on the screen indicates the position within the file copy.

The environment variable **TERM** must specify a terminal type that is defined in the **terminfo** database (see *terminfo*(4)). Otherwise, a message is displayed and the line-editor is invoked.

As with **ex**, editor initialization scripts can be placed in the environment variable **EXINIT**, or in the file **.exrc** in the current or home directory.

Options and Arguments

vi recognizes the following command-line options and arguments:

-	Suppress all interactive-user feedback. This is useful when editor commands are taken from scripts.
-l	Set the **lisp** editor option (see *ex*(1)). Provides indents appropriate for **lisp** code. The **(**, **)**, **{**, **}**, **[[**, and **]]** commands in **vi** are modified to function with **lisp** source code.
-r	Recover the specified *file*s after an editor or system crash. If no *file* is specified, a list of all saved files is printed. You must be the owner of the saved file in order to recover it (superuser cannot recover files owned by other users).
-R	Set the **readonly** editor option to prevent overwriting a file inadvertently (see *ex*(1)).
-t *tag*	Execute the **tag** *tag* command to load and position a predefined file. See the **tag** command and the **tags** editor option in *ex*(1).
-v	Invoke visual mode (**vi**). Useful with **ex**, it has no effect on **vi**.
-V	Set verbose mode. Editor commands are displayed as they are executed when input from a **.exrc** file or a source file (see the **source** command in *ex*(1)).
-w*size*	Set the value of the **window** editor option to *size*. If *size* is omitted, it defaults to **3**.
-x	Set encryption mode. You are prompted for a key to allow for the creation or editing of an encrypted file. This command makes an educated guess to determine whether text read in is encrypted or not. The temporary buffer file is encrypted also, using a

transformed version of the key typed in for the **-x** option (see the **crypt** command in *ex*(1)).

-C Encryption option. Same as the **-x** option, except that all text read in is assumed to have been encrypted.

-c *command* (XPG4 only.)
+*command* (Obsolescent) Begin editing by executing the specified **ex** command-mode commands. As with the normal **ex** command-line entries, the **command** option-argument can consist of multiple **ex** commands separated by vertical-line commands (|). The use of commands that enter input mode in this manner produces undefined results.

file Specify the file or files to be edited. If more than one *file* is specified, they are processed in the order given. If the **-r** option is also specified, the files are read from the recovery area.

(XPG4 only.) If both the **-t** tag and **-c** command (or the obsolescent **+***command*) options are given, the **-t** tag will be processed first, that is, the file containing the tag is selected by **-t** and then the command is executed.

When invoked, **vi** is in *command mode*. *input mode* is initiated by several commands used to insert or change text.

In input mode, ESC (escape) is used to leave input mode; however, two consecutive ESC characters are required to leave input mode if the **doubleescape** editor option is set (see *ex*(1)).

In command mode, ESC is used to cancel a partial command; the terminal bell sounds if the editor is not in input mode and there is no partially entered command.

WARNING: ESC *completes* a "bottom line" command (see below).

The last (bottom) line of the screen is used to echo the input for search commands (**/** and **?**), **ex** commands (**:**), and system commands (**!**). It is also used to report errors or print other messages.

The receipt of **SIGINT** during text input or during the input of a command on the bottom line terminates the input (or cancels the command) and returns the editor to command mode. During command mode, **SIGINT** causes the bell to be sounded. In general the bell indicates an error (such as an unrecognized key).

Lines displayed on the screen containing only a **~** indicate that the last line above them is the last line of the file (the **~** lines are past the end of the file). Terminals with limited local intelligence might display lines on the screen marked with an **@**. These indicate space on the screen not corresponding to lines in the file. (These lines can be removed by entering a **^R**, forcing the editor to retype the screen without these holes.)

If the system crashes or **vi** aborts due to an internal error or unexpected signal, **vi** attempts to preserve the buffer if any unwritten changes were made. Use the **-r** command line option to retrieve the saved changes.

The **vi** text editor supports the **SIGWINCH** signal, and redraws the screen in response to window-size changes.

Command Summary

Most commands accept a preceding number as an argument, either to give a size or position (for display or movement commands), or as a repeat count (for commands that change text). For simplicity, this optional argument is referred to as *count* when its effect is described.

The following operators can be followed by a movement command to specify an extent of text to be affected: **c**, **d**, **y**, **<**, **>**, **!**, and **=**. The region specified begins at the current cursor position and ends just prior to the cursor position indicated by the move. If the command operates on lines only, all the lines that fall partly or wholly within this region are affected. Otherwise the exact marked region is affected.

In the following description, control characters are indicated in the form **^X**, which represents Ctrl-**X**. Whitespace is defined to be the characters space, tab, and alternative space. Alternative space is the first character of the **ALT_PUNCT** item described in *langinfo*(5) for the language specified by the **LANG** environment variable (see *environ*(5)).

Unless otherwise specified, the commands are interpreted in command mode and have no special effect in input mode.

^B	Scroll backward to display the previous window of text. A preceding *count* specifies the number of windows to go back. Two lines of overlap are kept if possible.
^D	Scroll forward a half-window of text. A preceding *count* gives the number of (logical) lines to scroll, and is remembered for future **^D** and **^U** commands.
^D	(input mode) Backs up over the indentation provided by **autoindent** or **^T** to the next multiple of **shiftwidth** spaces. Whitespace inserted by **^T** at other than the beginning of a line cannot be backed over using **^D**. A preceding ^ removes all indentation for the current and subsequent input lines of the current input mode until new indentation is established by inserting leading whitespace, either by direct input or by using **^T**.
^E	Scroll forward one line, leaving the cursor where it is if possible.
^F	Scroll forward to display the window of text following the current one. A preceding *count* specifies the number of windows to advance. Two lines of overlap are kept if possible. (XPG4 only.) The current line is displayed and the cursor is moved to the first nonblank character of the current line or the first character if the line is a blank line.
^G	Print the current file name and other information, including the number of lines and the current position (equivalent to the **ex** command **f**).
^H	Move one space to the left (stops at the left margin). A preceding *count* specifies the number of spaces to back up. (Same as **h**).
^H	(input mode) Move the cursor left to the previous input character without erasing it from the screen. The character is deleted from the saved text.
^J	Move the cursor down one line in the same column, if possible. A preceding *count* specifies the number of lines to move down. (Same as **^N** and **j**).
^L	Clear and redraw the screen. Use when the screen is scrambled for any reason.
^M	Move to the first nonwhitespace character in the next line. A preceding *count* specifies the number of lines to advance.
^N	Same as **^J** and **j**.
^P	Move the cursor up one line in the same column. A preceding *count* specifies the number of lines to move up (same as **k**).
^R	Redraw the current screen, eliminating the false lines marked with **@** (which do not correspond to actual lines in the file).
^T	Pop the tag stack. See the **pop** command in *ex*(1).
^T	(input mode) Insert **shiftwidth** whitespace. If at the beginning of the line, this inserted space can only be backed over using **^D**.
^U	Scroll up a half-window of text. A preceding *count* gives the number of (logical) lines to scroll, and is remembered for future **^D** and **^U** commands.
^V	In input mode, **^V** quotes the next character to permit the insertion of special characters (including ESC) into the file.
^W	In input mode, **^W** backs up one word; the deleted characters remain on the display.
^Y	Scroll backward one line, leaving the cursor where it is, if possible.
^[	Cancel a partially formed command; **^[** sounds the bell if there is no partially formed command. In input mode, **^[** terminates input mode. However, two consecutive ESC characters are required to terminate input mode if the **doubleescape** editor option is set (see *ex*(1)). When entering a command on the bottom line of the screen (**ex** command line or search pattern with \ or **?**), terminate input and execute command. On many terminals, **^[** can be entered by pressing the ESC or ESCAPE key.
^	Exit **vi** and enter *ex* command mode. If in input mode, terminate the input first.

^]	Take the word at or after the cursor as a tag and execute the **tag**MbobC editor command (see *ex*(1)).
^^	Return to the previous file (equivalent to **:ex #**).
space	Move one space to the right (stops at the end of the line). A preceding *count* specifies the number of spaces to go forward (same as **l**).
erase	Erase, where *erase* is the user-designated erase character (see *stty*(1)). Same as **^H**.
kill	Kill, where *kill* is the user-designated kill character (see *stty*(1)). In input mode, *kill* backs up to the beginning of the current input line without erasing the line from the screen display.
susp	Suspend the editor session and return to the calling shell, where *susp* is the user-designated process-control suspend character (see *stty*(1)). See *ex*(1) for more information on the **suspend** editor command.
!	An operator that passes specified lines from the buffer as standard input to the specified system command, and replaces those lines with the standard output from the command. The **!** is followed by a movement command specifying the lines to be passed (lines from the current position to the end of the movement) and then the command (terminated as usual by a return). A preceding *count* is passed on to the movement command after **!**.
	Doubling **!** and preceding it by *count* causes that many lines, starting with the current line, to be passed.
"	Use to precede a named buffer specification. There are named buffers **1** through **9** in which the editor places deleted text. The named buffers **a** through **z** are available to the user for saving deleted or yanked text; see also **y**, below.
$	Move to the end of the current line. A preceding *count* specifies the number of lines to advance (for example, **2$** causes the cursor to advance to the end of the next line).
%	Move to the parenthesis or brace that matches the parenthesis or brace at the current cursor position.
&	Same as the *ex* command **&** (that is, **&** repeats the previous **substitute** command).
’	When followed by a **’**, **vi** returns to the previous context, placing the cursor at the beginning of the line. (The previous context is set whenever a nonrelative move is made.) When followed by a letter **a**-**z**, returns to the line marked with that letter (see the **m** command), at the first nonwhitespace character in the line.
	When used with an operator such as **d** to specify an extent of text, the operation takes place over complete lines (see also **‘**).
‘	When followed by a **‘**, **vi** returns to the previous context, placing the cursor at the character position marked (the previous context is set whenever a nonrelative move is made). When followed by a letter **a z**, returns to the line marked with that letter (see the **m** command), at the character position marked.
	When used with an operator such as **d** to specify an extent of text, the operation takes place from the exact marked place to the current position within the line (see also **’**).
[[	Back up to the previous section boundary. A section is defined by the value of the **sections** option. Lines that start with a form feed (**^L**) or **{** also stop **[[**.
	If the option **lisp** is set, the cursor stops at each **(** at the beginning of a line.
]]	Move forward to a section boundary (see **[[**).
^	Move to the first nonwhitespace position on the current line.
(	Move backward to the beginning of a sentence. A sentence ends at a **.**, **!**, or **?** followed by either the end of a line or by two spaces. Any number of closing **)**, **]**, **"**, and **’** characters can appear between the **.**, **!**, or **?** and the spaces or end of line. If a *count* is specified, the cursor moves back the specified number of sentences.
	If the **lisp** option is set, the cursor moves to the beginning of a **lisp** *s*-expression. Sentences also begin at paragraph and section boundaries (see **{** and **[[**).

) Move forward to the beginning of a sentence. If a *count* is specified, the cursor advances the specified number of sentences (see **(**).

{ Move back to the beginning of the preceding paragraph. A paragraph is defined by the value of the **paragraphs** option. A completely empty line and a section boundary (see **[[** above) are also interpreted as the beginning of a paragraph. If a *count* is specified, the cursor moves backward the specified number of paragraphs.

} Move forward to the beginning of the next paragraph. If a *count* is specified, the cursor advances the specified number of paragraphs (see **{**).

| Requires a preceding *count*; the cursor moves to the specified column of the current line (if possible).

+ Move to the first nonwhitespace character in the next line. If a *count* is specified, the cursor advances the specified number of lines (same as **^M**).

, The comma (**,**) performs the reverse action of the last **f**, **F**, **t**, or **T** command issued, by searching in the opposite direction on the current line. If a *count* is specified, the cursor repeats the search the specified number of times.

- The hyphen character (**-**) moves the cursor to the first nonwhitespace character in the previous line. If a *count* is specified, the cursor moves back the specified number of times.

_ The underscore character (_) moves the cursor to the first nonwhitespace character in the current line. If a *count* is specified, the cursor advances the specified number of lines, with the current line being counted as the first line; no *count* or a *count* of 1 specifies the current line.

. Repeat the last command that changed the buffer. If a *count* is specified, the command is repeated the specified number of times.

/ Read a string from the last line on the screen, interpret it as a regular expression, and scan forward for the next occurrence of a matching string. The search begins when the user types a carriage return to terminate the pattern; the search can be terminated by sending **SIGINT** (or the user-designated interrupt character).

When used with an operator to specify an extent of text, the defined region begins with the current cursor position and ends at the beginning of the matched string. Entire lines can be specified by giving an offset from the matched line (by using a closing **/** followed by a **+***n* or **-***n*).

0 Move to the first character on the current line (the **0** is not interpreted as a command when preceded by a nonzero digit).

: The colon character (**:**) begins an **ex** command. The **:** and the entered command are echoed on the bottom line; the **ex** command is executed when the user types a carriage return.

; Repeat the last single character find using **f**, **F**, **t**, or **T**. If a *count* is specified, the search is repeated the specified number of times.

< An operator that shifts lines to the left by one **shiftwidth**. The **<** can be followed by a move to specify lines. A preceding *count* is passed through to the move command.

When repeated (**<<**), shifts the current line (or *count* lines starting at the current one).

> An operator that shifts lines right one **shiftwidth** (see **<**).

= If the **lisp** option is set, **=** reindents the specified lines, as if they were typed in with **lisp** and **autoindent** set. **=** can be preceded by a *count* to indicate how many lines to process, or followed by a move command for the same purpose.

? Scan backwards, the reverse of **/** (see **/**).

@*buffer* Execute the commands stored in the named *buffer*. Be careful not to include a <return> character at the end of the buffer contents unless the <return> is part of the command stream. Commands to be executed in *ex* mode should be preceded by a colon (**:**).

~ The tilde (**~**) switches the case of the character under the cursor (if it is a letter), then moves one character to the right, stopping at the end of the line). A preceding *count* specifies how many characters in the current line are switched.

A Append at the end of line (same as **$a**).

B Back up one word, where a word is any nonblank sequence, placing the cursor at the beginning of the word. If a *count* is specified, the cursor moves back the specified number of words.

C Change the rest of the text on the current line (same as **c$**).

D Delete the rest of the text on the current line (same as **d$**).

E Move forward to the end of a word, where a word is any nonblank sequence. If a *count* is specified, the cursor advances the specified number of words.

F Must be followed by a single character; scans backwards in the current line, searching for that character and moving the cursor to it, if found. If a *count* is specified, the search is repeated the specified number of times.

G Go to the line number given as preceding argument, or the end of the file if no preceding *count* is given.

H Move the cursor to the top line on the screen. If a *count* is given, the cursor moves to *count* number of lines from the top of the screen. The cursor is placed on the first nonwhitespace character on the line. If used as the target of an operator, entire lines are affected.

I Insert at the beginning of a line (same as **^** followed by **i**).

J Join the current line with the next one, supplying appropriate whitespace: one space between words, two spaces after a period, and no spaces at all if the first character of the next line is a closing parenthesis (**)**). A preceding *count* causes the specified number of lines to be joined, instead of just two.

L Move the cursor to the first nonwhitespace character of the last line on the screen. If a *count* is given, the cursor moves to *count* number of lines from the bottom of the screen. When used with an operator, entire lines are affected.

M Move the cursor to the middle line on the screen, at the first nonwhitespace position on the line.

N Scan for the next match of the last pattern given to **/** or **?**, but in the opposite direction; this is the reverse of **n**.

O Open a new line above the current line and enter input mode.

P Put back (replace) the last deleted or yanked text before/above the cursor. Entire lines of text are returned above the cursor if entire lines were deleted or yanked. Otherwise, the text is inserted just before the cursor.

(XPG4 only.) In this case, the cursor is moved to last column position of the inserted characters.

If **P** is preceded by a named buffer specification (*x*), the contents of that buffer are retrieved instead.

Q Exit **vi** and enter **ex** command mode.

R Replace characters on the screen with characters entered, until the input is terminated with ESC.

S Change entire lines (same as **cc**). A preceding *count* changes the specified number of lines.

T Must be followed by a single character; scan backwards in the current line for that character, and, if found, place the cursor just after that character. A *count* is equivalent to repeating the search the specified number of times.

U Restore the current line to its state before the cursor was last moved to it.

(XPG4 only.) The cursor position is set to the column position 1 or to the position indicated by the previous line if the **autoindent** is set.

W	Move forward to the beginning of a word in the current line, where a word is a sequence of nonblank characters. If the current position is at the beginning of a word, the current position is within a bigword or the character at that position cannot be a part of a bigword, the current position shall move to the first character of the next bigword. If no subsequent bigword exists on the current line, the current position shall move to the first character of the first bigword on the first following line that contains the bigword. For this command, an empty or blank line is considered to contain exactly one bigword. The current line is set to the line containing the bigword selected and the current position is set to the first character of the bigword selected. A preceding *count* specifies the number of words to advance.
X	Delete the character before the cursor. A preceding *count* repeats the effect, but only characters on the current line are deleted.
Y	Place (yank) a copy of the current line into the unnamed buffer (same as **yy**). If a *count* is specified, *count* lines are copied to the buffer. If the **Y** is preceded by a buffer name, the lines are copied to the named buffer.
ZZ	Exit the editor, writing out the buffer if it was changed since the last write (same as the **ex** command **x**). Note that if the last write was to a different file and no changes have occurred since, the editor exits without writing out the buffer.
a	Enter input mode, appending the entered text after the current cursor position. A preceding *count* causes the inserted text to be replicated the specified number of times, but only if the inserted text is all on one line.
b	Back up to the previous beginning of a word in the current line. A word is a sequence of alphanumerics or a sequence of special characters. A preceding *count* repeats the effect.
c	Must be followed by a movement command. Delete the specified region of text, and enter input mode to replace deleted text with new text. If more than part of a single line is affected, the deleted text is saved in the numeric buffers. If only part of the current line is affected, the last character deleted is marked with a **$**. A preceding *count* passes that value through to the move command. If the command is **cc**, the entire current line is changed.
d	Must be followed by a movement command. Delete the specified region of text. If more than part of a line is affected, the text is saved in the numeric buffers. A preceding *count* passes that value through to the move command. If the command is **dd**, the entire current line is deleted.
e	Move forward to the end of the next word, defined as for **b**. A preceding *count* repeats the effect.
f	Must be followed by a single character; scan the rest of the current line for that character, and moves the cursor to it if found. A preceding *count* repeats the action that many times.
h	Move the cursor one character to the left (same as **^H**). A preceding *count* repeats the effect.
i	Enter input mode, inserting the entered text before the cursor (see **a**).
j	Move the cursor one line down in the same column (same as **^J** and **^N**).
k	Move the cursor one line up (same as **^P**).
l	Move the cursor one character to the right (same as **<space>**).
m*x*	Mark the current position of the cursor. **x** is a lowercase letter, **a**–**z**, that is used with the **‘** and **’** commands to refer to the marked line or line position.
n	Repeat the last **/** or **?** scanning commands.
o	Open a line below the current line and enter input mode; otherwise like **O**.
p	Put text after/below the cursor; otherwise like **P**.
r	Must be followed by a single character; the character under the cursor is replaced by the specified one. (The new character can be a new-line.) If **r** is preceded by a *count*, *count* characters are replaced by the specified character.

s Delete the single character under the cursor and enter input mode; the entered text replaces the deleted character. A preceding *count* specifies how many characters on the current line are changed. The last character being changed is marked with a **$**, as for **c**.

t Must be followed by a single character; scan the remainder of the line for that character. The cursor moves to the column prior to the character if the character is found. A preceding *count* is equivalent to repeating the search *count* times.

u Reverse the last change made to the current buffer. If repeated, **u** alternates between these two states; thus is its own inverse. When used after an insertion of text on more than one line, the lines are saved in the numerically named buffers.

w Move forward to the beginning of the next word (where word is defined as in **b**). A preceding *count* specifies how many words the cursor advances.

x Delete the single character under the cursor. When **x** is preceded by a *count*, **x** deletes the specified number of characters forward from the cursor position, but only on the current line.

y Must be followed by a movement command; the specified text is copied (yanked) into the unnamed temporary buffer. If preceded by a named buffer specification, **"***x*, the text is placed in that buffer also. If the command is **yy**, the entire current line is yanked.

z Redraw the screen with the current line placed as specified by the following options: **z**<return> specifies the top of the screen, **z.** the center of the screen, and **z-** the bottom of the screen. The commands **z^** and **z+** are similar to **^B** and **^F**, respectively. However, **z^** and **z+** do not attempt to maintain two lines of overlap. A *count* after the **z** and before the following character to specifies the number of lines displayed in the redrawn screen. A *count* before the **z** gives the number of the line to use as the reference line instead of the default current line.

Keyboard Editing Keys

At initialization, the editor automatically maps some terminal keyboard editing keys to equivalent visual mode commands. These mappings are only established for keys that are listed in the following table and defined in the *terminfo*(4) database as valid for the current terminal (as specified by the **TERM** environment variable).

Both command and input mode mappings are created (see the **map** command in *ex*(1)). With the exception of the **insert char** keys, which simply toggle input mode on and off, the input mode mappings exit input mode, perform the same action as the command mode mapping, and then reenter input mode.

On certain terminals, the character sequence sent by a keyboard editing key, which is then mapped to a visual mode command, can be the same character sequence a user might enter to perform another command or set of commands. This is most likely to happen with the input mode mappings; therefore, on these terminals, the input mode mappings are disabled by default. Users can override the disabling and enabling of both the command and input mode keyboard editing key mappings by setting the **keyboardedit** and **keyboardedit!** editor options as appropriate (see *ex*(1)). The **timeout**, **timeoutlen**, and **doubleescape** editor options are alternative methods of addressing this problem.

terminfo entry	command mode map	input mode map	map name	description
key_ic	**i**	**^[**	**inschar**	insert char
key_eic	**i**	**^[**	**inschar**	end insert char
key_up	**k**	**^[ka**	**up**	arrow up
key_down	**j**	**^[ja**	**down**	arrow down
key_left	**h**	**^[ha**	**left**	arrow left
key_right	**l**	**^[la**	**right**	arrow right
key_home	**H**	**^[Ha**	**home**	arrow home
key_il	**o^[**	**^[o^[a**	**insline**	insert line
key_dl	**dd**	**^[dda**	**delline**	delete line
key_clear	**^L**	**^[^La**	**clear**	clear screen
key_eol	**d$**	**^[d$a**	**clreol**	clear line
key_sf	**^E**	**^[^Ea**	**scrollf**	scroll down
key_dc	**x**	**^[xa**	**delchar**	delete char
key_npage	**^F**	**^[^Fa**	**npage**	next page
key_ppage	**^B**	**^[^Ba**	**ppage**	previous page

key_sr	**^Y**	**^[^Ya**	**sr**	scroll up
key_eos	**dG**	**^[dGa**	**clreos**	clear to end of screen

EXTERNAL INFLUENCES

Support for international codes and environment variables are as follows:

Environment Variables

UNIX95 specifies using the XPG4 behaviour for this command.

COLUMNS overrides the system-selected horizontal screen size.

LINES overrides the system-selected vertical screen size, used as the number of lines in a screenful and the vertical screen size in visual mode.

SHELL is a variable that shall be interpreted as the preferred command-line interpreter for use in **!**, **shell**, **read**, and other commands with an operand of the form **!string**. For the **shell** command the program shall be invoked with the two arguments **-c** and **string**. If this variable is null or not set, the **sh** utility shall be used.

TERM is a variable that shall be interpreted as the name of the terminal type. If this variable is unset or null, an unspecified default terminal type shall be used.

PATH determines the search path for the shell command specified in the editor commands, **shell**, **read**, and **write**. **EXINIT** determines a list of *ex* commands that will be executed on editor startup, before reading the first file. The list can contain multiple commands by separating them using a vertical line (|) character.

HOME determines a pathname of a directory that will be searched for an editor startup file named **.exrc**.

LC_ALL This variable shall determine the locale to be used to override any values for locale categories specified by the setting of **LANG** or any environment variables beginning with **LC_**.

LC_MESSAGES determines the locale that should be used to affect the format and contents of diagnostic messages written to standard error and informative messages written to standard output.

LC_COLLATE determines the collating sequence used in evaluating regular expressions and in processing the *tags* file. **LC_CTYPE** determines the interpretation of text as single and/or multi-byte characters, the classification of characters as uppercase or lowercase letters, the shifting of letters between uppercase and lowercase, and the characters matched by character class expressions in regular expressions.

LANG determines the language in which messages are displayed.

LANGOPTS specifies options determining how text for right-to-left languages is stored in input and output files. See *environ*(5).

If **LC_COLLATE** or **LC_CTYPE** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**. If any internationalization variable contains an invalid setting, the editor behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported.

WARNINGS

See also the WARNINGS section in *ex*(1).

Program Limits

vi places the following limits on files being edited.

Maximum Line Length

LINE_MAX characters (defined in **<limits.h>**), including 2-3 bytes for overhead. Thus, if the value specified for **LINE_MAX** is 2048, a line length up to 2044 characters should cause no problem.

If you load a file that contain lines longer than the specified limit, the lines are truncated to the stated maximum length. Saving the file will write the truncated version over the original file, thus overwriting the original lines completely.

Attempting to create lines longer than the allowable maximum for the editor produces a **line too long** error message.

Maximum File Size

The maximum file length of 234,239 lines is silently enforced.

Other limits

- 256 characters per global command list.
- 128 characters in a file name in **vi** or **ex** **open** mode. On short-file-name HP-UX systems, the maximum file name length is 14 characters.
- 128 characters in a previous insert/delete buffer.
- 100 characters in a shell-escape command.
- 63 characters in a string-valued option (**:set** command).
- 30 characters in a program tag name.
- 32 or fewer macros defined by **map** command.
- 512 or fewer characters total in combined **map** macros.

Do not use the **-C** option to edit unencrypted files. The **-C** option is meant to be used only on files that are already encrypted. If the **-C** option is used on files which are not yet encrypted, a write in the edit session is likely to corrupt the file.

AUTHOR

vi was developed by the University of California, Berkeley. The 16-bit extensions to **vi** are based in part on software of the Toshiba Corporation.

SEE ALSO

ctags(1), ed(1), ex(1), stty(1), write(1), terminfo(4), environ(5), lang(5), regexp(5).

The Ultimate Guide to the vi and ex Text Editors, Benjamin/Cummings Publishing Company, Inc., ISBN 0-8053-4460-8, HP part number 97005-90015.

STANDARDS CONFORMANCE

vi: SVID2, SVID3, XPG2, XPG3, XPG4

NAME

vis, inv - make unprintable and non-ASCII characters in a file visible or invisible

SYNOPSIS

vis [**-n**] [**-s**] [**-t**] [**-u**] [**-x**] *file* ...

inv [**-n**] [**-s**] [**-t**] [**-u**] [**-x**] *file* ...

DESCRIPTION

vis reads characters from each *file* in sequence and writes them to the standard output, converting those that are not printable or not ASCII into a visible form. *inv* performs the inverse function, reading printable characters from each *file*, returning them to non-printable or non-ASCII form, if appropriate, then writing them to standard output;

Non-printable ASCII characters are represented using C-like escape conventions:

\	backslash
\b	backspace
\e	escape
\f	form-feed
\n	new-line
\r	carriage return
\s	space
\t	horizontal tab
\v	vertical tab
\*n*	the character whose ASCII code is the 3-digit octal number *n*.
\x*n*	the character whose ASCII code is the 2-digit hexadecimal number *n*.

Non-ASCII single- or multi-byte characters are examined one byte at a time. For each byte, if it can be displayed as an ASCII character, it is treated as if it is an ASCII character; Otherwise, it is represented in the following conventions:

\*n*	the 8-bit character whose code value is the 3-digit octal number *n*.
\x*n*	the 8-bit character whose code value is the 2-digit hexadecimal number *n*.

Space, horizontal-tab, and new-line characters can be treated as printable (and therefore passed unaltered to the output) or non-printable depending on the options selected. Backslash, although printable, is expanded by *vis*, to a pair of backslashes so that when they are passed back through *inv*, they convert back to a single backslash.

If no input file is given, or if the argument **-** is encountered, **vis** and *inv* read from the standard input.

Options

vis and **inv** recognize the following options:

-n Treat new-line, space, and horizontal tab as non-printable characters. **vis** expands them visibly as **\n**, **\s**, and **\t**, rather than passing them directly to the output. **inv** discards these characters, expecting only the printable expansions. New-line characters are inserted by **vis** every 16 bytes so that the output will be in a form that is usable by most editors.

-s Make **vis** and **inv** silent about non-existent files, identical input and output, and write errors. Normally, no input file can be the same as the output file unless it is a special file.

-t Treat horizontal-tab and space characters as non-printable in the same manner that **-n** treats them.

-u Cause output to be unbuffered (byte-by-byte); normally, output is buffered.

-x Cause **vis** output to be in hexadecimal form rather than the default octal form. Either form is accepted to **inv** as input.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the language in which messages are displayed.

International Code Set Support

Single- and multi-byte character code sets are supported.

AUTHOR

vis was developed by HP.

SEE ALSO

cat(1), echo(1), od(1).

WARNINGS

Redirecting output to an input file destroys the original data. Therefore, command forms such as

```
vis file1 file2 >file1
```

should be avoided unless the source file can be safely discarded.

NAME

vmstat - report virtual memory statistics

SYNOPSIS

vmstat [**-dnS**] [*interval* [*count*]]

vmstat -f | **-s** | **-z**

DESCRIPTION

The **vmstat** command reports certain statistics kept about process, virtual memory, trap, and CPU activity. It also can clear the accumulators in the kernel **sum** structure.

Options

vmstat recognizes the following options:

-d	Report disk transfer information as a separate section, in the form of transfers per second.
-n	Provide an output format that is more easily viewed on an 80-column display device. This format separates the default output into two groups: virtual memory information and CPU data. Each group is displayed as a separate line of output. On multiprocessor systems, this display format also provides CPU utilization on a per CPU basis.
-S	Report the number of processes swapped in and out (**si** and **so**) instead of page reclaims and address translation faults (**re** and **at**).
interval	Display successive lines which are summaries over the last *interval* seconds. If *interval* is zero, the output is displayed once only. If the **-d** option is specified, the column headers are repeated. If **-d** is omitted, the column headers are not repeated. The command **vmstat 5** prints what the system is doing every five seconds. This is a good choice of printing interval since this is how often some of the statistics are sampled in the system; others vary every second.
count	Repeat the summary statistics *count* times. If *count* is omitted or zero, the output is repeated until an interrupt or quit signal is received. From the terminal, these are commonly **^C** and **^**, respectively (see *stty*(1)).
-f	Report on the number of forks and the number of pages of virtual memory involved since boot-up.
-s	Print the total number of several kinds of paging-related events from the kernel **sum** structure that have occurred since boot-up or since **vmstat** was last executed with the **-z** option.
-z	Clear all accumulators in the kernel **sum** structure. This option is restricted to the super user.

If none of these options is given, **vmstat** displays a one-line summary of the virtual memory activity since boot-up or since the **-z** option was last executed.

Column Descriptions

The column headings and the meaning of each column are:

procs Information about numbers of processes in various states.

r	In run queue
b	Blocked for resources (I/O, paging, etc.)
w	Runnable or short sleeper (< 20 secs) but swapped

memory Information about the usage of virtual and real memory. Virtual pages are considered active if they belong to processes that are running or have run in the last 20 seconds.

avm	Active virtual pages
free	Size of the free list

page Information about page faults and paging activity. These are averaged each five seconds, and given in units per second.

	`re`	Page reclaims (without `-S`)
	`at`	Address translation faults (without `-S`)
	`si`	Processes swapped in (with `-S`)
	`so`	Processes swapped out (with `-S`)
	`pi`	Pages paged in
	`po`	Pages paged out
	`fr`	Pages freed per second
	`de`	Anticipated short term memory shortfall
	`sr`	Pages scanned by clock algorithm, per second
`faults`	Trap/interrupt rate averages per second over last 5 seconds.	
	`in`	Device interrupts per second (nonclock)
	`sy`	System calls per second
	`cs`	CPU context switch rate (switches/sec)
`cpu`	Breakdown of percentage usage of CPU time	
	`us`	User time for normal and low priority processes
	`sy`	System time
	`id`	CPU idle

EXAMPLES

The following examples show the output for various command options. For formatting purposes, some leading blanks have been deleted.

1. Display the default output.

`vmstat`

```
         procs            memory                        page
                   faults       cpu
    r     b        w      avm    free    re    at    pi    po    fr    de    sr
         in        sy     cs  us  sy id
    0     0        0     1158     511     0     0     0     0     0     0     0
        111        18      7   0   0 100
```

2. Add the disk tranfer information to the default output.

`vmstat -d`

```
         procs            memory                        page
                   faults       cpu
    r     b        w      avm    free    re    at    pi    po    fr    de    sr
         in        sy     cs  us  sy id
    0     0        0     1158     511     0     0     0     0     0     0     0
        111        18      7   0   0 100

    Disk Transfers
      device    xfer/sec
      c0t6d0        0
      c0t1d0        0
      c0t3d0        0
      c0t5d0        0
```

3. Display the default output in 80-column format.

`vmstat -n`

```
    VM
        memory                     page                            faults
      avm     free    re    at    pi   po    fr    de    sr    in      sy    cs
     1158      430     0     0     0    0     0     0     0   111      18     7
    CPU
        cpu            procs
     us sy id     r     b     w
      0  0 100    0     0     0
```

4. Replace the page reclaims and address translation faults with process swapping in the default output.

```
vmstat -S
```

```
          procs            memory                    page
                 faults        cpu
    r     b     w      avm     free    si    so    pi    po    fr    de    sr
          in     sy    cs  us sy id
    0     0     0     1158      430     0     0     0     0     0     0     0
        111     18     7   0  0 100
```

5. Display the default output twice at five-second intervals. Note that the headers are *not* repeated.

```
vmstat 5 2
```

```
          procs            memory                    page
                 faults        cpu
    r     b     w      avm     free    re    at    pi    po    fr    de    sr
          in     sy    cs  us sy id
    0     0     0     1158      456     0     0     0     0     0     0     0
        111     18     7   0  0 100
    0     0     0     1221      436     5     0     5     0     0     0     0
        108     65    18   0  1 99
```

6. Display the default output twice in 80-column format at five-second intervals. Note that the headers are *not* repeated.

```
vmstat -n 5 2
```

```
    VM
       memory                     page                          faults
      avm    free   re   at    pi   po    fr   de    sr     in      sy    cs
    1221     436    0    0     0    0     0    0     0    111      18     7
    CPU
         cpu            procs
      us sy id     r     b     w
       0  0 100    0     0     0
    1221     435    2    0     2    0     0    0     0    109      35    17
       0  1 99     0     0     0
```

7. Display the default output and disk transfers twice in 80-column format at five-second intervals. Note that the headers *are* repeated.

```
vmstat -dn 5 2
```

```
    VM
       memory                     page                          faults
      avm    free   re   at    pi   po    fr   de    sr     in      sy    cs
    1221     435    0    0     0    0     0    0     0    111      18     7
    CPU
         cpu            procs
      us sy id     r     b     w
       0  0 100    0     0     0

    Disk Transfers
       device    xfer/sec
       c0t6d0        0
       c0t1d0        0
       c0t3d0        0
       c0t5d0        0

    VM
       memory                     page                          faults
```

V

```
         avm    free   re   at    pi   po    fr   de    sr     in      sy    cs
        1219     425    0    0     0    0     0    0     0    111      54    15
        CPU
           cpu          procs
         us sy id    r     b     w
          1  8 92    0     0     0

        Disk Transfers
          device    xfer/sec
          c0t6d0       0
          c0t1d0       0
          c0t3d0       0
          c0t5d0       0
```

8. Display the number of forks and pages of virtual memory since boot-up.

```
vmstat -f

    24558 forks, 1471595 pages, average=  59.92
```

9. Display the counts of paging-related events.

```
vmstat -s

    0 swap ins
    0 swap outs
    0 pages swapped in
    0 pages swapped out
    1344563 total address trans. faults taken
    542093 page ins
    2185 page outs
    602573 pages paged in
    4346 pages paged out
    482343 reclaims from free list
    504621 total page reclaims
    124 intransit blocking page faults
    1460755 zero fill pages created
    404137 zero fill page faults
    366022 executable fill pages created
    71578 executable fill page faults
    0 swap text pages found in free list
    162043 inode text pages found in free list
    196 revolutions of the clock hand
    45732 pages scanned for page out
    4859 pages freed by the clock daemon
    36680636 cpu context switches
    1497746186 device interrupts
    1835626 traps
    87434493 system calls
```

WARNINGS

Users of `vmstat` must not rely on the exact field widths and spacing of its output, as these will vary depending on the system, the release of HP-UX, and the data to be displayed.

AUTHOR

`vmstat` was developed by the University of California, Berkeley and HP.

SEE ALSO

iostat(1).

NAME

vt - log into another system over lan

SYNOPSIS

/usr/bin/vt *nodename* [*lan_device*]

/usr/bin/vt -p [*lan_device*]

DESCRIPTION

vt enables a user to log into another HP 9000 system (*nodename*) over an HP local area network. The **-p** option causes **vt** to send a poll request over the local area network to find out what systems currently have **vtdaemon** running (see *vtdaemon*(1M)). An asterisk (*****) following a *nodename* in the response indicates that the system is a **vt** gateway. Plus signs (**+**) following the *nodename* indicate how many **vt** gateways must be traversed to reach that system.

The optional argument *lan_device* specifies a character special device name to use instead of the default device name to send and receive data to and from the local area network. The major number for this device must correspond to a CIO IEEE 802.3 local area network device.

Once a connection has been established, **vt** enters input mode. In this mode, text typed is sent to the remote host. To issue **vt** commands when in input mode, precede them with the **vt** escape character (see Commands below). When in command mode, normal terminal editing conventions are available.

The connection should terminate automatically upon termination of the login shell on the remote machine. If the connection is not terminated upon exit, it is likely that the **ptydaemon** on the remote system has either been terminated or restarted. To terminate a vt connection, enter command mode and use the **quit** command to terminate the connection.

Commands

vt recognizes the following commands. Commands can be abbreviated by typing enough of the command to uniquely identify it.

cd *remote-directory*	Change the working directory on the remote machine to *remote-directory*. This command is applicable for file transfer only.
escape [*escape-char*]	Set the **vt** escape character. If a character is not specified **vt** prompts for one. To print or display the current **vt** escape character simply press Return in response to this prompt.
help or **?**	Print a **vt** command summary.
lcd [*directory*]	Change the local working directory. If no *directory* is specified, use the user's home directory. This command is applicable for file transfer and shell escapes only.
get *remote-file local-file* **receive** *remote-file local-file*	Copy *remote-file* to local machine and store as *local-file*. **vt** prompts for the file names if they are not specified. The file transfer can be aborted by typing the interrupt character or pressing the break key.
put *local-file remote-file* **send** *local-file remote-file*	Copy *local-file* to the remote machine and store as *remote-file*. **vt** prompts for the file names if they are not specified. The file transfer can be aborted by typing the interrupt character or pressing the break key.
quit	Terminate the connection and exit *vt*.
user*user-name*[**:**[*password*]]	Identify yourself to the remote **vt** server. **vt** prompts for a password (after disabling local echo) if a colon (**:**) is appended to *user-name*. This command must be executed before any file transfer command can be used.
! [*shellcommand*]	Shell escape. The given command is passed to a sub-shell for execution. If no command is given, a shell is started and connected to the user's terminal.

Access Control Lists (ACLs)

When sending or receiving files using **vt**, optional ACL entries are removed. New files have a summary of the access modes (as returned in **st_mode** by **stat()** of the file being transferred (see *stat*(2)).

DIAGNOSTICS

The diagnostics produced by **vt** are intended to be self-explanatory.

WARNINGS

vt uses the Hewlett-Packard LLA (Link Level Access) direct interface to the HP network drivers. **vt** uses the multicast address **0x01AABBCCBBAA.** It should not be used or deleted by other applications accessing the network. **vt** uses the following IEEE 802.3 *sap* (service access point) values: **0x90**, **0x94**, **0x98**, **0x9C**, **0xA0**, **0xA4**, **0xA8**, **0xAC**, **0xB0**, **0xB4**, **0xB8**, **0xBC**, **0xC0**, **0xC4**, **0xC8**, **0xCC**, **0xD0**, and **0xD4**. They should not be used by other applications accessing the network.

When using **vt** on a system with multiple LAN cards installed, the optional command-line argument *lan_device* may be required if the remote system is not accessible through the default LAN device. The appropriate *lan_device* is the one connected (either directly or by way of other gateways) to the remote system.

Desktop HP-UX

If your system has been installed with the Desktop HP-UX product, neither **ptydaemon** nor **vtdaemon** will be started by default. To start these daemons, change *PTYDAEMON_START* and *VTDAEMON_START* from a **0** to a **1** in the **/etc/rc.config.d/ptydaemon** and **/etc/rc.config.d/vt** files, respectively. The system must be either rebooted for these changes to take effect, or you can start both daemons manually by typing the following commands:

```
/usr/sbin/ptydaemon
/usr/sbin/vtdaemon /dev/lan0
```

where */dev/lan0* is the character special device file corresponding to the IEEE 802.3 local area network device.

FILES

/dev/lan0	Default lan device name.
/etc/rc.config.d/ptydaemon	
/etc/rc.config.d/vt	

SEE ALSO

vtdaemon(1M), stat(2), lan(7), acl(5).

NAME

wait - await process completion

SYNOPSIS

wait [*pid*]

DESCRIPTION

If no argument is specified, **wait** waits until all processes (started with **&**) of the current shell have completed, and reports on abnormal terminations. If a numeric argument *pid* is given and is the process ID of a background process, **wait** waits until that process has completed. Otherwise, if *pid* is not a background process, **wait** exits without waiting for any processes to complete.

Because the **wait()** system call must be executed in the parent process, the shell itself executes **wait** without creating a new process (see *wait*(2)).

Command-Line Arguments

wait supports the following command line arguments:

pid The unsigned decimal integer process ID of a command, whose termination **wait** is to wait for.

WARNINGS

Some processes in a 2-or-more-stage pipeline may not be children of the shell, and thus cannot be waited for.

SEE ALSO

csh(1), ksh(1), sh-posix(1), sh(1), wait(2).

STANDARDS CONFORMANCE

wait: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

NAME

wc - count words, lines, and bytes or characters in a file

SYNOPSIS

wc [**-c**|**-m**] [**-lw**] [*file*]...

DESCRIPTION

The **wc** command counts lines, words, and bytes or characters in the named files, or in the standard input if no file names are specified. It also keeps a total count for all named files.

A word is a string of characters delimited by spaces, tabs, or newlines.

Options

wc recognizes the following options:

-c	Report the number of bytes in each input file.
-l	Report the number of newline characters in each input file.
-m	Report the number of characters in each input file.
-w	Report the number of words in each input file.

The **c** and **m** options are mutually exclusive. Otherwise, the **l**, **w**, and **c** or **m** options can be used in any combination to specify that a subset of lines, words, and bytes or characters are to be reported.

When any option is specified, **wc** reports only the information requested. If no option is specified, the default output is **-lwc**.

When a *file* is specified on the command line, its name is printed along with the counts.

Standard Output (XPG4 only)

By default, the standard output contains an entry for each input file in the form:

newlines words bytes file

If the **-m** option is specified, the number of characters replaces the *bytes* field in this format.

If any option is specified, the fields for the unspecified options are omitted.

If no *file* operand is specified, neither the file name nor the preceding blank character is written.

If more than one *file* operand is specified, an additional line is written at the end of the output, of the same format as the other lines, except that the word **total** (in the POSIX locale) is written instead of a file name and the total of each column is written as appropriate.

EXIT STATUS

wc exits with one of the following values:

0	Successful completion.
>0	An error occurred.

EXTERNAL INFLUENCES

Environment Variables

LC_CTYPE determines the range of graphics and space characters, and the interpretation of text as single- and/or multibyte characters.

LC_MESSAGES determines the language in which messages are displayed.

If **LC_CTYPE** or **LC_MESSAGES** is not specified in the environment or is null, they default to the value of **LANG**.

If **LANG** is not specified or is null, it defaults to **C** (see *lang*(5)).

If any internationalization variable contains an invalid setting, they all default to **C**. See *environ*(5).

International Code Set Support

Single- and multibyte character code sets are supported. with a newline character, the count will be off by one.

WARNINGS

The **wc** command counts the number of newlines to determine the line count. If a text file has a final line that is not terminated

EXAMPLES

Print the number of words and characters in **file1**:

```
wc -wm file1
```

The following is printed when the above command is executed:

words chars **file1**

where *words* is the number of words and *chars* is the number of characters in **file1**.

STANDARDS CONFORMANCE

wc: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

NAME

what - get SCCS identification information

SYNOPSIS

what [**-s**] *file* ...

DESCRIPTION

The **what** command searches the given files for all occurrences of the pattern that *get*(1) substitutes for **%Z%** (currently **@(#)** at this printing) and prints out what follows until the first ", **>**, new-line, ****, or null character. For example, if the C program in file **f.c** contains

```
char ident[] = "@(#)identification information";
```

and **f.c** is compiled to yield **f.o** and **a.out**, the command

```
what f.c f.o a.out
```

prints

```
f.c:        identification information
f.o:        identification information
a.out:      identification information
```

what is intended to be used in conjunction with the SCCS **get** command (see *get*(1)) which automatically inserts identifying information, but it can also be used where the information is inserted manually.

Options

what recognizes the following option:

-s	Quit after finding the first occurrence of pattern in each file.

EXTERNAL INFLUENCES

Environment Variables

LC_CTYPE determines the interpretation of the pattern substituted for **%Z%** as single- and/or multi-byte characters.

LC_MESSAGES determines the language in which messages are displayed.

If **LC_CTYPE** or **LC_MESSAGES** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**. If any internationalization variable contains an invalid setting, **what** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single-byte and multi-byte character code sets are supported.

DIAGNOSTICS

Exit status is 0 if any matches are found, otherwise 1. Use **help** for explanations (see *sccshelp*(1)).

WARNINGS

The pattern **@(#)** may occasionally appear unintentionally in random files, but this causes no harm in nearly all cases.

SEE ALSO

get(1), sccshelp(1).

STANDARDS CONFORMANCE

what: SVID2, SVID3, XPG2, XPG3, XPG4

NAME

whereis - locate source, binary, and/or manual for program

SYNOPSIS

whereis [**-bsm**] [**-u**] [**-BMS** *dir* ... **-f**] *name* ...

DESCRIPTION

whereis locates source, binary, and manuals sections for specified files. The supplied names are first stripped of leading path name components and any (single) trailing extension of the form **.***ext* (such as **.c**). Prefixes of **s.** resulting from use of SCCS are also dealt with. **whereis** then attempts to locate the desired program in a list of standard places.

Options

whereis recognizes the following command-line options:

-b	Limit search to binary files. Can be used in conjunction with **-s** or **-m**.
-s	Limit search to source-code files. Can be used in conjunction with **-b** or **-m**.
-m	Limit search to manual entry files. Can be used in conjunction with **-b** or **-s**.
-u	Search for unusual entries. A file is said to be unusual if it does not have one entry of each requested type. Thus, **whereis -m -u *** searches for those files in the current directory that have no corresponding manual entry.
-B *dir* [*dir* ...]	Limit search to binaries located in one or more specified directories. Can be used in conjunction with **-S** or **-M**.
-S *dir* [*dir* ...]	Limit search to source files located in one or more specified directories. Can be used in conjunction with **-B** or **-M**.
-M *dir* [*dir* ...]	Limit search to manual entry files located in one or more specified directories. Can be used in conjunction with **-B** or **-S**.
-f	Terminates the last directory list (**-B**, **-S**, or **-M** options) and identifies the start of file names.

EXTERNAL INFLUENCES

International Code Set Support

Single- and multi-byte character code sets are supported.

EXAMPLES

Find all the files in **/usr/bin** that are not documented in **/usr/share/man/man1** with source files in **/usr/src/cmd**:

```
cd /usr/bin
whereis -u -M /usr/share/man/man1 -S /usr/src/cmd -f *
```

WARNINGS

whereis uses the **chdir()** system call (see *chdir*(2)) to run faster. Therefore, path names given with the **-B**, **-M**, and **-S** options must be absolute path names.

AUTHOR

whereis was developed by the University of California, Berkeley.

W

FILES

```
/usr/src/*
/usr/sbin, /sbin, /usr/bin, /usr/lbin, /usr/ccs/bin
/usr/share/man/*
/usr/local/{man/*, bin, games, include, lib}
/usr/contrib/{man/*, bin, games, include, lib}
/usr/share/man/$LANG/*
/usr/local/man/$LANG/*
/usr/contrib/man/$LANG/*
```

NAME

which - locate a program file including aliases and paths

SYNOPSIS

which [*name* ...]

DESCRIPTION

For each *name* given, **which** searches for the file that would be executed if *name* were given as a command, and displays the absolute path of that file. Each argument is expanded if it is aliased, and searched for along the user's path. Both aliases and path are determined by sourcing (executing) the user's **.cshrc** file.

DIAGNOSTICS

A diagnostic is given for names that are aliased to more than a single word, or if an executable file with the argument name was not found in the path.

EXAMPLES

The command:

```
which sh
```

specifies where the executable program of the *sh*(1) command is found. For example, the response might be:

```
/usr/bin/sh
```

if the *sh*(1) being used is located in **/usr/bin**.

WARNINGS

which reports **.cshrc** aliases even when not invoked from *csh*.

which cannot find **csh** built-in commands (e.g. jobs).

which's information may be incorrect because it is unaware of any path or alias changes that have occurred in the current shell session.

AUTHOR

which was developed by the University of California, Berkeley.

FILES

~/.cshrc	source of aliases and path values

NAME

who - who is on the system

SYNOPSIS

who [**-muTlHqpdbrtasAR**] [*file*]

who am i

who am I

DESCRIPTION

The **who** command can list the user's name, terminal line, login time, elapsed time since input activity occurred on the line, the user's host name, and the process-ID of the command interpreter (shell) for each current system user. It examines the **/etc/utmp** file to obtain its information. If *file* is given, that file is examined. Usually, *file* is **/var/adm/wtmp**, which contains a history of all of the logins since the file was last created.

The **who** command with the **am i** or **am I** option identifies the invoking user.

Except for the default **-s** option, the general format for output entries is:

name [state] line time activity pid [comment] [exit]

With options, **who** can list logins, logoffs, reboots, and changes to the system clock, as well as other processes spawned by the **init** process.

Options

-m Output only information about the current terminal. This option is equivalent to the **am i** and **am I** options described above.

-u Lists only those users who are currently logged in. *name* is the user's login name. *line* is the name of the line as found in the directory **/dev**. The *time* field indicates when the user logged in.

activity is the number of hours and minutes since input activity last occurred on that particular line. A dot (**.**) indicates that the terminal has seen activity in the last minute and is therefore "current". If more than twenty-four hours have elapsed or the line has not been used since boot time, the entry is marked **old**. This field is useful when trying to determine whether a person is working at the terminal or not. The *pid* is the process-ID of the user's login process. The *comment* is the comment field associated with this line as found in **/etc/inittab** (see *inittab*(4)). This can contain information about where the terminal is located, the telephone number of the dataset, type of terminal if hard-wired, etc. If no such information is found, then **who** prints, as the *comment*, the user's host name as it was stored in the **/etc/utmp** or named *file*. Note that the user's host name is printed instead of comments from the **/etc/inittab** file if the **-u** option is used in conjunction with the **-R** option.

-T Same as the **-u** option, except that the *state* of the terminal line is printed. *state* describes whether someone else can write to that terminal. A **+** appears if the terminal is writable by anyone; a **-** appears if it is not. **root** can write to all lines having a **+** or a **-** in the *state* field. If a bad line is encountered, a **?** is printed.

(XPG4 only.) Only the following fields are displayed: *name state line time*

-l Lists only those lines on which the system is waiting for someone to login. The *name* field is **LOGIN** in such cases. Other fields are the same as for user entries except that the *state* field does not exist.

-H Prints column headings above the regular output.

-q A quick **who**, displaying only the names and the number of users currently logged in. When this option is used, all other options are ignored.

-p Lists any other process which is currently active and has been previously spawned by *init*. The *name* field is the name of the program executed by **init** as found in **/etc/inittab**. The *state*, *line*, and *activity* fields have no meaning. The *comment* field shows the *id* field of the line from **/etc/inittab** that spawned this process. See *inittab*(4).

-d This option displays all processes that have expired and have not been respawned by **init**. The *exit* field appears for dead processes and contains the termination and exit values of the dead process (as returned by **wait()** — see *wait*(2)). This can be useful in determining why a process terminated.

-b Indicates the time and date of the last reboot.

-r Indicates the current *run-level* of the **init** process. The last three fields contain the current state of **init**, the number of times that state has been previously entered, and the previous state. These fields are updated each time **init** changes to a different run state.

-t Indicates the last change to the system clock (via the **date** command) by **root**. See *su*(1).

-a Processes **/etc/utmp** or the named *file* with all options turned on.

-s Default. Lists only the *name*, *line*, and *time* fields.

-A When the **/var/adm/wtmp** file is specified, this option indicates when the accounting system was turned on or off using the **startup** or **shutacct** commands (see *acctsh*(1M)). The *name* field is **.** . The *line* field is **acctg on**, **acctg off**, or a reason that was given as an option to the **shutacct** command. The *time* is the time that the on/off activity occurred.

-R Displays the user's host name. If the user is logged in on a tty, **who** displays the string returned from **gethostname()** (see *gethostname*(2)). If the user is not logged in on a tty and the host name stored in the **/etc/utmp** or named file has not been truncated when stored (meaning that the entire host name was stored with no loss of information), it is displayed as it was stored. Otherwise, the **gethostbyaddr()** function is called with the internet address of the host (see *gethostent*(3N)). The host name returned by **gethostbyaddr()** is displayed unless it returns an error, in which case the truncated host name is displayed.

(XPG4 only. The **-s** option can not be used with **-d**, **-a** or **-T** options. If **-u** option is used with **-T**, the idle time is added to the end of the **-T** format.)

EXTERNAL INFLUENCES

Environment Variables

LANG determines the locale to use for the locale categories when both **LC_ALL** and the corresponding environment variable (beginning with **LC_**) do not specify a locale. If **LANG** is not set or is set to the empty string, a default of "C" (see *lang*(5)) is used.

LC_CTYPE determines the locale for interpretation of sequences of bytes of text data as characters (e.g., single- verses multibyte characters in arguments and input files).

LC_TIME determines the format and contents of date and time strings.

LC_MESSAGES determines the language in which messages are displayed.

If any internationalization variable contains an invalid setting, **who** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported.

EXAMPLES

Check who is logged in on the system:

```
who
```

Check whether or not you can write to the terminal that another user is using:

```
who -T
```

and look for a plus (**+**) after the user ID.

AUTHOR

who was developed by AT&T and HP.

FILES

`/etc/inittab`
`/etc/utmp`
`/var/adm/wtmp`

SEE ALSO

date(1), login(1), init(1M), mesg(1), su(1), gethostname(2), wait(2), gethostent(3N), inittab(4), utmp(4).

STANDARDS CONFORMANCE

`who`: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

NAME
whoami - print effective current user id

SYNOPSIS
whoami

DESCRIPTION
whoami prints your *current* user name, even if you have used **su** to change it since your initial login (see *su*(1)). The command **who am i** reports your initial login name because it uses **/etc/utmp**.

FILES
/etc/passwd name data base

AUTHOR
whoami was developed by the University of California, Berkeley.

SEE ALSO
who(1).

NAME

whois - Internet user name directory service

SYNOPSIS

whois [**-h** *hostname*] *name*

DESCRIPTION

whois looks up records in the Network Information Center database.

The operands specified to whois are concatenated together (separated by white-space) and presented to the whois server.

The default action, unless directed otherwise with a special name, is to do a very broad search, looking for matches to name in all types of records and most fields (name, nicknames, hostname, net address, etc.) in the database. For more information as to what name operands have special meaning, and how to guide the search, use the special name ‘‘help’’.

Options

whois supports the following options:

-h use the specified host instead of the default (nic.ddn.mil).

EXAMPLES

Look up user John Doe

whois *Doe,John*

DEPENDENCIES

The machine making the **whois** enquiry must be able to route the request over the network to the specified host, or to the default NIC (nic.ddn.mil).

AUTHOR

whois was developed by the University of California, Berkeley.

SEE ALSO

RFC 854: Nicname/Whois

NAME

write - interactively write (talk) to another user

SYNOPSIS

write *user* [*terminal*]

DESCRIPTION

The **write** command copies lines from your terminal to that of another user. When first called, it sends the message:

> **Message from** *yourname* **(***yourterminal***) [** *date* **] ...**

to the receiving *user*'s terminal. When it has successfully completed the connection, it also sends two bells to your own terminal to indicate that what you are typing is being sent.

To set up two-way communication, the recipient of the message (*user*) must execute the command:

> **write** *yourname* [*yourterminal*]

(*yourterminal* is only required if the originator is logged in more than once.)

Communication continues until an end of file is read from the terminal, an interrupt is sent, or the recipient executes **mesg n**. At that point, **write** writes **<EOT>** on the other terminal and exits.

To write to a user who is logged in more than once, use the *terminal* argument to indicate which line or terminal to send to (e.g., **tty00**). Otherwise, the first writable instance of the user found in **/etc/utmp** is assumed and the following message is displayed:

```
user is logged on more than one place.
You are connected to "terminal".
Other locations are:
terminal
...
```

Permission to write may be denied or granted with the **mesg** command (see *mesg*(1)). Writing to others is normally allowed by default. Certain commands, in particular **nroff** and **pr** disallow messages in order to prevent interference with their output. However, if the user has the appropriate privileges, messages can be forced onto a write-inhibited terminal.

If the character **!** is found at the beginning of a line, **write** calls the Bourne shell (see *sh-bourne*(1)) to execute the rest of the line as a command.

The following protocol is suggested for using **write**: When you first **write** to another user, wait for the user to **write** back before starting to send. Each person should end a message with a distinctive signal (such as "**(o)**" for "over") so that the other person knows when to reply. Similarly, the signal "**(oo)**" (for "over and out") can be used to indicate the end of the conversation.

EXTERNAL INFLUENCES

Environment Variables

LANG determines the locale to use for the locale categories when both **LC_ALL** and the corresponding environment variable (beginning with **LC_**) do not specify a locale. If **LANG** is not set or is set to the empty string, a default of "C" (see *lang*(5)) is used.

LC_TIME determines the format and contents of date and time strings.

LC_MESSAGES determines the language in which messages are displayed.

If any internationalization variable contains an invalid setting, **write** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported.

DIAGNOSTICS

user **is not logged on.**
The user you are trying to write to is not logged on.

Can no longer write to *terminal*
Your correspondent has denied write permission (**mesg n**) after your **write** session started. Your **write** session is ended.

<EOT>
Your correspondent sent end-of-file, or you set your terminal to **mesg n** and your correspondent tried to write to you. If you have a **write** session established, you can continue to write to your correspondent.

Permission denied.
The user you are trying to write to has denied write permission (with **mesg n**).

Warning: You have your terminal set to "mesg -n". No reply possible.
Your terminal is set to **mesg n** and the recipient cannot respond to you.

EXAMPLES

By issuing the command:

```
write matthew
```

user **linda** sends a message to user **matthew**'s screen. If **matthew** responds:

```
write linda
```

two-way communication between **matthew** and **linda** is established.

FILES

/etc/utmp	To find user
/usr/bin/sh	To execute **!** shell commands

SEE ALSO

elm(1), mail(1), mailx(1), mesg(1), nroff(1), pr(1), sh(1), who(1).

STANDARDS CONFORMANCE

write: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

NAME

xargs - construct argument list(s) and execute command

SYNOPSIS

xargs [*options*] [*command* [*initial-arguments*]]

DESCRIPTION

xargs combines the fixed *initial-arguments* with arguments read from standard input to execute the specified *command* one or more times. The number of arguments read for each *command* invocation and the manner in which they are combined are determined by the options specified.

command, which can be a shell file, is searched for, using the **$PATH** environment variable. If *command* is omitted, **/usr/bin/echo** is used.

Arguments read in from standard input are defined to be contiguous strings of characters delimited by one or more blanks, tabs, or new-lines; empty lines are always discarded. Spaces and tabs can be embedded as part of an argument if escaped or quoted. Characters enclosed in quotes (single or double) are taken literally, and the delimiting quotes are removed. Outside of quoted strings, a backslash (\) escapes the next character.

The amount of memory available for the execution of *command* is limited by the system parameter **ARG_MAX**. By default, the size of the argument list is limited to **LINE_MAX** bytes. See *limits*(5) and *sysconf*(2) for a description of these system parameters and how their values can be determined. To increase the available argument list space, use the **-s** option.

Each argument list is constructed starting with the *initial-arguments*, followed by some number of arguments read from standard input (exception: see **-i** or **-I** option). The **-i**, **-I**, **-l**, **-L**, and **-n** options determine how arguments are selected for each command invocation. When none of these options is specified, the *initial-arguments* are followed by arguments read continuously from standard input until an internal buffer is full, then *command* is executed with the accumulated args. This process is repeated until there are no more args. When there are option conflicts (such as **-l** or **-L** versus **-n**), the last option has precedence. *option* values are:

-L *number*
: *command* is executed for each non-empty *number* lines of arguments from standard input. The last invocation of *command* will be with fewer lines of arguments if fewer than *number* remain. A line is considered to end with the first new-line *unless* the last character of the line is a blank or a tab; a trailing blank/tab signals continuation through the next non-empty line. The **-L**, **-l**, and **-n** options are mutually exclusive. The last one specified takes effect.

-l[*number*]
: This option is equivalent to the **-L** option. **1** is assumed if *number* is omitted or is given as the empty string (""). Option **-x** is forced.

-I *replstr*
: Insert mode: *command* is executed for each line from standard input, taking the entire line as a single arg, inserting it in *initial-arguments* for each occurrence of *replstr*. A maximum of 5 arguments in *initial-arguments* can each contain one or more instances of *replstr*. Blanks and tabs at the beginning of each line are discarded. Constructed arguments must not grow larger than 255 bytes, and option **-x** is also forced. The **-I** and **-i** options are mutually exclusive. The last one specified takes effect.

-i[*repstr*]
: This option is equivalent to the **-I** option. **{}** is assumed if *replstr* is omitted or is given as the empty string ("").

-n *number*
: Execute *command* using as many standard input arguments as possible, up to *number* arguments maximum. Fewer arguments are used if their total size is greater than *size* bytes, and for the last invocation if there are fewer than *number* arguments remaining. If option -x is also coded, each *number* arguments must fit in the *size* limitation or **xargs** terminates execution.

-s *size*
: The maximum total size of each argument list is set to *size* bytes; *size* must be a positive integer less than **LINE_MAX** (see *limits*(5), *sysconf*(2)). If **-s** is not coded, **LINE_MAX** is taken as the default. Note that the bytes count for *size* includes one extra bytes for each argument and the count of bytes in the command name.

-t
: Trace mode: The *command* and each constructed argument list are echoed to standard error just prior to their execution.

-p Prompt mode: The user is asked whether to execute *command* prior to each invocation. Trace mode (**-t**) is turned on to print the command instance to be executed, followed by a **?...** prompt. An affirmative reply (by default, an affirmative reply is **y** optionally followed by anything) executes the command; anything else, including pressing Return, skips that particular invocation of *command*.

-x Causes *xargs* to terminate if any argument list would be greater than *size* bytes. **-x** is forced by the options **-i**, **-I**, **-l**, and **-L**. When none of the options **-i**, **-I**, **-l**, **-L**, or **-n** is coded, the total length of all arguments must be within the *size* limit.

-e[*eofstr*] *eofstr* is taken as the logical end-of-file string. Underscore (_) is assumed for the logical **EOF** string if neither **-e** nor **-E** is used. The value **-e** with *eofstr* given as the empty string ("") turns off the logical **EOF** string capability (underscore is taken literally). *xargs* reads standard input until either end-of-file or the logical **EOF** string is encountered.

-E *eofstr* Specify a logical end-of-file string to replace the default underscore (_) character. Equivalent to the **-e** option above.

xargs terminates if it receives a return code of −1 from *command* or if it cannot execute *command*. When *command* is a shell program, it should explicitly **exit** (see *sh*(1)) with an appropriate value to avoid accidentally returning with −1.

RETURN VALUE

xargs exits with one of the following values:

0 All invocations of *command* completed successfully.

1-125
One or more invocations of *command* did not complete successfully.

126 The *command* specified was found but could not be invoked.

127 The *command* specified could not be found.

EXTERNAL INFLUENCES

Environment Variables

LC_CTYPE determines the space characters and the interpretation of text as single- and/or multi-byte characters.

LC_MESSAGES determines the language in which messages are displayed, and the local language equivalent of an affirmative reply when the **-p** prompt option is specified.

If **LC_CTYPE** or **LC_MESSAGES** is not specified in the environment or is set to the empty string, the value of **LANG** is used as a default for each unspecified or empty variable. If **LANG** is not specified or is set to the empty string, a default of "C" (see *lang*(5)) is used instead of **LANG**.

If any internationalization variable contains an invalid setting, **xargs** behaves as if all internationalization variables are set to "C". See *environ*(5).

International Code Set Support

Single- and multi-byte character code sets are supported.

EXAMPLES

Move all files from directory **$1** to directory **$2**, and echo each move command just before doing it:

```
ls $1 | xargs -i -t mv $1/{} $2/{}
```

Combine the output of the parenthesized commands onto one line, then echo to the end of file **log**:

```
(logname; date; echo $0 $*) | xargs >>log
```

Ask the user which files in the current directory are to be archived then archive them into **arch** one at a time:

```
ls | xargs -p -l ar r arch
```

or many at a time:

```
ls | xargs -p -l | xargs ar r arch
```

Execute **diff** (see *diff*(1)) with successive pairs of arguments originally typed as shell arguments:

```
echo $* | xargs -n2 diff
```

SEE ALSO

sh(1).

STANDARDS CONFORMANCE

xargs: SVID2, SVID3, XPG2, XPG3, XPG4, POSIX.2

NAME

xstr - extract strings from C programs to implement shared strings

SYNOPSIS

xstr [**-c**] [**-**] [*file*]

DESCRIPTION

xstr maintains a file **strings** into which strings in component parts of a large program are hashed. These strings are replaced with references to this common area. This serves to implement shared constant strings, which are most useful if they are also read-only.

The command:

> **xstr -c** *name*

extracts the strings from the C source in *name*, replacing string references with expressions of the form **(&xstr[***number***])** for some *number*. An appropriate declaration of **xstr** is placed at the beginning of the file. The resulting C text is placed in the file **x.c**, for subsequent compiling. The strings from this file are placed in the **strings** database if they are not there already. Repeated strings and strings that are suffixes of existing strings do not cause changes to the data base.

After all components of a large program have been compiled, a file **xs.c** declaring the common **xstr** space, can be created by the command:

> **xstr**

This **xs.c** file should then be compiled and loaded with the rest of the program. If possible, the array can be made read-only (shared), saving space and swap overhead.

xstr can also be used on a single file. A command:

> **xstr** *name*

creates files **x.c** and **xs.c** as before, without using or affecting any **strings** file in the same directory.

It may be useful to run **xstr** after the C preprocessor if any macro definitions yield strings or if there is conditional code containing strings that are not, in fact, needed. **xstr** reads from its standard input when the argument **-** is given. An appropriate command sequence for running **xstr** after the C preprocessor is:

```
cc -E name.c | xstr -c -
cc -c x.c
mv x.o name.o
```

xstr does not touch the file **strings** unless new items are added, thus **make** can avoid remaking **xs.o** unless truly necessary (see *make*(1)).

WARNINGS

If a string is a suffix of another string in the data base, but the shorter string is seen first by **xstr**, both strings are placed in the data base, when placing only the longer one there would be sufficient.

AUTHOR

xstr was developed by the University of California, Berkeley.

FILES

strings	Data base of strings
x.c	Massaged C source
xs.c	C source for definition of array **xstr**
/tmp/xs*	Temp file when 'xstr name' does not touch **strings**

X

SEE ALSO

mkstr(1).

NAME

yes - be repetitively affirmative

SYNOPSIS

yes [*expletive*]

DESCRIPTION

yes repeatedly outputs **y**, or if *expletive* is given, the *expletive* is output repeatedly. Termination is by interrupt.

AUTHOR

yes was developed by the University of California, Berkeley.

NAME

ypcat - print all values in Network Information Service map

SYNOPSIS

ypcat [**-k**] [**-t**] [**-d** *domain*] *mname*

ypcat -x

Remarks

The Network Information Service (NIS) was formerly known as Yellow Pages (YP). The functionality of the two remains the same; only the name has changed.

DESCRIPTION

ypcat prints all values in a Network Information Service (NIS) map specified by *mname*, which can be either a *mapname* or a map *nickname*. A map *nickname* is a synonym by which a NIS map can be referenced. Values are listed, one per line.

Options

ypcat recognizes the following options:

-k Print the associated key preceding each value. This option is useful for examining maps in which the values are null or the keys are not part of the value, such as the *ypservers* map. The maps derived from files that have an ASCII version in **/etc** (such as **passwd** and **hosts**) are not in this category.

-t Inhibit the translation of a map's *nickname* to its corresponding *mapname*. For example, **ypcat -t passwd** fails because there is no map named **passwd**, whereas **ypcat passwd** translates to **ypcat passwd.byname**.

-d Specify a *domain* other than the one returned by **domainname** (see *domainname*(1)).

-x Display the table that lists the *nickname* for each NIS map.

AUTHOR

ypcat was developed by Sun Microsystems, Inc.

EXAMPLES

Display the network-wide password database whose *mapname* is **passwd.byname** and *nickname* is **passwd :**

```
ypcat passwd
```

SEE ALSO

domainname(1), ypmatch(1), ypserv(1M), ypfiles(4).

y

NAME

ypmatch - print values of selected keys in Network Information Service map

SYNOPSIS

ypmatch [**-k**] [**-t**] [**-d** *domain*] **key** [**key**...] **mname**

ypmatch -x

Remarks

The Network Information Service (NIS) was formerly known as Yellow Pages (yp). Although the name has changed, the functionality of the service remains the same.

DESCRIPTION

ypmatch prints the values associated with one or more keys in a Network Information Service (NIS) map specified by *mname*. The *mname* can be either a *mapname* or a map *nickname*. A map *nickname* is a synonym by which a NIS map can be referenced.

If multiple keys are specified, the same map is searched for an occurrence of each key. A match is made only when the case and length of a key is the same as that stored in the database. No pattern matching is available. If a key is not matched, a diagnostic message is produced.

Options

ypmatch recognizes the following command-line options:

-k Before printing the value associated with a key, print the key followed by a colon (**:**). This option is useful if the keys are not part of the values (as in a **ypservers** map), or so many keys were specified that the output could be confusing.

-t Inhibit the translation of a map's *nickname* to its corresponding *mapname*. For example, **ypmatch -t zippy passwd** fails because there is no map named **passwd**, while **ypmatch zippy passwd** is translated to **ypmatch zippy passwd.byname**.

-d Specify a *domain* other than the one returned by **domainname** (see *domainname*(1)).

-x Display the table that lists the *nickname* for each NIS map.

AUTHOR

ypmatch was developed by Sun Microsystems, Inc.

SEE ALSO

domainname(1), ypcat(1), ypserv(1M), ypfiles(4).

NAME

yppasswd - change login password in Network Information System (NIS)

SYNOPSIS

yppasswd [*name*]

Remarks

The Network Information Service (NIS) was formerly known as Yellow Pages (YP). The functionality remains the same; only the name has changed.

DESCRIPTION

yppasswd changes or installs a password associated with the login *name* in the Network Information System (NIS). The NIS password can be different from the one on your own machine. If *name* is omitted, it defaults to the name returned by **getlogin()** (see *getlogin*(3C)).

yppasswd prompts for the old NIS password (even if it does not exist), then twice for the new one. The old password must be entered correctly for the change to take effect. Checks occur to ensure that the new password meets the following construction requirements.

- Only the first eight characters are significant.
- A password can be as few as four characters long if it contains
  - at least one special character or
  - a mixture of numeric, uppercase and lowercase letters.
- A password can be as few as five characters long if it contains a mixture of
  - uppercase and lowercase letters or
  - numeric and either uppercase or lowercase letters.
- A password must contain at least six characters if it contains only monocase letters.

All these rules except the first are relaxed if you try three times to enter an unacceptable new password. You cannot, however, enter a null password.

Only the owner of the *name* or the superuser can change a password.

The Network Information System password daemon, *yppasswdd*(1M), must be running on the master NIS password server to change NIS passwords.

WARNINGS

The password update protocol passes the old and new passwords to the master NIS server at once. Thus, if the old NIS password is incorrect, no notification is given until the new NIS password is successfully entered.

The **yppasswd** password construction rules are different from those of the HP-UX **passwd** command (see *passwd*(1)).

User applications that call this routine must be linked with **/usr/include/librpcsvc.a**. For example,

```
cc my_source.c -lrpcsvc
```

AUTHOR

yppasswd was developed by Sun Microsystems, Inc.

SEE ALSO

id(1), passwd(1), su(1), yppasswdd(1M), getlogin(3C), yppasswd(3N), ypfiles(4).

NAME

ypwhich - list which host is Network Information System server or map master

SYNOPSIS

ypwhich

ypwhich [**-d** *domain*] [**-V1** | **-V2**] [*hostname*]

ypwhich [**-d** *domain*] [**-t**] [**-m** [*mname*]]

ypwhich -x

Remarks

The Network Information Service (NIS) was formerly known as Yellow Pages (yp). Although the name has changed, the functionality of the service remains the same.

DESCRIPTION

ypwhich lists the host name of the Network Information System (NIS) server that supplies NIS services to a NIS client. It can also print the NIS server that is the master for *mname*. The *mname* can be either a *mapname* or a map *nickname*. A map *nickname* is a synonym by which a NIS map can be referenced.

If invoked without arguments, **ypwhich** prints the host name of the NIS server serving the local machine. If *hostname* is specified, that machine is queried to determine which NIS server it is using.

Options

ypwhich recognizes the following command-line options and arguments:

-d Specify a *domain* other than the one returned by *domainname*(1).

-V1 List the server that is serving Version 1 NIS protocol-speaking client processes.

-V2 List the server that is serving Version 2 NIS protocol-speaking client processes.

If neither version is specified, **ypwhich** locates the server supplying the Version 2 (current) services. However, if no Version 2 server is found, **ypwhich** attempts to locate the server supplying the Version 1 services. Since NIS servers and NIS clients are both backward compatible, the user seldom needs to know which version is being used.

-t Inhibit the translation of a map's *nickname* to its corresponding *mapname*. For example, **ypwhich -t -m passwd** fails because there is no map named **passwd**, whereas **ypwhich -m passwd** translates to **ypwhich -m passwd.byname**. This option is useful if a *mapname* is identical to a *nickname* (which is not true of any HP map).

-m [*mname*]
List the master NIS server for a map. No *hostname* can be specified with **-m**. The *mname* can be a *mapname* or a map *nickname*. If *mname* is omitted, a complete list of available maps and the corresponding host names of the master NIS servers is produced.

-x Display the table that lists the *nickname* for each NIS map.

AUTHOR

ypwhich was developed by Sun Microsystems, Inc.

SEE ALSO

domainname(1), ypserv(1M), ypset(1M), ypfiles(4).