POSIX Conformance Document

HP 9000 Computers

HEWLETT PACKARD

HP Part No. B2355-90049
Printed in USA June 1994

Third Edition
E0694

Legal Notices

Printing History

New editions of this manual will incorporate all material updated since the previous edition.

The manual printing date and part number indicate its current edition. The printing date changes when a new edition is printed. (Minor corrections and updates which are incorporated at reprint do not cause the date to change.) The manual part number changes when extensive technical changes are incorporated.

August 1992. Second Edition. Replaces previous First Edition, HP Part Number B1864-90001. Updated to reflect conformance to POSIX.2 at HP-UX Release 9.0. Deleted conformance information for POSIX.1-1988 which is no longer necessary with adoption of FIPS 151-2 which is based on POSIX.1-1990.

June 1994. Third Edition. Replaces Second Edition, HP Part Number B2355-90034. Revised to delete additional conformance information not permitted by FIPS 151-2, and to improve correctness.

This edition of the *POSIX Conformance Document* consists of two parts:

- Part 1 is the POSIX.1-1990 Conformance Document which contains implementation-specific information required by IEEE Standard 1003.1-1990. Federal Information Processing Standard (FIPS) 151-2 also refers to this specification.
- Part 2 is the POSIX.2-1992 Conformance Document which contains implementation-specific information required by IEEE Standard 1003.2-1992, also called POSIX.2-1992 or simply POSIX.2.

0 Introduction

This Conformance Document is divided into two parts, each partaining to a corresponding POSIX standard:

- Part 1 contains the POSIX.1-1990 Conformance Document, and describes those items specified in the POSIX.1-1990 standard as implementation-defined that must be documented in order for the HP-UX operating system to claim conformance to it. (If you require a copy of the POSIX.1-1988 Conformance Document, obtain the previous edition of this document, HP part number B1864-90001.)

  Part 1 applies to the IEEE Standard Portable Operating System Interface for Computer Environments, IEEE Standard 1003.1-1990 (also known as ISO/IEC 9945-1: 1990), referred to herein as POSIX or POSIX.1. Other standards are also specified in various chapters as applicable.

- The POSIX.2-1992 Conformance Document describes those items specified in the POSIX.2-1992 standard as implementation-defined that must be documented in order for the HP-UX operating system to claim conformance to it.

  Part 2 applies to the IEEE Standard Portable Operating System Interface for Computer Environments, IEEE Standard 1003.2-1992, referred to herein as POSIX or POSIX.2. Other standards are also specified in various chapters as applicable.

This document contains references to other HP-UX system manuals, particularly the *HP-UX Reference*. References to entries in the *HP-UX Reference* are of the form *name*(1), *name*(2), and so forth where *name* is the name of the entry, and the number between parentheses indicates the section in the *HP-UX Reference* where the entry is found.

The remainder of this document is arranged by chapter where the chapter number sequence corresponds to chapter numbers in the POSIX.1-1990

Standard for Part 1 or the POSIX.2-1992 Standard for Part 2, and section and subsection numbers correspond to sections and subsections in the respective standards that require coverage in this document.

Part I
POSIX.1-1990

Contents

1

General

1.3 Conformance

1.3.1 Implementation Conformance

1.3.1.1 Requirements

HP-UX supports many facilities beyond POSIX. Use of some of these extensions can generate an environment in which a Conforming POSIX Application would not always see behavior conforming to the POSIX standard. The following guidelines can be observed to ensure this does not occur. Note that it takes explicit action on the part of the system administrator, and possibly on the part of the user as well, to violate these guidelines, and thus they should only be of concern when use of the HP-UX facilities mentioned is desired.

- The HP-UX kernel must be configured with the value of `maxuprc` greater than or equal to the value of {`_POSIX_CHILD_MAX`} defined in POSIX.1. See the *System Administrator Tasks* and *System Administrator Concepts* manuals for more details.
- A process requiring POSIX conformance should not have its root directory modified (see *chroot*(1) or *chroot*(2) in the *HP-UX Reference* for more details). It may be possible to create a file hierarchy with a different root directory that does result in full POSIX conformance, but the information and support to do this are not provided.
- A process requiring POSIX conformance must not have an effective group ID or supplementary group ID with the privilege `SETRUGID`. See *setprivgrp*(1m) and *setuid*(2) in the *HP-UX Reference* for more details.

- An application requiring POSIX conformance must not access any files over a network by use of NFS^{TM} network services. This can be enforced absolutely by the system administrator by not setting up NFS or by not mounting any NFS file systems. It can also be observed on an individual basis if NFS is being used. See the NFS networking manuals provided with your system for more details.
- Any terminal being used in a way requiring POSIX conformance must not have a delayed suspend character defined. See *termio*(7) in the *HP-UX Reference* for more details.
- An application requiring POSIX conformance must not access any files on a CD-ROM file system. This can be enforced absolutely by the system administrator by not mounting any CD-ROM file systems. It can also be observed on an individual basis if CD-ROM file systems are present on the system. Note that the only lack of POSIX conformance is in the value of the limits {`NAME_MAX`} and {`LINK_MAX`} returned by the `pathconf()` function. These limits affect application portability only when creating files. Since CD-ROM file systems are inherently read-only, file creation portability is meaningless when using them.
- Optional Trusted System software that provides Mandatory Access Control must not be used.

1.3.1.2 Documentation

The HP-UX operating system conforms to ISO/IEC 9945-1: 1990 (IEEE Std 1003.1-1990), Standard for Information Technology - Portable Operating System Interface (POSIX) - Part 1: System Application Program Interface (API) [C Language].

1.3.3 Language-Dependent Services for the C Programming Language

1.3.3.2 C Standard Language-Dependent System Support

The HP-UX operating system provides C Standard Language-Dependent System Support using ISO/IEC 9989: 1990, Information Technology - Programming Languages - C.

2

Definitions and General Requirements

2.2 Definitions

2.2.2 General Terms

Appropriate Privilege

On HP-UX systems, a process with an effective user ID of zero (which is known as the super-user's user ID) has all appropriate privileges. In addition, on a call to chown(), a process that is a member of a privileged group with the `PRIV_CHOWN` privilege (see getprivgrp(1)) and that is the owner of the specified file has the appropriate privilege to change the owner and group of the file.

Character Special File

HP-UX supports various character special files. These character special files access corresponding drivers which are identified by the major number of the special file. Many of the drivers supplied with HP-UX are described in Section 7 of the *HP-UX Reference* and also shown in the system file `/etc/master`. In addition, user-written drivers can be installed that can behave in any way as dictated by the driver program code.

File

In addition to regular, character special, block special, FIFO, and directory files, HP-UX supports symbolic links (see *symlink*(4)), network special files (see *HP-UX Reference* Section 7 entries), and sockets (see *socket*(7) in *HP-UX Reference*.

Parent Process ID

On HP-UX systems, if a child process continues to exist after its creator process ceases to exist, the *parent process ID* is set to 1 which is the *process ID* of the system initialization process.

Pathname

Multiple consecutive slashes in a pathname are treated as equivalent to a single slash. Multiple consecutive leading slashes in a pathname are treated in the same manner as multiple slashes elsewhere in the pathname.

Read-Only File System

File systems are made part of the HP-UX file hierarchy via the *mount*(1M) utility, the *mount*(2) function, or the *vfsmount*(2) function. When this is done, the file system can be designated as a read-only file system. This designation causes various functions described in POSIX and analogous HP-UX extensions that require creating, writing, updating files that reside on the file system, or otherwise modifying the file system to return an [`EROFS`] error indication. Files and directories on a read-only file system may only be read, not written to or updated. Read requests will not update file access times. Also, in the event of an un-handled signal (see *signal*(5)), a core image will not be generated by a process whose current working directory is in a read-only file system.

Some file systems can only be mounted read-only due to hardware restrictions or protection by a network server, while those file systems without such restrictions can instead be mounted as a read-write file system, permitting modification. HP-UX has a root file system which is always effectively mounted as a read-write file system.

2.3 General Concepts

Extended Security Controls

See Appropriate Privilege in Section 2.3 and File Access Permissions below.

File Access Permissions

The HP-UX operating system provides access control lists as an *alternate* file access mechanism (see *acl*(5)). HP-UX does not provide any *additional* file access mechanism.

File Times Update

HP-UX immediately updates fields that are “marked for update”.

2.4 Error Numbers

The HP-UX system establishes an address space for each process and reliably prevents the process from referencing memory outside that address space. When any POSIX function is invoked with a pointer argument that is in violation of the process's address space, it will not actually violate the memory protection. If the part of the function that would de-reference the invalid pointer is inside the HP-UX kernel, the function returns an error indication and sets *errno* to [`EFAULT`]. If the part of the function that would de-reference the invalid pointer is in a library outside the HP-UX kernel, a signal is generated for the process (see section 3.3.1.2 for details of which signal – usually `SIGSEGV`). Note that invalid pointers do not necessarily violate the process's address space, but may point to objects within the address space; the use of such invalid pointers is not detected, and can cause unpredictable results.

The maximum file size, exceeding which generates the [`EFBIG`] error from the **`write`**() function, is determined as follows. On a file system with 4K blocks the maximum file size is slightly larger than one gigabyte, while on a file system with 8K blocks the maximum size is {`SSIZE_MAX`} bytes. However, the maximum *user file size* is configurable by the kernel, and is dynamically changeable by the *ulimit*(2) function.

2.6 Environment Description

HP-UX permits any character except "NUL" or `=` in environment variable *names*, in addition to the portable filename character set. However, use of characters not in the portable filename character set is discouraged.

2.7 C Language Definitions

2.7.2 POSIX.1 Symbols

Additional feature test macros are defined for HP-UX; see *stdsyms*(5) in the *HP-UX Reference*.

The symbol `L_cuserid` is defined in <`stdio.h`> and a declaration for the function `cuserid()` is provided in <`unistd.h`>.

2.8 Numerical Limits

The following limits defined in <`limits.h`> vary among POSIX.1 implementations. On HP-UX systems, they have the following values:

Name	Value	Name	Value
`NGROUPS_MAX`	20	`SSIZE_MAX`	2147483647
`TZNAME_MAX`	19	`SCHAR_MAX`	127
`CHAR_BIT`	8	`SCHAR_MIN`	−128
`CHAR_MAX`	127	`SHRT_MAX`	32767
`CHAR_MIN`	−128	`SHRT_MIN`	−32768
`MB_LEN_MAX`	2	`UCHAR_MAX`	255
`INT_MAX`	2147483647	`UINT_MAX`	4294967295U
`INT_MIN`	−2147483647 − 1	`ULONG_MAX`	4294967295UL
`LONG_MAX`	2147483647L	`USHRT_MAX`	65535
`LONG_MIN`	−2147483647L − 1		

The following limits are not defined in `<limits.h>`, but are available at run time via the `sysconf`() or `pathconf`() functions. Although the values returned by `sysconf`() or `pathconf`() for some of these limits are constants, they may vary in future HP-UX releases.

Name	Value
`ARG_MAX`	20478 (bytes)
`CHILD_MAX`	Configurable by `MAXUPRC`[1]
`OPEN_MAX`	Configurable by `MAXFILES`[1]
`LINK_MAX`	1 for CD-ROM file systems[2]; 1000 for other file systems
`MAX_CANON`	512
`MAX_INPUT`	512
`NAME_MAX`	14 for short-filename system; 255 for long-filename system; 12 for CD-ROM file system[2]
`PATH_MAX`	1023
`PIPE_BUF`	8192
`STREAM_MAX`	Same as `OPEN_MAX`

[1] See *HP-UX System Administrator* manuals for more information about `MAXUPRC` and `MAXFILES`.

[2] Because of these limits, the CD-ROM file system does not strictly conform to POSIX. See explanation in Section 2.2.1.

2.9 Symbolic Constants

On HP-UX systems, <`unistd.h`> defines the following values:

Name	Value
`_POSIX_JOB_CONTROL`	2
`_POSIX_SAVED_IDS`	1
`_POSIX_VERSION`	199009L
`_POSIX_VDISABLE`	'\377'

The following values are not defined in <`unistd.h`>, but are available at run time via the `pathconf()` function. They vary as described:

`_POSIX_CHOWN_RESTRICTED`	1 if the privilege `PRIV_CHOWN` (see getprivgrp(1)) is not granted globally; otherwise −1
`_POSIX_NO_TRUNC`	1 for long-filename and CD-ROM file systems; −1 for short-filename file systems.

3 Process Primitives

3.1 Process Creation and Execution

3.1.1 Process Creation

3.1.1.2 Description

All process characteristics not defined by POSIX.1 are inherited across a fork except those listed in the *fork*(2) entry in the *HP-UX Reference* as differences between parent and child processes.

3.1.1.4 Errors

In HP-UX, the `fork()` function detects the [`ENOMEM`] error condition when the system has insufficient RAM or swap space to create the new process.

3.1.2 Execute a File

3.1.2.2 Description

If the environment variable `PATH` is not present, the search is conducted as if the `PATH` variable has the value of `:/bin:/usr/bin` as a default.

Process characteristics not defined by POSIX.1 are inherited across an exec as listed in the *exec*(2) entry in the *HP-UX Reference*.

3.1.2.4 Errors

In HP-UX, the `exec` functions detect the [`ENOMEM`] error condition when the system has insufficient RAM or swap space for the new process image.

If an attempt is made to execute a file that is not a regular file, the `exec`-type functions return −1 and set *errno* to [`EACCES`].

3.2 Process Termination

3.2.1 Wait for Process Termination

3.2.1.2 Description

`Wait()` or `waitpid()` also reports status when a process that is being traced stops because that traced process has hit a break point (see *ptrace*(2)).

If a parent process terminates without waiting for its child processes to terminate, the parent process ID of each child process is set to `1`. This means the initialization process inherits the child processes.

3.2.2 Terminate a Process

3.2.2.2 Description

When a process terminates without waiting for its children, the parent process ID of all of the calling process's existing child processes is set to `1`. This means the initialization process inherits each of these processes.

3.3 Signals

3.3.1 Signal Concepts

3.3.1.1 Signal Names

The following additional signals are defined (see *signal*(5)):

`SIGBUS`	`SIGLOST`	`SIGSYS`	`SIGVTALRM`
`SIGEMT`	`SIGPROF`	`SIGTRAP`	`SIGWINDOW`
`SIGIO`	`SIGPWR`	`SIGURG`	

3.3.1.2 Signal Generation and Delivery

If a subsequent occurrence of a pending signal is generated, the HP-UX operating system delivers the signal only once.

HP-UX can generate the following signals, in addition to those specified by POSIX.1, for a Conforming POSIX Application that is correctly written, without the invocation of any extended features.

Signal	Circumstances
SIGKILL	Error in one of the `exec` functions after the old process image has been discarded.
	I/O error encountered in paging or swapping System runs out of swap space while swapping out the process.
	A parity error or double-bit memory error is encountered on a series 300/400.
SIGHUP	A member of an orphaned process group would be stopped by a SIGTSTP, SIGTTIN, or SIGTTOU signal (the stop signal is discarded as required by POSIX)
SIGPWR	Series 800 only: sent to all processes upon recovery from a power failure.
SIGSEGV	A process runs out of stack space while attempting to grow the user stack.
SIGILL	Series 700/800: A process cannot write to the dedicated signal stack while attempting to deliver a signal.

Certain programming errors can cause the following signals to be generated as well:

Signal	Circumstances
SIGSYS	An illegal value for the *whence* argument was passed to the `lseek()` function.
SIGBUS	Series 300/400: An attempt was made to write to a read-only memory location, or to execute instructions from an odd address.
	Series 700/800: An attempt was made to access an unaligned address or a memory location to which the process has incorrect access right.
SIGIOT	Series 300/400: A "line E Emulator" instruction is executed. (SIGIOT is an alias for SIGABRT)
SIGEMT	Series 300/400: A "line F Emulator" instruction is executed.

HP-UX can generate the following signals for a Conforming POSIX Application if the specified extended features are explicitly invoked either directly by the application or by another application directed at the conforming application.

Signal	Extended Feature
SIGTRAP	*ptrace*(2)
SIGIO	*ioctl*(5)
SIGURG	*socket*(7)
SIGPROF	*getitimer*(2)
SIGVTALRM	*getitimer*(2)
any signal	*pty*(7)

In addition, there are other signals that can be generated only if an application explicitly invokes extended features, or if it is run in a non-conforming environment. These are not listed, since they do not impact portability.

3.3.3 Manipulate Signal Sets

3.3.3.4 Errors

In HP-UX, the `sigaddset()`, `sigdelset()`, and `sigismember()` functions detect the [`EINVAL`] error condition when the signal number specified is less than 1 or greater than 256. There are values between 1 and 256 which are invalid but for which the error is not detected.

3.3.6 Examine Pending Signals

3.3.6.4 Errors

The `sigpending()` function returns [`EFAULT`] if the argument set points to an invalid address.

4

Process Environment

4.2 User Identification

4.2.4 Get User Name

4.2.4.4 Errors

The following error conditions are detected for `getlogin()`:

[EBADF]	An invalid file descriptor was obtained.
[EMFILE]	Too many file descriptors are in use by the process.
[ENFILE]	The system file table is full.

4.4 System Identification

4.4.1 System Name

4.4.1.2 Description

The format of each member of structure `utsname` is:

Member	Contents
`sysname`	"HP-UX"
`nodename`	an arbitrary string up to eight characters set by the system administrator (see uname(1))
`release`	Series 300/400: A string of the form *x*.*y* where *x* is the major release number and *y* is the minor release number. Series 700/800: A string of the form `A.B`*x*.*yz* where *x* is the major release number, *y* is the one-digit minor release number, and *z* is the one-digit media release number.
`version`	`A` on a single-user system `B` on a 16-user system `C` on a 32-user system `D` on a 64-user system `U` on a system supporting an unlimited number of users
`machine`	`9000/`*nnn* where *nnn* is a three-digit hardware model number

4.4.1.4 Errors

[`EFAULT`] is returned if the argument *name* points to an illegal address.

4.5 Time

4.5.1 Get System Time

4.5.1.4 Errors

The `time()` function returns [`EFAULT`] if the argument *tloc* points to an illegal address.

4.5.2 Process Times

4.5.2.4 Errors

The `times()` function returns [`EFAULT`] if the argument *buffer* points to an illegal address.

4.6 Environment Variables

4.6.1 Environment Access

4.6.1.4 Errors

No error conditions are detected for `getenv()`.

4.7 Terminal Identification

4.7.1 General Terminal Pathname

4.7.1.4 Errors

No error conditions are detected for `ctermid()`.

4.7.2 Determine Terminal Device Name

4.7.2.4 Errors

The following error conditions are detected for `ttyname()` and `isatty()`:

[EBADF]	The *fildes* argument is invalid.
[ENOTTY]	The *fildes* argument specifies a file that is not a terminal device.

Files and Directories

5.1 Files and Directories

5.1.1 Format of Directory Entries

When a directory is created, the link count of its parent directory is incremented to reflect the reference of the `dot-dot` entry in the newly created directory to its parent directory.

5.1.2 Directory Operations

5.1.2.4 Errors

In HP-UX, the `opendir()` function detects the [`EMFILE`] and [`ENFILE`] error conditions as described in POSIX.1.

The `readdir()` function detects the [`EBADF`] error condition when the *dirp* argument is a NULL pointer or when it points to a valid data structure that contains an invalid file descriptor.

The `closedir()` function detects the [`EBADF`] error condition when the *dirp* argument is a NULL pointer or when it points to a valid data structure that has been previously closed or that contains an invalid file descriptor.

5.2 Working Directory

5.2.2 Working Directory Pathname

5.2.2.4 Errors

In HP-UX, as installed, the `getcwd()` function detects the [`EACCES`] error condition for all cases described in POSIX.1. If the system administrator modifies the permissions of the file `/bin/pwd` so that its set-user-ID and/or set-group-ID bit is set, the `getcwd()` function will only detect the [`EACCES`] error condition when both the calling process and a process with the effective user ID and effective group ID resulting from executing `/bin/pwd` would be denied the necessary permissions.

5.3 General File Creation

5.3.1 Open a File

5.3.1.2 Description

On HP-UX systems, when the value of *oflag* is `O_CREAT`, bits in *mode* other than the file permission bits are set, the set-user-ID and set-group-ID bits will be passed on to the file mode and all other bits are silently ignored.

The group ID of a newly created file is determined by the set-group-ID bit of the parent directory. If its parent directory's set-group-ID bit is set, the new file's group ID is set to that of the parent directory. Otherwise, the new file's group ID is set to the effective group ID of the process that calls `open`().

If `O_EXCL` is set when `O_CREAT` is not set, `O_EXCL` is silently ignored.

If the `O_TRUNC` flag is set and the file is a regular file or a network special file, the file is truncated if the process has write permission. For other file types, the flag has no effect.

5.3.3 Set File Creation Mask

5.3.3.2 Description

On HP-UX systems, all bits in *cmask* other than the file permission bits are ignored.

5.3.4 Link to a File

5.3.4.2 Description

HP-UX permits a process with user ID equal to 0 to use `link()` on directories, but no other user has this privilege.

5.4 Special File Creation

5.4.1 Make a Directory

5.4.1.2 Description

If bits in the *mode* argument to the `mkdir()` function other than the file permission bits are set, only the `S_ISUID`, `S_ISGID`, and "save-text" bits (see *chmod*(2)) are relevant to the mode of the newly created directory; any of these that are set are set in the mode of the newly created directory. Other bits are silently ignored.

The group ID and set-group-ID bits of a newly created directory are determined by the set-group-ID bit of the parent directory. If its parent directory's set-group-ID bit is set, the new directory's group ID is set to that of the parent directory, and the set-group-ID bit of the new directory is set. Otherwise, the new directory's group ID is set to the effective group ID of the process that calls `mkdir()`, and the set-group-ID bit of the new directory is determined from the *mode* argument.

5.4.2 Make a FIFO Special File

5.4.2.2 Description

If any bits in the *mode* argument to the `mkfifo()` function other than the file permission bits are set, they are silently ignored.

The group ID of a newly created FIFO is determined by the set-group-ID bit of the parent directory. If its parent directory's set-group-ID bit is set, the new FIFO's group ID is set to that of the parent directory. Otherwise, the new FIFO's group ID is set to the effective group ID of the process that calls `mkfifo`().

5.5 File Removal

5.5.1 Remove Directory Entries

5.5.1.2 Description

HP-UX permits a process with user ID equal to 0 to use `unlink()` to remove directories, but no other user has this privilege.

5.5.2 Remove a Directory

5.5.2.2 Description

If the directory indicated in a call to `rmdir()` is the root directory for the current process or for the system, `rmdir()` returns `-1` and sets *errno* to [`EISDIR`].

If the directory is the current working directory for the calling process, `-1` is returned and *errno* is set to [`EINVAL`].

If the directory is the current working directory or root directory for another process, no error is detected and, if permissions allow, the directory is removed.

5.6 File Characteristics

5.6.1 File Characteristics: Header and Data Structure

5.6.1.2 `<sys/stat.h>` File Modes

If the `S_ISUID`, `S_ISGID`, or `S_ISVTX` bits are set for a file, the value of the st_mode field of the stat structure returned by `stat`() and `fstat`() includes these bits.

`S_IRWXU`, `S_IRWXG`, and `S_IRWXO` include only the bits required in this section.

5.6.1.3 `<sys/stat.h>` Time Entries

The `close()` function modifies the *st_mtime* and *st_ctime* fields of a FIFO if the file contains unread data at the time of the close.

Other HP-UX functions that are not defined by POSIX also modify the *st_atime* field if they access file data, the *st_mtime* field if they modify file data, or the *st_ctime* field if they change file status.

5.6.2 Get File Status

5.6.2.2 Description

There are no conditions under which additional or alternate file access controls will cause `stat()` or `fstat()` to fail, except for the effect of an access control list on the ordinary file access permission mechanism (see *acl*(5)).

5.6.4 Change File Modes

5.6.4.2 Description

`chmod()` has no effect on open file descriptors.

5.6.5 Change Owner and Group of a File

5.6.5.2 Description

If `chown()` is called by the super-user and the *path* argument is a regular file, the set-user-ID and set-group-ID bits are not cleared.

5.7 Configurable Pathname Variables

5.7.1 Get Configurable Pathname Variables

5.7.1.4 Errors

In HP-UX, the `pathconf()` function detects the [`EACCES`], [`EINVAL`], [`ENAMETOOLONG`], [`ENOENT`], and [`ENOTDIR`] error conditions for all cases described in POSIX.1.

The `fpathconf()` function detects the [`EBADF`] and [`EINVAL`] error conditions for all cases described in POSIX.1.

6

Input and Output Primitives

6.4 Input and Output

6.4.1 Read from a File

6.4.1.2 Description

On HP-UX systems, if a `read()` is interrupted by a signal after it has successfully read some data, it returns the number of bytes read.

Behavior of device special file read requests after end-of-file varies from device to device and is documented in section 7 of the *HP-UX Reference*. For terminal devices, this behavior conforms with Section 7.1.1 of POSIX.

If *nbyte* is greater than {`SSIZE_MAX`}, −1 is returned and `errno` is set to [`EINVAL`].

6.4.1.4 Errors

HP-UX systems return an [`EIO`] error when a device-specific hardware error is detected during a `read()` request.

6.4.2 Write to a File

6.4.2.2 Description

On HP-UX systems, if a `write()` is interrupted by a signal after it has successfully written some data, it returns the number of bytes written.

If *nbyte* is zero and the file is a character special file, the result is driver dependent, and any special behavior is documented in section 7 of the *HP-UX Reference*. For other types of files, `write()` immediately returns 0.

If *nbyte* is greater than {`SSIZE_MAX`}, −1 is returned and `errno` is set to [`EINVAL`].

6.4.2.4 Errors

HP-UX systems return an [`EIO`] error when a device-specific hardware error is detected during a `write()` request.

6.5 Control Operations on Files

6.5.2 File Control

6.5.2.2 Description

`F_SETFL`:

In addition to the file status flags listed in Table 6-5, HP-UX also supports the flags `O_SYNC` and `O_NDELAY` (see *open*(2)). Any other bits set in the third argument, *arg*, are ignored.

`F_GETLK`, `F_SETLK`, `F_SETLKW`:

If the *l_len* is negative, the `fcntl()` function returns −1 and sets *errno* to [`EINVAL`].

6.5.2.4 Errors

In the HP-UX operating system, the `fcntl()` function detects the [`EDEADLCK`] error condition when *cmd* is `F_SETLKW` and waiting for the requested lock, in combination with existing locks, would create a deadlock condition.

6.5.3 Reposition Read/Write File Offset

6.5.3.2 Description

On devices incapable of seeking, `lseek` always succeeds but has no effect.

7 Device- and Class-Specific Functions

7.1 General Terminal Interface

The HP-UX general terminal interface supports asynchronous terminals. It also supports network connections such as *rlogin*(1) and *telnet*(1) through the pseudo terminal interface (see *pty*(7)) as well as other applications that use the pseudo terminal interface such as *shl*(1) and terminal emulators running in windowed environments.

The Internal Terminal Emulator (ITE) uses the general terminal interface to control a bit-mapped display and keyboard as the system console.

Synchronous ports are not supported.

7.1.1 Interface Characteristics

7.1.1.3 The Controlling Terminal

If a session leader has no controlling terminal and opens a terminal device file that is not already associated with a session without using the `O_NOCTTY` option (see *open*(2)), the terminal becomes the controlling terminal of the session and the controlling terminal's foreground process group is set to the process group of the session leader. This is the only way to allocate a controlling terminal in HP-UX.

7.1.1.5 Input Processing and Reading Data

When the input limit {`MAX_INPUT`} is exceeded in the input queue, all saved characters may be discarded without notice, although they may be retained.

7.1.1.6 Canonical Mode Input Processing

When the {`MAX_CANON`} limit is reached in the input queue, all characters in the current undelimited line are discarded without notice.

7.1.1.7 Non-Canonical Mode Input Processing

In the HP-UX operating system, the `MIN` member of the *c_cc* array can never be greater than {`MAX_INPUT`} because {`MAX_INPUT`} is too large to be represented as type `cc_t`.

7.1.1.8 Writing Data and Output Processing

The terminal interface provides a buffering mechanism. When characters are written, they are placed on the output queue. Characters on the output queue are transmitted to the terminal as soon as previously-written characters are sent.

7.1.1.9 Special Characters

The NL, CR, START, and STOP characters cannot be changed or disabled.

One additional special character is provided in the `termios` structure:

SWTCH	A special character on input used only by the shell layers facility *shl*(1). The shell layers facility is not part of the general terminal interface. No special functions are performed by the general terminal interface when SWTCH characters are encountered.

Another special character, `t_dsuspc`, is recognized by the general terminal interface, but is not included in the `<termios.h>` structure. The `t_dsuspc` character functions the same as `t_suspc` (that is, SUSP), except that the suspend signal (`SIGTSTP`) is sent when a process reads the character, rather than when it is typed. This character is read and set by the `ioctl()` function with the commands `TIOCGLTC` and `TIOCSLTC` (see *termio*(7)). By default, this special character function is disabled. It is enabled only by explicit action.

7.1.2 Parameters That Can Be Set

7.1.2.2 Input Modes

Break conditions can also be generated in HP-UX contexts other than asynchronous serial data transmission. These are generated by using `tcsendbreak()` on the slave side of pseudoterminals, and by the Internal Terminal Emulator (ITE) in HP-UX workstations when the user hits the [BREAK] key.

If `IXOFF` is set, start/stop input control is enabled. The system transmits a STOP character when the number of characters in the input queue exceeds a hardware-dependent value (high water mark). This is intended to cause the terminal device to stop transmitting data, as needed to prevent the number of characters in the input queue from exceeding `MAX_INPUT`. When enough characters have been read from the input queue that the number of remaining characters is less than another hardware-dependent value (low water mark), the system transmits a START character which is intended to cause the terminal device to resume transmitting data.

The initial input control value after `open()` is all bits clear.

7.1.2.3 Output Modes

The initial output value after `open()` is all bits clear.

If `OPOST` is set, output characters are post-processed as indicated by the remaining flags in a fashion so that lines of text are modified to appear appropriately on the terminal device; otherwise characters are transmitted without change. These flags provide for special handling of carriage returns, new-lines, fill characters, and delays. See *termio*(7) for more details.

7.1.2.4 Control Modes

On HP-UX systems, the initial hardware control value after `open()` is all bits clear.

7.1.2.5 Local Modes

If `ICANON`, `ECHO`, and `ECHOE` are set, the erase character is echoed as the three-character ASCII sequence BS-SP-BS, which clears the last character from a CRT screen. If ECHOE is set and ECHO is clear, the erase character is echoed as the two-character ASCII sequence SP-BS, which clears the current character from a CRT screen (the cursor remains in the same position). No determination is made as to whether an erasable character exists.

If ICANON, ECHO, and ECHOK are set, a new line is sent to the terminal after the kill character. If ICANON and ECHOK are set but ECHO is not set, the kill character is echoed as a new line.

If IEXTEN is set, the IXANY, XCASE, and IUCLC functions are allowed (see *termio*(7)). IEXTEN does not affect or interact with any other functions.

The initial local control value *after open* is all bits clear.

7.1.2.6 Special Control Characters

In the `termios` structure, the `c_cc` array contains 16 elements.

Attempts to change the START and STOP characters are ignored.

The initial values of the special control characters are as follows:

EOF	Control-D
EOL	NUL
ERASE	#
INTR	DEL
KILL	@
QUIT	Control-|
START	Control-Q
STOP	Control-S
SUSP	disabled
SWTCH	NUL

The following special control characters are interpreted as numeric values, not as characters:

MIN	0
TIME	4

7.1.3 Baud Rate Functions

7.1.3.2 Description

`cfsetispeed()` and `cfsetospeed()` can be used to set speed codes that are defined in the `termios` structure whether or not those speeds are supported by the terminal hardware. No errors are detected by these functions.

7.2 General Terminal Interface Control Functions

7.2.2 Line Control Functions

7.2.2.2 Description

If the *duration* parameter to the `tcsendbreak()` function is not zero, `tcsendbreak()` sends zero-valued bits for 0.25 second.

If a terminal is not using asynchronous serial data transmission, the `tcsendbreak()` function returns without taking any action unless the application is using a pty, in which case the process that has the master side of the pty open may be notified and may take action (see *pty*(7)).

8 Language-Specific Services for the C Programming Language

HP-UX systems conform to C Language Binding (C Standard Language-Dependent System Support).

The behavior of all library functions conforms to the *American National Standard for Information Systems: Programming Language C* ANS X3.159-1989.

8.1 Referenced C Language Routines

8.1.1 Extensions to Time Functions

A leading colon is ignored and the remainder of the `TZ` string is processed as if the colon was not present.

8.1.2 Extensions to `setlocale`() Function

8.1.2.2 Description

HP-UX supports an additional `LC_MESSAGES` locale category, as defined in IEEE Std 1003.2-1993.

There are three cases where POSIX.1 leaves `setlocale()` behavior implementation-defined:

- For the call:

  `setlocale(`*category*`, "")`

  where *category* is one of `LC_CTYPE`, `LC_COLLATE`, `LC_TIME`, `LC_NUMERIC`, `LC_MONETARY`, or `LC_MESSAGES`, and when none of: the environment variable corresponding to *category*, the `LANG` environment variable, or the `LC_ALL` environment variable are set to a non-empty string, the implementation-defined behavior for HP-UX is to set the specified category to the `C` locale.

- For the call:

  ```
  setlocale(LC_ALL, "")
  ```

  when none of: the environment variable corresponding to a category (`LC_CTYPE`, `LC_COLLATE`, `LC_TIME`, `LC_NUMERIC`, `LC_MONETARY`, or `LC_MESSAGES`), the `LANG` environment variable, or the `LC_ALL` environment variable is set to a non-empty string, the implementation-defined behavior for HP-UX is to set that category to the `C` locale.

- When *locale* is a specific string, it must match one of the forms described for the `LC_*` environment variables given on *environ*(5) in the *HP-UX Reference*. The various possibilities for the language name components of locale strings is configuration-dependent, but the available names are given on *lang*(5). The possibilities for the other components are language- and configuration-dependent. The complete list of configured locale names and other locale string components can be determined using the *locale*(1) utility.

  The locale strings `C` and `POSIX` are always supported.

8.2 C Language Input/Output Functions

8.2.2 Open a Stream on a File Descriptor

8.2.2.2 Description

The following additional *type* arguments to the `fdopen()` function are supported:

`rb`	open binary file for reading
`wb`	open or create binary file for writing
`ab`	append; open binary file for writing at end-of-file, or create binary file

`r+b` or `rb+`	open binary file for update (reading and writing)
`w+b` or `wb+`	open or create binary file for update
`a+b` or `ab+`	append; open or create binary file for update at end-of-file

8.2.3 Interactions of Other *FILE*-Type C Functions

When all the rules in 8.2.3 are followed, all input will be seen exactly once in at least the following situations (assuming no errors occur):

There is no input data.

No buffered stream handles are opened for the file, and all input is read by one or more unbuffered stream handles and/or file descriptor handles.

Exactly one buffered stream handle is opened for the file, and all input is read via that handle.

No read via any particular buffered stream handle is intermixed in time with any reads or writes from any other handle, at the last read on that handle consumes all the remaining buffered input, and the last read for any handle consumes all of the remaining input. That is, all reads between open and close of a given handle must occur as a group, not mixed with reads from any other open handle, all buffered input is consumed before the next open handle is allowed to read, and the last reader sees all the remaining input.

All reads via buffered stream handles are of a size that is a common multiple of all the stream buffer sizes (so that there is never any unused buffered input), and all data is consumed.

8.3 Other C Language Functions

8.3.2 Set Time Zone

8.3.2.2 Description

On HP-UX systems, if the environment variable `TZ` is absent, behavior is the same as when `TZ` is set to `EST5EDT`.

9 System Databases

9.1 System Databases

If the initial working directory field is null, that field is interpreted by the `login` utility (see *login*(1)) as meaning the root directory for super-user. For other users, if the field is null, the user cannot log in.

If the initial user program field is null, that field is interpreted by the `login` utility (see *login*(1)) as meaning the POSIX shell, `/bin/sh`.

9.2 Database Access

9.2.1 Group Database Access

9.2.1.4 Errors

The `getgrgid()` and `getgrnam()` functions detect only those errors that can be detected by opening, reading, or closing a file.

9.2.2 User Database Access

9.2.2.4 Errors

The `getpwuid()` and `getpwnam()` functions detect only those errors that can be detected by opening, reading, or closing a file.

10

Data Interchange Format

10.1 Archive/Interchange File Format

See the *cpio*(1) and *tar*(1) entries in the *HP-UX Reference* for a description of how the `tar` and `cpio` utilities translate from the file system to the formats defined in Chapter 10 of POSIX.

10.1.1 Extended `tar` Format

If a file name is found on the medium that would create an invalid file name on the system, the data from the file is not stored on the file hierarchy. The `tar` utility ignores such files and produces an error indicating that the file is being ignored.

The `TSVTX` mode bit is not mentioned in POSIX. It corresponds to the HP-UX "save text" bit (see *chmod*(2)), and can only be set by a user with appropriate privilege.

If the *typeflag* field is set to `CHARTYPE` or `BLKTYPE`, the *size* field contains the *cnodeid* value from `st_rcnode` in the <`stat.h`> structure (see *stat*(5)). If the *typeflag* field is set to `FIFOTYPE`, the *size* field is set to zero when creating an archive and is ignored when reading an archive.

For character special and block special files (represented by the ASCII digits `3` and `4`), for `CHARTYPE` and `BLKTYPE` records, the *devmajor* field contains the octal digits representing the upper eight bits of the `st_rdev` field returned by `stat()`, and *devminor* contains the octal digits representing the lower 24 bits of the `st_rdev` field returned by `stat()`.

10.1.2 Extended `cpio` Format

10.1.2.1 `cpio` Header

The values of the `c_dev` and `c_ino` fields are generated arbitrarily by the `cpio` utility and carry no meaning other than the fact that entries with the same pair of values are links to the same file.

For character or block special files, `c_rdev` contains the `st_rcnode` field of the <`stat.h`> structure (see *stat*(5)) which contains the *cnodeid* of that system in the cluster where the cnode-specific device file can be used (see `mkrnod()` description in *mknod*(2)).

10.1.2.2 `cpio` File Name

If a file name is found on the medium that would create an invalid file name on the system, the data from the file is not stored on the file hierarchy. The `cpio` utility ignores such files and produces an error indicating that the file is being ignored.

10.1.2.4 `cpio` Special Entries

For special files other than FIFO special files, directories, and the trailer, *c_filesize* contains the major and minor number of the field corresponding to `st_rdev` in the <`stat.h`> structure (see *stat*(5) and *mknod*(2)).

10.3 Multiple Volumes

The HP-UX utilities `tar` and `cpio` determine what file to read as the next file at the end of a volume by prompting the user at the controlling terminal and reading the user response from the controlling terminal keyboard. The user response is expected to consist of the next file's pathname.

Part II
POSIX.2-1992

Contents

1

General

1.3 Conformance

1.3.1 Implementation Conformance

The HP-UX operating system conforms to IEEE Std 1003.2-1992 (POSIX.2) and the following options:

- Software Development Utilities Option
- C-Language Bindings Option
- C-Language Development Utilities Option
- FORTRAN Development Utilities Option
- FORTRAN Runtime Utilities Option

The HP-UX operating system supports many facilities beyond POSIX. A few of these facilities can set up an environment in which a Conforming POSIX Application would not always see behavior conforming to the POSIX standard. The following guidelines can be observed to ensure this does not occur. Note that it takes explicit action on the part of the system administrator, and possibly on the part of the user as well, to violate these guidelines, and thus they should only be of concern when use of the HP-UX facilities mentioned is desired.

- The HP-UX kernel must be configured with the value of `maxuprc` greater than or equal to the value of `{_POSIX_CHILD_MAX}` defined in POSIX.1. See the *System Administrator Tasks* and *System Administrator Concepts* manuals for more details.

- A process requiring POSIX conformance should not have its root directory modified (see *chroot*(1) or *chroot*(2) in the *HP-UX Reference* for more details). It may be possible to create a file hierarchy with a different root directory that does result in full POSIX conformance, but the information and support to do this are not provided.

- A process requiring POSIX conformance must not have an effective group ID or supplementary group ID with the privilege SETRUGID. See *setprivgrp*(1m) and *setuid*(2) in the *HP-UX Reference* for more details.
- An application requiring POSIX conformance must not access any files over a network by use of NFS(TM) network services. This can be enforced absolutely by the system administrator by not setting up NFS or by not mounting any NFS file systems. It can also be observed on an individual basis if NFS is being used. See the NFS networking manuals provided with your system for more details.
- Any terminal being used in a way requiring POSIX conformance must not have a delayed suspend character defined. See *termio*(7) in the *HP-UX Reference* for more details.
- An application requiring POSIX conformance must not access any files on a CD-ROM file system. This can be enforced absolutely by the system administrator by not mounting any CD-ROM file systems. It can also be observed on an individual basis if CD-ROM file systems are present on the system. Note that the only lack of POSIX conformance is in the value of the limits `{NAME_MAX}` and `{LINK_MAX}` returned by the pathconf() function. These limits affect application portability only when creating files. Since CD-ROM file systems are inherently read-only, file creation portability is meaningless when using them.
- Optional Trusted System software that provides Mandatory Access Control must not be used.

2

Terminology and General Requirements

2.2 Definitions

2.2.2 General Terms

2.2.2.6 Appropriate Privileges

On HP-UX systems, a process with an effective user ID of zero (which is known as the super-user's user ID) has all appropriate privileges. In addition, on a call to chown(2), a process that is a member of a privileged group with the `PRIV_CHOWN` privilege (see *getprivgrp*(1)) and that is the owner of the specified file has the appropriate privilege to change the owner and group of the file.

2.2.2.22 Byte

On HP-UX systems, a byte is composed of 8 bits.

2.2.2.51 Extended Security Controls

See Appropriate Privileges above, and File Access Permissions below.

2.2.2.54 File

In addition to regular, character special, block special, FIFO and directory files, HP-UX supports symbolic links (see *symlink*(4)), network special files (see *HP-UX Reference* Section 7 networking entries), and sockets (see *socket*(7)).

2.2.2.55 File Access Permissions

The HP-UX operating system provides access control lists as an alternate file access mechanism (see *acl*(5)). It does not provide any additional file access mechanism.

2.2.2.57 File Group Class

The HP-UX operating system does not define any additional members of the *file group class* of a *file* other than those defined in POSIX.

2.2.2.69 File Times Update

The HP-UX operating system immediately updates fields that are "marked for update".

2.2.2.101 Parent Process ID

On HP-UX systems, if a child process continues to exist after its creator process ceases to exist, the *parent process ID* is set to 1 which is the *process ID* of the system initialization process.

2.2.2.102 Pathname

Multiple consecutive slashes in a pathname are treated as equivalent to a single slash. Multiple consecutive leading slashes in a pathname are treated in the same manner as multiple slashes elsewhere in the pathname.

2.2.2.120 Read-Only File System

File systems are made part of the HP-UX file hierarchy via the *mount*(1M) utility, the *mount*(2) function, or the *vfsmount*(2) function. When this is done, the file system can be designated as a read-only file system. This designation causes various functions described in POSIX and analogous HP-UX extensions that require creating, writing, updating files that reside on the file system, or otherwise modifying the file system to return an [`EROFS`] error indication. Files and directories on a read-only file system can only be read, not written to or updated. Read requests do not update file access times. Also, in the event of an un-handled signal (see *signal*(5)), a core image is not generated by a process whose current working directory is in a read-only file system.

Some file systems can only be mounted read-only due to hardware restrictions or protection by a network server, while those file systems without such restrictions can instead be mounted as a read-write file system, permitting modification. The HP-UX operating system has a root file system which is always effectively mounted as a read-write file system.

The HP-UX operating system does not treat a dollar-sign character as an anchor when used as the last character of a subexpression.

2.4 Character Set

The HP-UX operating system does not support the use of either locking-shift or single-shift state-dependent character encodings with any of the standard utilities or C-language functions.

2.5 Locale

Users cannot create locales other than `POSIX` and `C` via the *localedef*(1) utility.

When the value of a locale environment variable contains a slash (`/`), it is interpreted as the pathname of the locale definition file, which is a binary file. By default, such files are stored in the `/usr/lib/nls/`*locale_name* directory, where *locale_name* is the value of the corresponding locale environment variable.

2.5.2 Locale Definition

The HP-UX system allows the following additional category to be present in a locale definition file: `LC_ALL`.

The following is a list of additional keywords recognized by *localedef*(1):

Keyword	Category
`langname`	Global Statement
`langid`	Global Statement
`revision`	Global Statement
`modifier`	All categories
`first`	`LC_CTYPE`
`second`	`LC_CTYPE`
`bytes_char`	`LC_CTYPE`
`alt_punct`	`LC_CTYPE`
`code_scheme`	`LC_CTYPE`
`cswidth`	`LC_CTYPE`
`crncystr`	`LC_MONETARY`
`alt_digit`	`LC_NUMERIC`
`year_unit`	`LC_TIME`
`mon_unit`	`LC_TIME`
`day_unit`	`LC_TIME`
`hour_unit`	`LC_TIME`
`min_unit`	`LC_TIME`
`sec_unit`	`LC_TIME`
`context`	`LC_ALL`
`direction`	`LC_ALL`

The value of a character, when represented by itself, is based on one of the following character encoding schemes, depending on the environment locale setting when *localedef*(1) is invoked: US ASCII, Roman8, ISO 8859/1, HP15 or EUC.

2.5.2.1 LC_CTYPE

If no charmap file is given, missing characters which are automatically included have one of the following character encoding schemes, depending on the environment locale setting when *localedef*(1) is invoked: US ASCII, Roman8, ISO 8859/1, HP15 or EUC.

2.5.2.5 LC_TIME

The following optional keywords are recognized: `era` and `era_d_fmt`.

2.5.3 Locale Definition Grammar

Grammars for additional categories and keywords are documented in *localedef*(4).

2.6 Environment Variables

The `LANG` environment variable is also used to locate message catalogs opened by *catopen*(3C).

The syntax for the following environment variables: `LANG`, `LC_ALL`, `LC_COLLATE`, `LC_CTYPE`, `LC_MONETARY`, `LC_NUMERIC`, and `LC_TIME` is:

[*language*[_*territory*] [.*codeset*] [@*modifier*]]

See *environ*(4) for more information.

If neither `LC_ALL` or `LANG` is set, the default locale of `C` is used.

`LANG` and any of the environment variables that begin with `LC_` can be set to any of the locale names found in the `/usr/lib/nls/config file. Valid locales are located in the /usr/lib/nls directory` and must also appear in the `/usr/lib/nls/config file. If the PATH` environment variable is unset or is set to null, the path search is limited to the current working directory. See *pwd*(1).

2.8 Regular Expression Notation

2.8.3 Basic Regular Expressions

2.8.3.5 BRE Expression Anchoring

The HP-UX operating system does not treat a circumflex as an anchor when used as the first character of a subexpression.

2.9 Dependencies on Other Standards

2.9.1 Features Inherited from POSIX.1

2.9.1.5 File Removal

If the directory indicated is the root directory for the system, it cannot be removed.

If the directory indicated is the root or current working directory for the current process, it cannot be removed.

If the directory indicated is the root or current working directory for another process, it can be removed if permissions allow.

2.13 Configuration Values

2.13.2 Symbolic Constants for Portability Specifications

Name	Value
`POSIX2_C_BIND`	Defined
`POSIX2_C_DEV`	Defined if `/bin/c89` exists and is executable, `/usr/bin/lex` exists and is executable, `/usr/bin/yacc` exists and is executable and `/lib/libc.a` exists and is readable.
`POSIX2_FORT_DEV`	Defined if `/usr/bin/fort77` exists and is executable and `/usr/bin/asa` exists and is executable.
`POSIX2_FORT_RUN`	Defined if /usr/bin/asa exists and is executable.
`POSIX2_LOCALEDEF`	Defined
`POSIX2_SW_DEV`	Defined if `/bin/ar` exists and is executable, `/bin/make` exists and is executable and `/bin/strip` exists and is executable.

3 Shell Command Language

3.7 Redirection

The largest file descriptor number is `{OPEN_MAX}`−1. `{OPEN_MAX}` can be configured using the `setrlimit()` function.

3.14 Special Built-in Utilities

3.14.13 trap – Trap signals

The implementation allows numeric signal numbers for the conditions as an extension.

4

Execution Environment Utilities

4.1 awk – Pattern scanning and processing language

4.1.7 Extended Description

4.1.7.3 Variables and Special Variables

Any modification of `ENVIRON` after the initial execution of awk does not affect the process' environment.

The default value of `SUBSEP`, the subscript separator string, is ASCII FS (field separator – octal 034).

4.1.7.6 Actions

4.1.7.6.2 Functions

4.1.7.6.2.3 Input/Output and General Functions

The total number of open *expression* arguments is limited to 40.

4.2 basename – Return nondirectory portion of pathname

4.2.2 Description

Step (1). If the string is "`//`", Steps (2) through (5) are processed.

4.5 cd – Change working directory

4.5.2 Description

If HOME is empty or undefined, invoking `cd` without operands causes an error message to be displayed and the current directory remains unchanged.

4.5.4 Operands

If directory is `-`, the directory is changed to the previous working directory.

4.7 chmod – Change file modes

4.7.2 Description

Access Control Lists are supported as an alternate file access mechanism. See *acl*(5).

4.7.7 Extended Description

When using the symbolic mode form on a regular file:

- Requests to set the set-user-ID-on-execution or set-group-ID-on-execution bit when all execute bits are currently clear and none are being set are honored.
- Requests to clear all execute bits do not clear the set-user-ID-on-execution and set-group-ID-on-execution bits.

- Requests to clear the set-user-ID-on-execution or set-group-ID-on-execution bits when all execute bits are currently clear are honored.

When using the symbolic mode form on file types other than regular files, requests to set or clear the set-user-ID-on-execution or set-group-ID-on-execution bits are honored.

For each bit set in the octal number form, the corresponding file permission bit is set and all other file permission bits are cleared. For file types other than regular files, if the set-user-ID-on-execution or the set-group-ID-on-execution bits are not set in the octal number, those attributes are cleared.

4.13 cp – Copy Files

4.13.2 Description

Step (2)(c). The options `-r` and `-R` produce the same behavior. FIFOs are not replicated, and device special files are replicated only when the invoking process is the super-user. The contents of symbolic links are replicated. Hard links within the copy hierarchy are preserved.

Step (4)(a). The options `-r` and `-R` produce the same behavior.

Step (4)(b)[2]. The file permission bits are the same as those of *source_file*, modified by the file creation mask of the user.

Access Control Lists are supported as an alternate file access mechanism. See *acl*(5) for information on the handling of optional Access Control Lists.

4.13.3 Description

Access Control Lists are supported as an alternate file access mechanism. See *acl*(5).

The options `-r` and `-R` produce the same behavior. FIFOs and device special files are replicated. The contents of symbolic links are replicated. Hard links within the copy hierarchy are preserved.

4.16 dd – Convert and copy a file

4.16.6 External Effects

4.16.6.2 Standard Error

Diagnostic messages are written to the standard error.

4.18 dirname – Return directory portion of pathname

4.18.2 Description

Step (6). If the remaining string is "`//`", steps (7) and (8) are processed.

4.19 echo – Write arguments to standard output

4.19.4 Operands

`echo` treats the string "`-n`" as one of the arguments to be printed.

The backslash character (`\`) introduces C language-like escape conventions:

`\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.

4.24 find – Find files

4.24.4 Operands

If a *utility_name* or *argument* string contains the two characters `{}`, but not just the two characters `{}`, `find` uses the string without change.

4.33 ln – Link files

4.33.4 Operands

On HP_UX, a directory can only be linked when the `-s` (symbolic link) option is used. See *ln*(1).

4.34 locale – Get locale-specific information

4.34.4 Operands

Locales that exist in the directory structure `/usr/lib/nls` and which are also in the `/usr/lib/nls/config` file are displayed.

The `locale` command does not display any of the keyword values specified by POSIX in the `LC_CTYPE` and `LC_COLLATE` categories. `locale` does display several additional keyword values for these categories. See *locale*(1).

4.35 localedef – Define locale environment

4.35.2 Description

Users cannot create their own locales.

4.35.3 Options

If the `-f` option is not present, depending on the locale environment setting, one of the following character encoding schemes are used: US ASCII, Roman8, ISO 8859/1, HP15 or EUC.

4.35.4 Operands

If the name does not contain any slash characters, and the input script does not contain the `langname` keyword then the *name* operand is used as the name of the locale. See *localedef*(4)).

4.35.9 Consequences of Errors

No other implementation-defined warning conditions exist.

4.36 logger – Log messages

4.36.2 Description

Messages can be written in any locale.

4.39 ls – List directory contents

4.39.3 Options

In addition to the `-a` option, the `-f` and `-A` options can also be used to display file entries whose names begin with a period (`.`).

If the `ls` command is invoked by the super-user, files whose names begin with a period are listed by default. To avoid this, use the `-A` option as super-user (opposite behavior from non-super-user).

4.39.5.3 Environment Variables

If the `COLUMNS` environment variable is not set it defaults to the value found in the *terminfo*(4) database based on the type of terminal used or the window size. If a terminfo entry is not available, `COLUMNS` defaults to 80.

4.39.6.1 Standard Output

If the output is to a terminal device, the output is formatted according to the `-C` option.

4.43 mv – Move files

4.43.2 Description

On HP-UX systems, all file types can be moved, but only the super-user can move a character or block special file to another file system.

Read access is required for each file duplicated to another file system.

4.45 od – Dump files in various formats

4.45.7 Extended Description

The default number of bytes transformed by output type specifiers `d`, `f`, `o`, `u`, and `x` correspond to the sizes of the C language types as used by the C compiler. See *c89*(1).

The byte order used when interpreting numeric values corresponds to the order which is used to store the values in memory.

When either the `-j` *skip* or `-N` *count* option is specified along with the `c` type specifier, and this results in a position that begins or ends in the the middle of a multibyte character, the interpretation of the byte stream may result in an invalid multibyte character being displayed.

4.48 pax – Portable archive interchange

4.48.2 Description

The default output archive format is tar. See *tar*(1).

Only the last of multiple `-f` and `-t` options are used. See *pax*(1).

4.48.3 Options

The ability to append to the end of an archive is device-dependent. See *mt*(7) and related entries in section 7: Device Special Files in the *HP-UX Reference* for more information.

The `-p` option preserves the last access time of the input files after they have been copied to the archive.

4.55 sed – Stream editor

4.55.7 Extended Description

4.55.7.3 sed Editing Commands

`sed` supports a maximum of 10 *wfile* arguments per script.

`sed` supports a maximum of 8 characters per label.

4.56 sh – Shell, the standard command language interpreter

4.56.4 Operands

The implementation does not perform a search for an executable file based on the `PATH` environment variable.

4.59 stty – Set the options for a terminal

4.59.4 Operands

4.59.4.4 Local Modes

Specifying `iexten` causes `IXANY`, `XCASE`, and `IUCLC` to be set or reset accordingly. See *stty*(1) and *termio*(7).

4.62 test – Evaluate expression

4.62.4 Operands

The following additional implementation-defined primaries are also provided:

`-k` *file*	True if file exists and its sticky bit is set.
`-L` *file*	True if file exists and is a symbolic link.
`-O` *file*	True if file exists and its owner is the effective user ID.
`-G` *file*	True if file exists and its group is the effective group Id.
`-S` *file*	True if file exists and it is a special file of type socket.
file1 `-nt` *file2*	True if file file1 is newer than file file2.
file1 `-ot` *file2*	True if file file1 is older than file file2.
file1 `-ef` *file2*	True if file1 is another name for file file2.

4.63 touch – Change file access and modification times

4.63.3 Options

The range of valid times extends to 01:00 18 January 2038 UTC.

4.68 uname – Return system name

4.68.2 Description

See *uname*(1) for details on the format and contents of each field displayed.

4.68.6.1 Standard Output

Additional information that is displayed when the `-a` option is specified are the machine identification number and a description of the version number.

5

User Portability Utilities Option

The HP-UX operating system does not support the User Portability Utilities Option.

6

Software Development Utilities Option

6.2 make – Maintain, update, and regenerate groups of programs

6.2.7 Extended Description

6.2.7.1 Makefile Syntax

The make utility tries to open the files `makefile`, `Makefile`, `s.makefile`, and `s.Makefile` in that order for input.

6.2.7.2 Makefile Execution

No additional variables are added to `make`'s environment.

The `MAKEFLAGS` environment variable can include other options. See *make*(1).

6.2.7.3 Target Rules

The characters "`%`" and "`\"`" have no special meaning in target names and are treated like regular characters.

6.2.7.4 Macros

Defining `SHELL` in the makefile or on the command line simply replaces the original value of the `SHELL` macro.

Annex A: C-Language Development Utilities Option

A.1 c89 – Compile Standard C programs

A.1.3 Options

The `-g` option is not compatible with the following options: `-O`, `+O1`, `+O2`, `+O3`, `+Obb`, or `+OV`.

If the `-g` and `-s` options are both specified, the `-s` option is ignored.

A.1.4 Operands

In addition to the `.a` filename suffix, `c89` also recognizes the `.sl` suffix as an object file (shared) library.

Files without a suffix, or with a suffix other than `.a`, `.c`, `.i`, `.o`, `.s`, or `.sl` are assumed to be relocatable object files.

A.1.5 External Influences

A.1.5.2 Input Files

A file with the suffix `.i` is assumed to be C source that has already been run through the C preprocessor (see *cpp*(1)). Such files are compiled without invoking the C preprocessor again.

A file with the suffix `.s` is assumed to be assembly source and is assembled.

A.1.7 Extended Description

A.1.7.2 External Symbols

The maximum symbol length is 255 bytes.

The maximum total number of external symbols supported by the C compiler is dependent upon available system resources.

A.2 lex – Generate programs for lexical tasks

A.2.6 External Effects

A.2.6.2 Standard Error

When the `-v` option is specified and the `-n` option is not specified, `lex` statistics are written to the standard error. See *lex*(1) for details on the format of the statistics.

A.2.7 Extended Description

A.2.7.1 lex Definitions

The default data type of *yytext* is `extern unsigned char yytext[];`

See *lex*(1) for a description of various internal table sizes and how they affect the execution of lex.

A.3 yacc – Yet another compiler compiler

A.3.7 Extended Description

A.3.7.9 Limits

See *yacc*(1) for a description of various internal table sizes and how they affect the execution of yacc.

Annex C: FORTRAN Development and Runtime Utilities Options

C

C.1 asa – Interpret carriage-control characters

C.1.2 Description

If the first character of the line is `1` (numeral one), it is replaced with an ASCII carriage-return, form-feed character pair in the output stream.

If the first character of the line is `+` (plus symbol), it is replaced with an ASCII carriage-return character in the output stream.

C.2 fort77 – FORTRAN compiler

C.2.3 Options

The `-g` option is not compatible with the following options: `-O`, `+O1`, `+O2`, `+O3`, `+Obb`, `+OP`, or `+OS`.

If the `-g` and `-s` options are both specified, the `-s` option is ignored.

C.2.4 Operands

In addition to the `.a` filename suffix, `fort77` also recognizes the `.sl` suffix as an object file (shared) library.

Files without a suffix, or with a suffix other than `.a`, `.c`, `.f`, `.F`, `.o`, `.r`, `.s`, or `.sl` are assumed to be relocatable object files.

C.2.5 External Influences

C.2.5.2 Input Files

A file with the suffix `.c` is assumed to be K&R C source that is to be compiled by the C compiler. See *cc*(1).

A file with the suffix `.F` is assumed to be FORTRAN source that is to be preprocessed by the C preprocessor prior to compilation. See *cpp*(1)).

A file with the suffix `.r` is assumed to be ratfor source that is to be preprocessed by the ratfor preprocessor prior to compilation. See *ratfor*(1)).

A file with the suffix `.s` is assumed to be assembly source and is assembled.

C.2.7 Extended Description

C.2.7.2 External Symbols

The maximum symbol length is 255 bytes.

The maximum total number of external symbols supported by the FORTRAN compiler is dependent upon available system resources.

C.2.9 Consequences of Errors

The FORTRAN compiler does not produce an object module if it encounters a compilation error.