900 Series HP 3000 Computer Systems

New Features of MPE/iX: Using the Hierarchical File System

HEWLETT PACKARD

HP Part No. 32650-90351
Printed in U.S.A. 1994

Second Edition
E0494

Acknowledgements

Printing History

The following table lists the printings of this document, together with the respective release dates for each edition. The software version indicates the version of the software product at the time that this document was issued. Many product releases do not require changes to the document; therefore, do not expect a one-to-one correspondence between product releases and document editions.

Edition	**Date**	**Software Version**
First Edition	October 1992	C.45.00
Second Edition	April 1994	C.50.00

Preface

MPE/iX, Multiprogramming Executive with Integrated POSIX, is the latest in a series of forward-compatible operating systems for the HP 3000 line of computers.

In HP documentation and in talking with HP 3000 users, you will encounter references to MPE XL, the direct predecessor of MPE/iX. MPE/iX is a superset of MPE XL. All programs written for MPE XL will run without change under MPE/iX. You can continue to use MPE XL system documentation, although it may not refer to features added to the operating system to support POSIX (for example, hierarchical directories).

Finally, you may encounter references to MPE V, which is the operating system for HP 3000s not based on PA-RISC architecture. MPE V software can be run on the PA-RISC (Series 900) HP 3000s in what is known as *compatibility mode*.

About This Manual

This manual describes changes to MPE/iX as of Release 4.5 and 5.0. The major changes introduced as of these releases enhance MPE/iX to make it "POSIX-compatible." POSIX is an IEEE standard for a portable operating system interface to support source-level application portability. This document introduces the enhancements by means of a tutorial, conceptual background chapters, and chapters that present tasks associated with the new features of MPE/iX.

Who This Manual Is For

This manual is for experienced HP 3000 system administrators, system programmers, application developers, and other MPE/iX system users who are interested in learning about Release 4.5 and 5.0 enhancements.

All readers of this document should be familiar with performing the following tasks:

- Logging on and off the system
- Accessing files
- Using the MPE command interpreter (CI)
- Executing commands such as `LISTF`, `NEWGROUP`, `CHGROUP`, and `ALTSEC`
- Using UDCs

Chapters 3, 7, and 8 are specifically for experienced system administrators. These readers should be familiar with the following tasks:

- Adding users
- Adding accounts
- Monitoring disk space usage
- Creating and managing files
- Backing up and recovering the system

What's in Each Chapter

Chapter 1 is a tutorial for anyone who wants to learn about what's new by trying some of the commands. This chapter also includes a brief overview of the new features. All users should read this chapter.

Chapter 2 provides a conceptual overview of what's new. It includes information about MPE/iX file and directory name syntax and lists new and changed commands and utilities. All users should read this chapter.

Chapter 3 summarizes the major changes that system administrators need to know about such as the addition of the user and group databases, backup and restore changes, expanded search capabilities, and security enhancements.

Chapter 4 provides an overview of new and enhanced features accessible through MPE/iX system intrinsics. The new features described are hierarchical directories, MPE/iX byte-stream files, renaming a file across account boundaries, and append mode. Security, object ownership, and file manipulation commands and intrinsics have been enhanced. Although the information in this chapter is particularly relevant for programmers, other advanced users would benefit from reading it.

Chapter 5 defines directories and explains how to name and create them. It also includes step-by-step instructions on how to list directories, list directory contents, move from one directory to another, determine the amount of space used by directories, and delete directories. All users should read this chapter.

Chapter 6 describes changes to tasks that relate to files including naming files, listing files, displaying files, deleting files, working with byte stream files, and spooling files. All users should read this chapter.

Chapter 7 describes the group and user databases and how they are created, and tells how to keep them up to date. It also discusses new options provided when adding users onto MPE/iX. This chapter is most relevant to system administrators.

Chapter 8 describes storing and restoring files, explains new options to the `STORE` and `RESTORE` commands, and reminds system administrators to modify the backup procedures to back up the hierarchical file system.

Chapter 9 presents enhancements to file system security. It includes a brief discussion of access control definitions and access modes and describes tasks such as listing ACDs and changing access to files and directories. This chapter is most relevant to system administrators, but it includes information for all users.

The Glossary defines many of the new terms used throughout this manual.

For More Information

For information on MPE/iX, refer to the full MPE/iX documentation set. The *MPE/iX Documentation Guide* (32650-90144) describes all available manuals.

For details on new commands, refer to the *MPE/iX Commands Reference Manual, Volumes I & II* (32650-60115).

For details on intrinsics, refer to the *MPE/iX Intrinsics Reference Manual* (32650-90028).

For information on error messages, refer to the *Error Message Manuals, Volumes I, II, & III* (32650-60016).

Refer to the IEEE 1003 Standards documents, for information about the POSIX standards:

- 1003.1 - C language bindings
- 1003.2 - Shell commands and utilities

Additional manuals are available with the MPE/iX Developer's Kit (36430A). This kit is an optional product for programmers planning to develop POSIX applications or use POSIX functions in programming applications on MPE/iX. The MPE/iX Developer's Kit includes the following manuals:

- *MPE/iX Developer's Kit Reference Manual, Volumes I and II* (36430-90001 and 36430-90002) describe programmatic interfaces and extensions to the C library.
  - *MPE/iX Shell and Utilities User's Guide* (36431-90002) provides a command overview and tutorials on the MPE/iX shell and describes utilities such as make, yacc, lex, and awk.

    - *MPE/iX Shell and Utilities Reference Manual, Volumes I and II* (36431-90001 and 36431-90003) contain complete descriptions of all MPE/iX shell commands.

- *HP C/iX Reference Manual* (31506-90005) provides a complete reference for the C programming language.
- *HP C Programmer's Guide* (92434-90002) includes information on programming in C.
- *HP C/iX Library Reference Manual* (30026-90001) describes C library routines that are available in the POSIX.1/iX library.
- *The POSIX.1 Standard: A Programmer's Guide* (36430-90003) by Fred Zlotnick, Benjamin/Cummings Publishing Company, Inc., 1991. A programmer's guide to using POSIX.2 C library functions to create portable programs.

How to Use This Document

This document is written for a broad audience. Certain sections are more relevant to some readers than others. The following table shows you what to read depending on your focus and what kind of information you need.

Note

If you normally work within an application while using MPE/iX, you don't need to read this manual. It is for people who work from the MPE/iX command interpreter (CI).

Where to Go for Information

Information Covered	What to Read	Who Should Read
Overview of new features in tutorial form	Tutorial, Chapter 1	All users
Overview of all MPE/iX changes including programming and system administration changes	Whole manual	Programmers, system managers, system operators, general users
Programming information including intrinsics changes	Chapters 2, 4, 6, 7	Programmers, advanced system managers
System administration information	Chapters 2, 3, 5, 6, 7, 8, 9	System managers, system operators
General overview of new features	Chapters 1, 2, 3, 4, 6, 7, 9	All users
Security enhancements	Chapter 3 or 4 and Chapter 9	System managers (read Chapters 3 and 9), programmers (read Chapters 4 and 9)
Backup and restore changes	Chapter 8	System managers, system operators, and programmers

Contents

Figures

Tables

1

What's New on MPE/iX: Tutorial

As of Release 4.5 and 5.0, MPE/iX has hierarchical directories, offers new commands, and allows you to use new file naming conventions among other new features. Many users want to learn by trying out the new features right away. This tutorial allows you to do just that. It doesn't try to explain everything. It just introduces you to some of the major enhancements. Other sections of this manual include more information and examples.

This tutorial is for anyone who wants to use the new features: system managers, system operators, programmers, or other system users. It assumes that you are familiar with MPE and with typing MPE commands from the MPE/iX command interpreter (CI).

You can also use the online POSIX Computer-Based training to learn about the new features of MPE/iX. To take the training, you need to log on to your MPE/iX system and type the following command:

```
:POSIXCBT.LSN.SYS
```

Summary of New Features

This section briefly introduces some of the new features of MPE/iX that you will encounter in this tutorial:

- Hierarchical file system (HFS)
- HFS file names
- HFS syntax
- New and enhanced commands

Hierarchical file system (HFS)

As of Release 4.5 and 5.0, the MPE/iX file system is *hierarchical* (tree structured) and can contain files at many different levels. This organization provides a special kind of file called a **directory**. Instead of holding data, directories contain lists of files and pointers to those files. A directory can also contain other directories. This organization is similar to the file systems on UNIX®or MS-DOS® systems.

The new file organization still includes the familiar accounts, groups, and users. The hierarchical file system (called HFS, for short) extends the traditional MPE file system features so the operating system is more flexible. You'll learn more about HFS as you work through the tutorial. But keep in mind, unless you're going to use a new application that requires the new items (such as directories), you can continue to use MPE as you have in the past.

You're used to referring to files, groups, and accounts using the traditional MPE syntax: `FILE1.PUB.SYS`. You can still use MPE syntax. You can also make use of a new syntax called HFS syntax, which looks like this: `/SYS/PUB/FILE1`.

The MPE/iX enhancements are compared to previous releases in Table 1-1.

Table 1-1.
Where Accounts, Groups, Directories, & Files Can Be Located

Location	Before Release 4.5	Release 4.5 and After
Highest level	Accounts	Root
Under root	Root not visible	Accounts, directories, or files
Under accounts	Groups	Groups, directories, and files
Under groups	Files	Directories or files
Under directories	Directories not available	Directories or files

Figure 1-1 shows how you can organize files, accounts, groups, and directories in the file system. Notice that accounts, directories, groups, and files all connect back to one directory designated by a slash (/). This is referred to as the *root* or the *root directory*.

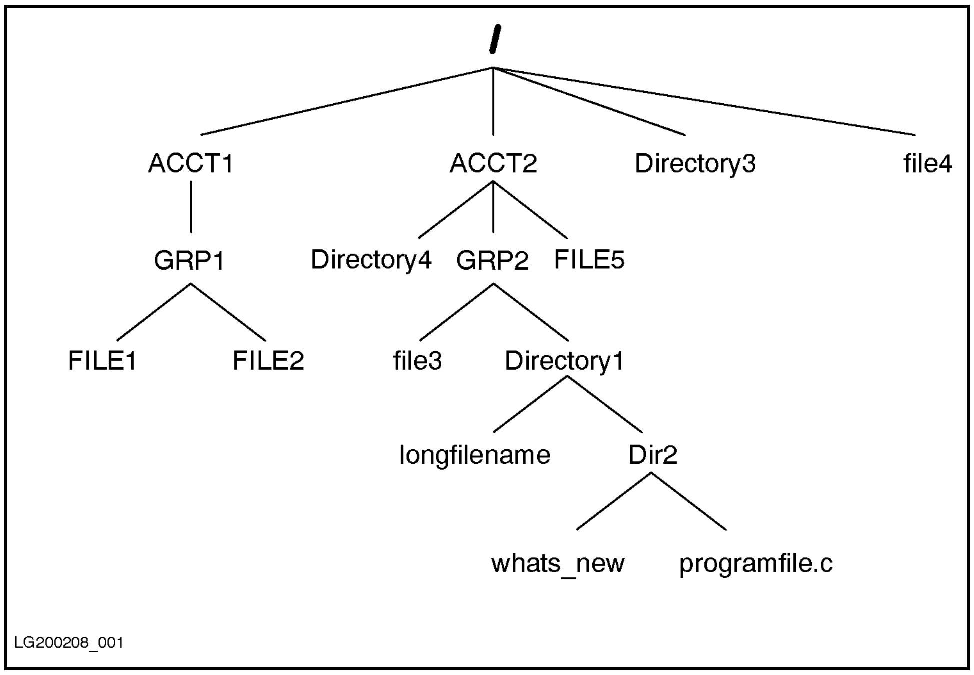


Figure 1-1. MPE/iX File System Example

HFS file names

MPE/iX now allows you to assign longer file names than in previous versions of MPE/iX. Table 1-2 summarizes name lengths for accounts, groups, directories, and files previous to Release 4.5 or greater.

Table 1-2. Maximum Lengths of Account, Group, Directory, and File Names

Type	MPE Syntax	HFS Syntax
Account name	8 uppercase characters	8 uppercase characters
Group name	8 uppercase characters	8 uppercase characters
Directory name	Not available	16 mixed case characters if directly under root, account, or a group. Up to 255 characters elsewhere.
File name	8 uppercase characters	16 mixed case characters if directly under root, account, or a group. Up to 255 characters elsewhere.

See “Summary of MPE/iX syntax features” in Chapter 2 for additional information about name lengths.

HFS syntax

Table 1-3 summarizes some of the syntax enhancements introduced by the MPE hierarchical file system. The syntax that you are used to still works for files in groups and accounts. So to use HFS syntax, you must precede file and directory names with `./` or `/`. Otherwise, MPE/iX treats the names using traditional MPE syntax rules.

This manual refers to files that are named using HFS syntax as *HFS files*.

Table 1-3. Syntax Summary

Item	MPE Syntax	HFS Syntax
Specify file name	No special beginning character required: `FILE.GRP.ACCT`	Name must be preceded by a ./ (dot slash) or / (slash): `/ACCT` or `./dir1`
Name separators	. (period); / separates lockwords	/ (slash)
Way of specifying files	Bottom up: `FILE.GRP.ACCT`	Top down: `/ACCT/GRP/FILE`
Case sensitivity	Not case sensitive; all characters are shifted to uppercase	Case sensitive: `/DIR/FILE1` and `/DIR/file1` are two different files
Special characters	Only alphanumeric characters	Alphanumeric, - (hyphen), . (dot), and _ (underscore) are allowed
First character	Must be alphabetic	Can be alphanumeric, _ (underscore), or . (dot) but not - (hyphen)

New commands

- `ALTFILE` alters a file's attributes such as UID and GID of a file, directory, and MPE group or account (except that the GID may not be altered for MPE groups or accounts).
- `CHDIR` lets you move your CWD (much like changing groups).
- `DISKUSE` displays disk space used by accounts, groups, and directories, that is, all items in the hierarchical file system. (The `REPORT` command still provides disk space accounting for accounts and groups. It just won't include detailed information about directories below accounts, and it does not report on directory structures directly below the root.)
- `NEWDIR` lets you create directories. After creating directories, you can use the `ALTSEC` command if you want to grant access to other users.
- `NEWLINK` creates a symbolic link to a file, group, account, or directory.
- `PURGEDIR` is for deleting directories.
- `PURGELINK` is for removing a link.
- `SETCLOCK` alters the system time or system time zone.
- `SHOWCLOCK` displays information about the system date and time.

Enhanced commands

- `FILE` declares the file attributes to be used when a file is opened. This declaration, informally known as a file equation, may be used to override programmatic or system default file specifications. With the addition of shared parameters from the NS3000/XL AdvanceNet subsystem, the declaration may specify a formal file designator that may be used to access a remote file or device in a subsequent command or intrinsic.
- `LISTF` (UDC) executes the `LISTFILE` command to list descriptions of one or more disk files.

- `LISTFILE` allows you to display directories and all HFS file names.
- `PURGE` deletes a file from the system.
- `RESTORE` returns files that have been stored on magnetic tape to the system.
- `RUN` executes a prepared or linked program. The only required parameter is *progfile*. If you specify any other parameters, they will override the default parameters that the creator of the program established, but only for that particular execution of the program.
- `SAVE` command saves a file from the temporary domain to the permanent domain.
- `SHOWDEV` reports the status of input/output devices.
- `SHOWPROC` displays information about one or more processes.
- `STORE` copies disk files onto a magnetic tape. Files copied to tape with the `STORE` command can be recovered with the `RESTORE` command.
- `VSTORE` verifies that the data on a backup media are valid and reports errors incurred by `STORE` when writing the tape. `VSTORE` only applies to `NMSTORE` tapes created in native mode. It does not work on tapes created in compatibility mode.

Symbolic Links

Symbolic linking is the concept of indirectly referring (pointing) to another file on the system. This concept is similar to MPE/iX file equations. A symbolic link is a type of file that contains another path name. It is a permanent file in the system directory. Symbolic links can be created, removed, renamed, and archived.

Before You Start

Log on to the system

Log on to your account as you normally do using your user name, account, and passwords.

Note

When trying commands in this tutorial, substitute the name of your account for YOURACCT and the group you are working in for YOURGRP. Your procedure may vary from the one shown above.

For example:

```
:HELLO YOURNAME.YOURACCT
ENTER ACCOUNT (ACCOUNT) PASSWORD:
ENTER USER (USER) PASSWORD:

HP3000 RELEASE: C.45.00 USER VERSION C.45.00 WED, MAY 10,
1992 9:18 AM
.
.
.
```

Find out which release you're running

To use the tutorial, you need to have MPE/iX Release 4.5 or greater installed on your system. You can find out what release is running by asking your system manager or by using the SHOWVAR command.

To use SHOWVAR to find out what release you are running:

```
:showvar hpversion
HPVERSION = C.45.00
```

If the HPVERSION is C.45.00 or later, continue with the tutorial.

Create a file to work with

To build a file in `YOURGRP.YOURACCT` that you will use later in this tutorial, enter the following command:

```
:BUILD FILE1
```

Learning by Doing

You can try some of the new features of MPE/iX with the examples in the rest of this tutorial.

Use the HFS syntax

When you log on to the system, you are in a group within an account such as `YOURGRP.YOURACCT`. Files you work with, such as `FILE1`, can be referred to as `FILE1.YOURGRP.YOURACCT` or `file1.yourgrp.youracct`. MPE syntax automatically converts lowercase letters to uppercase. With the advent of the hierarchical file system, you can also refer to a file using the HFS syntax. That same file can be referred to as `/YOURACCT/YOURGRP/FILE1`. It is the path to the file or the *pathname*.

Note

To refer to an MPE file name using the HFS syntax, you *must* type the name using all capital letters; otherwise, the HFS syntax looks for the name using lowercase letters and won't locate it. HFS syntax does not convert lowercase letters to uppercase.

```
/YOURACCT/YOURGRP/FILE1 = FILE1.YOURGRP.YOURACCT
```

The first `/` in the pathname refers to the level above the account and is called the *root directory*. It is used as a way to reference files on the system. See Figure 1-2.

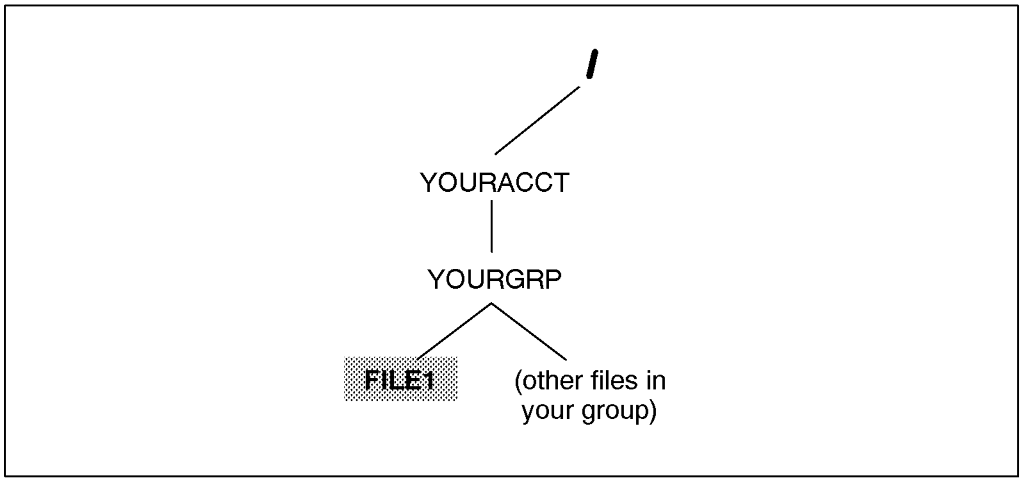


Figure 1-2. Location of the File That You Created

Figure 1-2 shows a picture of your file's location in the file system. `YOURACCT` is one level below the root directory. You logged on to `YOURACCT` and were placed into `YOURGRP`. Then you built a file called `FILE1`. Any other files that you have in `YOURGRP` are also located there with `FILE1`.

List the file

You can refer to a file using either the MPE or HFS syntax. Earlier you created a file called `FILE1`.

Note

The use of the `LISTFILE` command in this section is intentional. The `LISTF` command does not recognize HFS syntax.

You can list the file with MPE syntax:

```
:LISTFILE FILE1.YOURGRP.YOURACCT
```

OR (using the exact capitalization shown)

```
:LISTFILE /YOURACCT/YOURGRP/FILE1
```

Both display the same result:

```
FILE1
```

Be careful! HFS syntax is case sensitive but MPE syntax is not. You can type the following command, using MPE syntax in lowercase, and locate uppercase `FILE1` in `YOURGRP` in `YOURACCT`. Try it.

```
:listfile file1.yourgrp.youracct
```

That's because MPE syntax upshifts everything. But because HFS syntax is case sensitive, the following command will not find `FILE1` in `YOURGRP` in `YOURACCT`:

```
:listfile /youracct/yourgrp/file1
A component of the pathname does not exist. (CIWARN 9053)
```

`YOURACCT` is not the same as `youracct` in HFS syntax.

Create a directory

A **directory** is a work area similar to an MPE group where you can put related files. You can create directories in your group and account. You must precede file and directory names with `./` or `/` to use HFS syntax. Dot (.) means **current working directory** or where you are working within the file system. So `./` lets you use HFS syntax in your current working directory.

```
:NEWDIR ./Directory1
```

This command creates `Directory1` in your current working directory.

HFS syntax accepts the name exactly as you type it in uppercase and lowercase, and the name can be up to 16 characters long when created directly under a group. (Refer to Table 1-2 shown previously for maximum lengths of directory names.)

Watch out! If you try to create the directory without including `./` before the name, MPE/iX treats the name as a regular MPE name. It converts the letters to all uppercase. Try typing the following command:

```
:NEWDIR Directory1
File name is more than eight characters long. (CIERR 532)
```

Also notice: If you type the following command, MPE/iX creates a directory called `DIR2` (uppercase). That is because you are using MPE syntax. Try it.

```
:NEWDIR dir2
```

In the next section, you'll learn how to list the directory.

Figure 1-3 shows a picture of the file system including the directories that you just created.

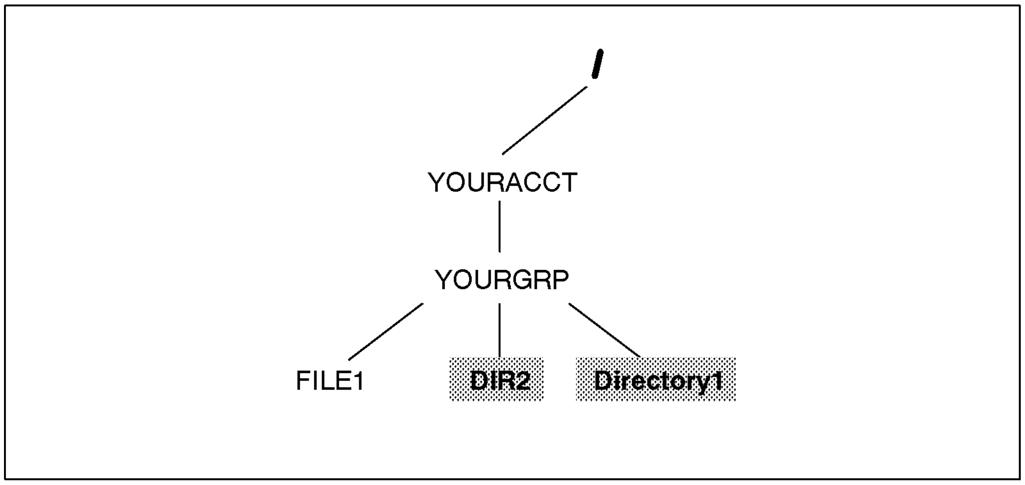


Figure 1-3. Location of the Directories That You Created

List the directories

You use the LISTFILE command to list directories as well as files. Directories are special types of files.

```
:LISTFILE ./Directory1


PATH=/YOURACCT/YOURGRP/./

Directory1/
```

You can tell that it's a directory (rather than a file) in the command output because its name is followed by a slash.

To list all files and directories in your current working directory:

```
LISTFILE ./@


PATH=/YOURACCT/YOURGRP/./

DIR2/          Directory1/    FILE1
```

All files (including directories) in your current working directory are listed.

If you omit the ./ from the command, the LISTFILE command assumes that you are looking only for MPE-named files, and HFS-named files are not displayed:

```
:LISTFILE @

FILENAME

DIR1
FILE1
```

If you have other files in the group where you are working, they are listed along with FILE1.

The result is the same if you omit @ and type LISTFILE since that is the default when no parameter is included with the command.

List directories another way

Note

You can only use the system-provided UDCs, such as LISTDIR, if the system manager has activated them. Refer to the section “UDCs, JCL, Command Files, and Programs” in Chapter 3 for more information.

You can also use the LISTDIR UDC to list directories:

```
:LISTDIR

/YOURACCT/YOURGRP/DIR2/
/YOURACCT/YOURGRP/Directory1/
```

All the directories in your current working directory are listed when you specify the UDC with no parameters.

You can also use the `LISTDIR` UDC to list directories in a specific location:

```
:LISTDIR /SYS/PUB
```

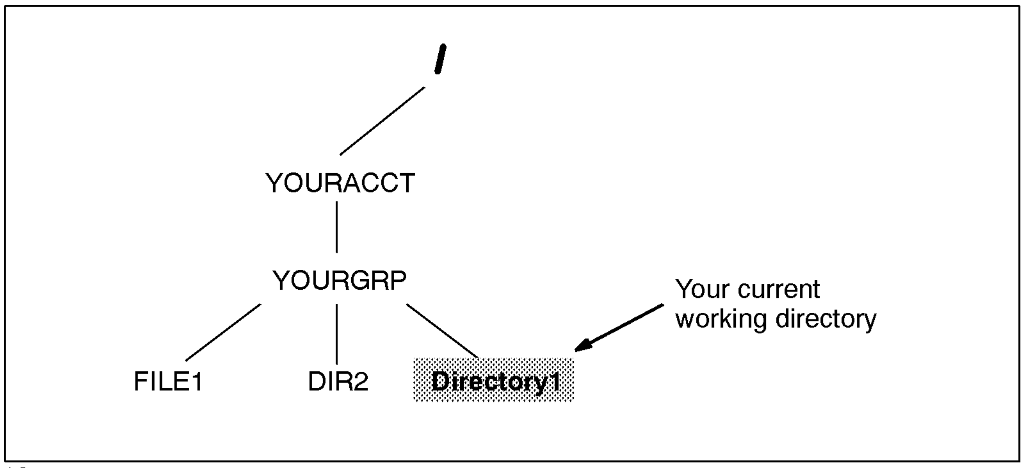
This command locates any directories located in `PUB.SYS`.

Move to the directory

After you create a directory, you can move to it.

```
:CHDIR ./Directory1
```

This command moves you to `Directory1`. Figure 1-4 shows you where you moved in the file system. `Directory1` is now your current working directory.



LG200208_004

Figure 1-4. Location after Changing Directories

Again, use a `./` to act on HFS files. If you don't, MPE/iX acts only on MPE-named files. If you try to change to `Directory1` without the preceding `./` the system treats the directory name as an MPE name and converts the letters to uppercase. It does not locate the directory.

```
:CHDIR Directory1

File name is more than eight characters long. (CIERR 532)
```

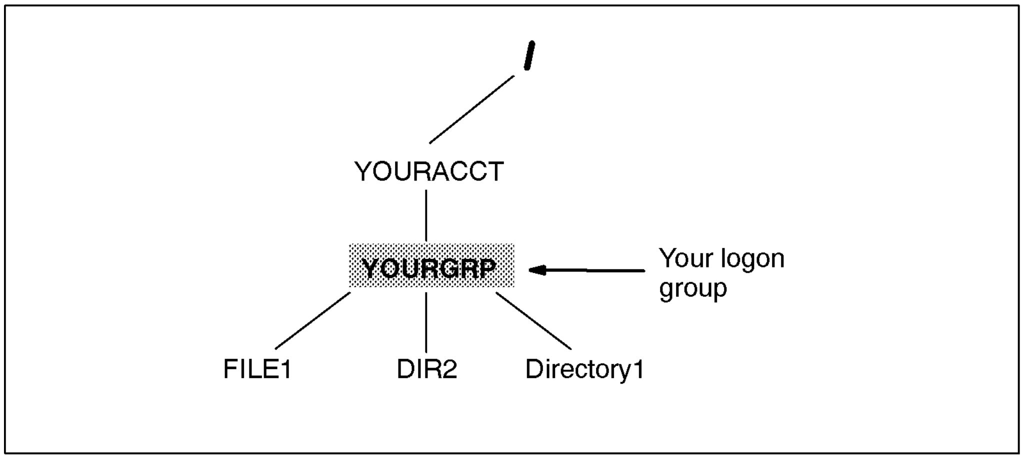
According to MPE syntax, the file name is limited to eight characters, but actually, two errors occur here. First, the system discovers that the name is longer than eight characters and reports the error. Second, the name is upshifted, and the system looks for `DIRECTORY1`, which it wouldn't find even if it were fewer than eight characters.

Move back to your logon group

If you type

```
:CHDIR
```

by itself, MPE/iX moves you back to your logon group. Try typing the command. Figure 1-5 shows you where you are now. `YOURGRP` is your current working directory again.



LG200208_005

Figure 1-5. Moving Back to Your Logon Group

Move to DIR2

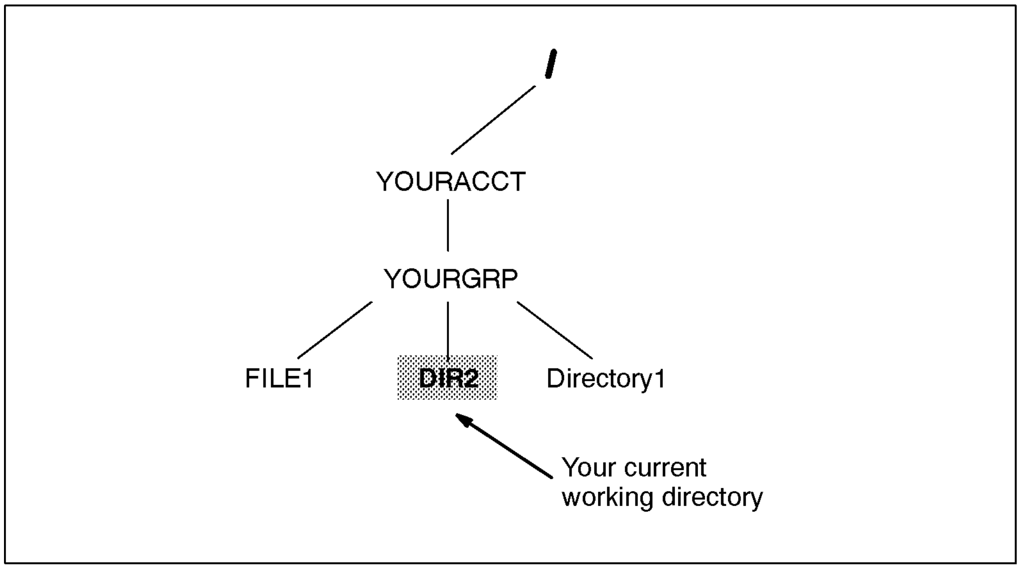
You can move to the DIR2 directory that you created earlier.

```
:chdir dir2
```

This command moves you to the directory called DIR2 since the MPE syntax rules are followed; these rules convert the name to uppercase. Notice when using HFS syntax, you can type the commands in uppercase or lowercase. Using lowercase is easier for some people. Otherwise, you can force specific case for a name by beginning the name with ./ as shown here. Since you have already changed directories, you'll need to change back to your logon group (using CHDIR) to try the following example:

```
:chdir
:chdir ./DIR2
```

Figure 1-6 shows you where you are now. DIR2 is your current working directory.



LG200208_006

Figure 1-6. Moving to DIR2

Create a subdirectory

You can also create a directory within a directory. The previous examples put you in the DIR2 directory. Now try creating another directory with a long name:

```
:newdir ./Long_Directory_Name
```

Note

HFS syntax allows you to use special characters in the name, such as "_" (underscore). You can also use characters like the "-" (hyphen) and "." (period or dot). By convention, the dot in a name is usually used to separate the name from an extension that has some significance. For example, names of source programs written in the C language often have a `.c` extension such as `prog1.c`.

List the subdirectory

Using LISTFILE or the LISTDIR UDC is the best way to list directories and subdirectories.

```
  :listfile ./@


  PATH=/YOURACCT/YOURGRP/DIR2/./

  Long_Directory_Name/
```

Subdirectories can contain files or directories with names of up to 255 characters.

Using LISTDIR also lists directories. The output is similar:

```
:listdir
/YOURACCT/YOURGRP/./DIR2/Long_Directory_Name/
```

Move to the subdirectory

You change to subdirectories as well as directories using the CHDIR command:

```
:chdir ./Long_Directory_Name
```

Show where you are

CWD is short for *current working directory.* That's where you are located in the hierarchical file system. To find out your CWD, type the following command:

```
:showvar hpcwd
HPCWD = /YOURACCT/YOURGRP/DIR2/Long_Directory_Name
```

The location of the directory is presented from the root directory down to the current location in a *pathname.* The pathname is the path to the directory.

Names are shown top-down rather than bottom-up. The first / refers to the root directory. All directories, accounts, and files exist under the root. Slashes separate the pathname components. The last component of the pathname is the name of the current working directory.

Change your prompt to show the current working directory

You used to use the SHOWME command to find out your logon group. The SHOWME command does not show the current working directory, but you can change your prompt so that it shows where you are at all times.

Note

Do not execute the following command unless you want to change your existing prompt.

```
:setvar save_prompt hpprompt
:setvar hpprompt "!!HPCWD:"
/YOURACCT/YOURGRP/DIR2/Long_Directory_Name:
```

Having your prompt set to the current working directory makes it obvious where you are as you move around in the hierarchical file system. The first command in the above example also saves your current prompt in case you want to return to it. Here's the command you use to reset your old prompt:

```
:setvar hpprompt save_prompt
```

Create a file with a long file name

Move back to your logon directory:

```
/YOURACCT/YOURGRP/DIR2/Long_Directory_Name:chdir
/YOURACCT/YOURGRP:
```

Both MPE syntax and HFS syntax will work with your existing files or files with names conforming to MPE syntax. However, you must use HFS syntax to specify file names using HFS naming syntax (such as those using special characters or lowercase letters). The rest of the examples in this tutorial use the colon prompt rather than the name of the current working directory. Try each of these commands to copy an existing MPE file into DIR2:

```
:COPY FILE1.YOURGRP.YOURACCT,/YOURACCT/YOURGRP/DIR2/VeryLongFilename
:

:COPY /YOURACCT/YOURGRP/FILE1,/YOURACCT/YOURGRP/DIR2/AnotherLongFilename
:
```

List files and directories in your CWD

Using the LISTFILE command with option 2 provides useful output.

In the following example, the file type for the directories, DIR2 and Directory1, is DBH. This is a new type that indicates that these are *directories* in *binary* form and they are *hierarchical* (as opposed to being groups or accounts).

```
:listfile /YOURACCT/YOURGRP/@,2

 PATH=/YOURACCT/YOURGRP/

 CODE  ------------LOGICAL RECORD-----------  ----SPACE----
FILENAME
         SIZE  TYP        EOF      LIMIT R/B  SECTORS #X MX

         128W  FB           0       1023   1        0  0  *
AnotherLongFilename
          16W  DBH          4   67107839   1       32  1  *
DIR2/
          16W  DBH          4   67107839   1       32  1  *
Directory1/
          80B  FA         411        411   1      144  2  *
FILE1
         128W  FB           0       1023   1        0  0  *
VeryLongFilename
```

You can use many options with the LISTFILE command to show additional information about the files and directories.

Delete a directory

To delete `Directory1`:

```
:purgedir ./Directory1
Directory "./Directory1" to be purged? (Yes/No)y
```

MPE/iX deleted the directory because it was empty. (You need to use the `TREE` option with the `PURGEDIR` command to delete a directory that is not empty.)

Learn about absolute and relative pathnames

HFS syntax, like MPE syntax, allows the use of absolute pathnames and relative naming. An *absolute pathname* is a pathname where all components, starting with the root, are specified (for example, `/SYS/PUB/CI`).

HFS syntax also allows the use of relative pathnames. A *relative pathname* allows you to specify a file by beginning from your CWD. For a relative pathname, start the name specification with a "./" (dot slash).

1. Make sure that you are working in `/YOURACCT/YOURGRP` by issuing the following command:

   ```
   :chdir
   ```

   (You can use `CHDIR` to move to a group because accounts and groups are special types of directories.)

2. Now try the following command:

   ```
   :LISTFILE /YOURACCT/YOURGRP/@,6
   ```

   The command says, "list everything in `/YOURACCT/YOURGRP`." It specifies the absolute pathname. Notice the output.

3. Try this command, which uses the relative pathname "./":

```
:LISTFILE ./@,6
```

The command says, "list everything in the current working directory." Like the last command, it also lists the contents of /YOURACCT/YOURGRP because you are currently working in that directory.

Try a few more LISTFILE commands

You can use the TREE option to show all accounts, groups, files and directories below the specified point.

```
:LISTFILE /YOURACCT;TREE
```

Including a slash at the end of a directory name in the LISTFILE command is the same as specifying the keyword TREE in the command line.

```
:LISTFILE /YOURACCT/
```

List sets of objects on the system

```
:LISTFILE /
```

This command shows all accounts, directories, and files on the system. (It replaces LISTF @.@.@.) In the command shown, the / does triple duty: first it tells MPE/iX to interpret the name using the HFS syntax; second, it calls the TREE option; and third, the / represents the root directory.

```
:LISTFILE /@
```

The above command shows all accounts, directories, and files exactly one level below root.

```
:LISTFILE /@/@/@/@
```

The above command shows all directories or files exactly four levels below the root directory.

Search for files

You can use the `NAME` option to look for an account, group, directory, or file.

```
:LISTFILE /,6; NAME=D@
```

This command searches for all directories, accounts, groups, and files on the system that begin with `D` (or `d`). It searches through all levels of the file system hierarchy.

In the MPE account structure prior to Release 4.5 (when there were only three levels), the following command sequence would have been used:

```
LISTF D@.@.@
LISTF @.D@.@
LISTF @.@.D@
```

You can still use the above `LISTF` commands to locate files beginning with D, groups beginning with D, and accounts beginning with D, respectively. However, the commands only locate files in groups. Files beginning with D in directories are not located. No directories are listed, and no files with names having lowercase letters are found using `LISTF`.

System Management Tasks

Additional tasks to try are restricted to users having appropriate authority, system managers (having SM capability).

Create a directory under root

MPE/iX HFS allows you to create directories and files directly under the root directory. These directories and files are at the same "file level" as MPE accounts.

```
:NEWDIR /dir10
:LISTFILE /dir10


PATH=/

dir10/
```

This example creates, then lists, the directory `dir10` created under the root directory.

Security and directories

The ability to move to, change, and work in directories is controlled by access control definitions (ACDs). ACDs pair up specific access permissions and users. The following access permissions apply to directories:

- Create directory entries (CD)
- Delete directory entries (DD)
- Read directory entries (RD)
- Traverse directory entries (TD)

As a system manager, you can access all directories. Other users are able to access them only if they have the proper permissions.

Check directory permissions

Use `LISTFILE` with the `ACD` option to check directory permissions.

For example:

```
:listfile /dir10,acd

 PATH=/

-------------------ACD ENTRIES---------------------FILENAME

@.@                         : RACD                 dir10/
```

In the example, ACDs are listed for the directory called `dir10`. All users only have read ACD access to `dir10`. Only the creator and the system manager can access the contents of the directory or traverse the directory.

Change directory permissions

To make directories accessible to your users, you will need to change the ACDs on the new directories. You do this using the `ALTSEC` command.

```
:ALTSEC /dir10;REPACD=(CD,DD,RD,TD,RACD : @.@)
```

This command gives all users in all accounts unlimited access to the directory `dir10`.

Move to a directory under root

Users with appropriate access (in this case, you only need TD access) can move to a directory that is directly below root.

```
:chdir /dir10
/dir10:
```

This command moves you into directory `dir10` and makes it your CWD.

Report disk usage

Use the `DISKUSE` command to report disk usage information about directories:

```
:diskuse /YOURACCT/YOURGRP
          SECTORS
    TREE       LEVEL   DIRECTORY
               BELOW
      96          64   /YOURACCT/YOURGRP/
```

The `REPORT` command provides information about files in groups only.

Compare the output by issuing the comparable `REPORT` command:

```
/dir10:REPORT YOURGRP.YOURACCT
ACCOUNT   FILESPACE-SECTORS   CPU-SECONDS   CONNECT-MINUTES
   /GROUP   COUNT    LIMIT   COUNT  LIMIT    COUNT   LIMIT
YOURACCT       64       **      18     **       26      **
   /YOURGRP    64       **       0     **        0      **
```

Back up all files and directories

You can still use the following command to store all of the files and directories on the system:

```
:STORE @.@.@
```

You can also use:

```
:STORE /
```

2

What's New: Overview

This chapter provides an overview of MPE/iX enhancements as of Release 4.5 and 5.0 for system administrators, programmers, and other MPE/iX system users. It discusses the following topics:

- Open systems environment
- Hierarchical file system (HFS)
- Expanded file naming syntax
- New and enhanced commands and utilities
- MPE/iX Shell and Utilities
- MPE/iX Developer's Kit

If you have read and worked through the tutorial in Chapter 1, the concepts explained in this chapter will be somewhat familiar to you.

Introducing a New Open Systems Environment

MPE/iX Releases 4.5 and 5.0 introduce open systems environment features that greatly enhance the flexibility of the MPE operating system with which you are already familiar. The MPE/iX enhancements implement features of POSIX, the IEEE operating system interface standard.

One of the primary advantages of MPE/iX is that the added flexibility is optional—you can use the system essentially as you always have, or you can use the enhanced features of MPE/iX. MPE/iX Release 5.0 is fully backwards and forward compatible with earlier versions of MPE.

What's POSIX?

POSIX is an acronym for Portable Operating System Interface. The POSIX standard, IEEE Std 1003.1-1990, is an international standard that defines common interfaces as a basis for open systems. IEEE POSIX 1003 is actually a group of standards, each of which addresses a specific area of system technology. The POSIX standards describe functions of an operating system interface that applications must use if they are to be "POSIX compliant." Systems that implement the standard functions are said to be POSIX compliant.

Initially, MPE/iX is implementing two of the standards to maximize the software portability of applications: C language application programming interfaces (also called POSIX.1) and a command interpreter (shell) and utilities (called POSIX.2).

The main reason for implementing POSIX.1 functions on a system is to increase software portability and minimize porting costs. Since POSIX.1 is based on Berkeley Software Distribution and AT&T's System V Interface Definition, UNIX applications can be ported more easily to HP 3000 computer systems. Applications developed on any POSIX-compliant system using POSIX.1 functions can be ported to other POSIX-compliant systems and run with little or no modification.

The initial implementation of POSIX.1 makes MPE/iX POSIX compatible. That is, it supports most of the key POSIX features including the hierarchical directory structure, long file names, and process signals. (**Signals** notify processes of events that occur on the system, such as hardware exceptions.)

POSIX.1 defines procedural interfaces for C programs to high-level, basic operating system services. These services include creating and purging directories and files; creating, controlling, and terminating processes; interprocess communications (for example, signals); byte- stream files; user identification; grouping users for file sharing purposes; defining and altering file security; grouping processes for signal sharing; and changing a user's location within the file system.

How does POSIX change MPE/iX?

In implementing the 1003.1 and 1003.2 functions, HP has enhanced the MPE/iX operating system. The main changes include

- Hierarchical file system (HFS)
- New file naming syntax (HFS syntax)
- Long file names
- Command enhancements
- Additional user environment (MPE/iX shell)
- Security enhancements (user identification ACD extensions)
- Byte-stream files (new record type)
- Symbolic links
- Enhanced process handling capabilities

These enhancements expand the current MPE file system structure. For example, prior to Release 4.5 and 5.0, the file system allowed you to create files under a group. As of Release 4.5 and 5.0, you can create HFS directories, as well as files, under groups, or accounts.

MPE has traditionally imposed constraints on naming syntax. Names could not exceed eight characters in length and could not contain special characters. MPE/iX now provides an expanded naming syntax, called HFS syntax, that allows for longer file names and has fewer constraints. HFS syntax is available for use in addition to the traditional MPE syntax.

What does POSIX offer you?

MPE/iX enhancements enable you to do the following:

- Port applications written to POSIX.1 standards.
- Use HFS directories to create multilevel data hierarchies.
- Provide case-sensitive names greater than eight characters in length.
- Rename and move files across account boundaries.
- Have a greater measure of control over files and directories on the system.

When MPE/iX Release 4.5 and 5.0 is installed, you automatically have access to POSIX functionality. No additional configuration is required for you to use many of the POSIX features described in this document.

If you are a general user on MPE/iX, your activities need not change. You can use the new features such as creating and using directories within groups accessible to you.

If you are a programmer, standard programmatic and interactive interfaces provide a standard development environment enabling source-sharing across platforms. This development environment allows you to implement open systems environment features.

If you are a system administrator, MPE/iX enables you to manage the system basically the same as you have for MPE/V or MPE XL.

Programmers using the MPE/iX system may want to use the *MPE/iX Developer's Kit (36430A)* to port open system applications onto MPE/iX. In this case, system administrators need to be aware of programming activities so that they can support them when managing the MPE/iX system.

Postponing the use of POSIX features

The introduction of POSIX features on MPE/iX places new demands on system managers. Today, system managers have a full workload without extra time to learn the new POSIX concepts. Chapter 3 discusses options available to those of you who may wish to postpone using POSIX features.

Hierarchical File System (HFS)

This section compares the traditional MPE file system with the enhanced MPE **hierarchical file system (HFS)**.

MPE/iX file system

Prior to Release 4.5, the MPE/iX file system includes accounts, groups, and files. This is the traditional MPE file system structure that you are used to.

Figure 2-1 shows an example of two accounts on an MPE/iX system: the `SYS` and `ACCT1` accounts. The `SYS` account contains the `PUB` and `GRP1` groups. The groups `PUB` and `GRP1` coexist as equals. `GRP1` cannot be a subgroup of `PUB` or vice versa. The `ACCT1` account has only one group, also called `PUB`. Files exist within the groups. The files cannot be located directly under the accounts. They can only exist in groups.

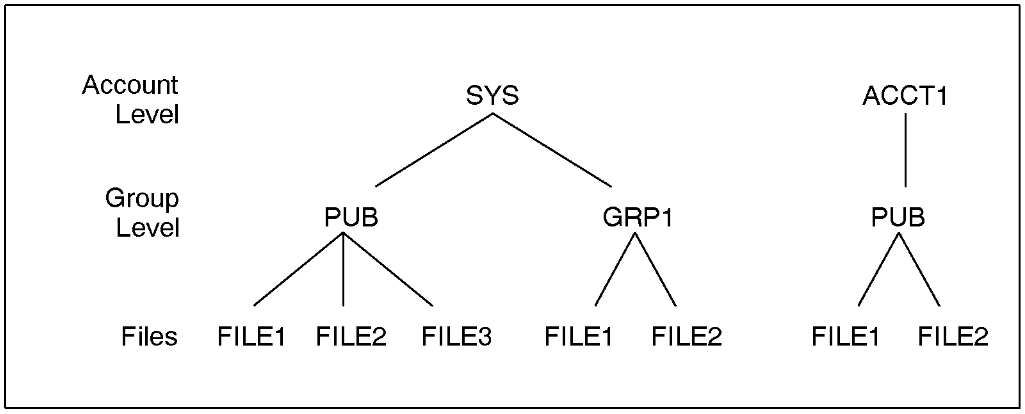


Figure 2-1. MPE/iX File System (Prior to Release 4.5)

MPE/iX hierarchical file system

Beginning with MPE/iX Release 4.5, the file system structure of MPE/iX is enhanced with the introduction of the MPE/iX hierarchical directory structure. This was accomplished by integrating the MPE/iX HFS structure within and around the traditional MPE file system. This provides the benefits of both structures to MPE/iX users without forcing them to choose one environment or the other.

MPE/iX hierarchical directory services present an integrated view of the file system. All file system components exist under one directory called the **root directory** (designated as **/**). Directories are structures that can contain files and other directories.

MPE/iX accounts and groups are considered to be special directories that serve as hierarchical directories while continuing to serve as traditional MPE accounts and groups. To ensure that the classic MPE file system view still exists, accounts can only be created under the root directory, and groups can only be created under accounts.

Figure 2-2 shows the MPE/iX file system structure introduced as of Release 4.5. In the figure, the boxed portion shows how the traditional account, group, and file system structure fits into the HFS structure. Notice that hierarchical directories and HFS files can now fall under traditional MPE groups. For example, the figure shows `lowercase_file` as being located in the `PUB` group of the `ACCT1` account. Also notice that the directory `dir3` falls under the `GRP1` group in the `SYS` account. The traditional MPE accounts, groups, and files can and do coexist with the HFS files and directories. As the figure shows, as of Release 5.0, files and directories can exist under accounts.

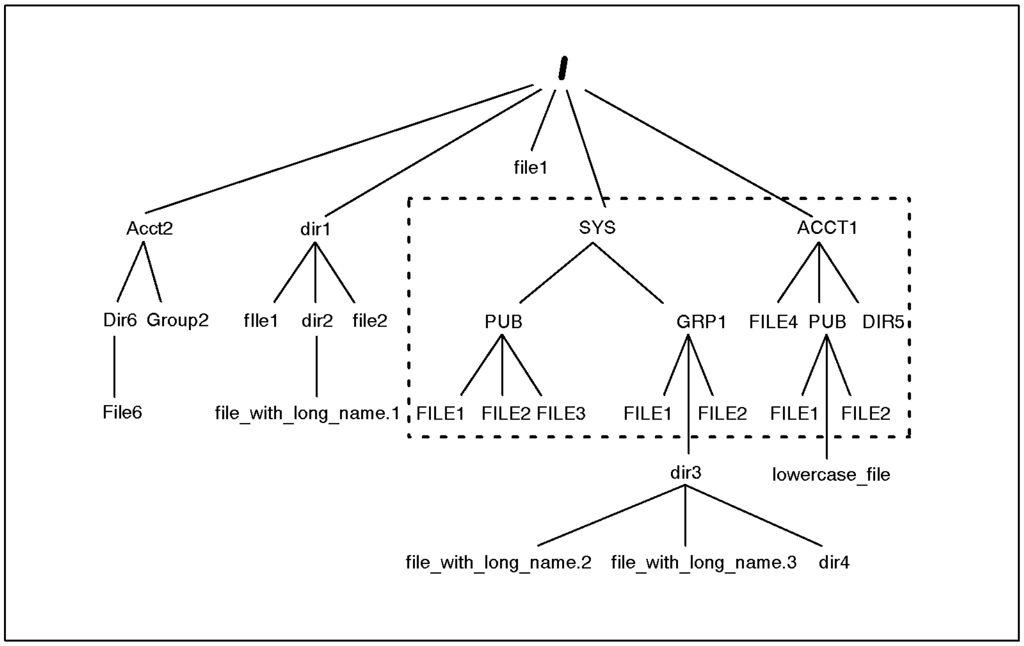


Figure 2-2. MPE/iX File System (Release 4.5 and Later)

Although accounts and groups are directories, they maintain their unique characteristics as well. This manual still refers to accounts and groups where necessary for clarity. When discussing characteristics of directories in general (including accounts and groups), this manual uses the term "directory." When referring specifically to a directory that exists outside of the traditional account-group structure, this manual uses the term "HFS directory."

Expanded File Naming Syntax

A name syntax is a set of rules that define the structure of valid names in that syntax. MPE/iX provides an expanded file naming syntax so you can refer to files, groups, accounts, and directories existing at all levels within the hierarchical file system. You can still name accounts, groups, and files by using the familiar MPE syntax. However, you'll need to use the expanded naming syntax, the HFS syntax, to refer to files or directories existing outside the traditional file system structure.

The following sections explain the differences between MPE syntax and HFS syntax.

MPE syntax

MPE syntax is the familiar syntax that is defined as follows:

- The file, group, account, and lockword names may each contain up to eight alphanumeric characters beginning with a letter. These names are upshifted by MPE, therefore, there is no case sensitivity.
- File names may have the form: *file[/lockword][.group[.account]][:envid]*. (The brackets indicate optional name components.)
- File names preceded with an asterisk are backreferences to file equations and have the form: *file[/lockword][.group[.account]][:envid]*. (The brackets indicate optional name components.)

- File names preceded with a dollar sign refer to special system-defined files.
- File, account, and group names cannot contain special characters.

The *envid* component in the file name specifies the remote environment and has two valid forms:

node[.domain[.organization]]

envname[.domain[.organization]]

Each *node*, *envname*, *domain*, or *organization* name can be up to 16 characters long and can include alphanumeric characters, the underscore (_), and the hyphen (-). The first character must be alphabetic.

Note You can only refer to lockwords, remote network environment (*envid*), file equations, and system-defined files (for example, `$STDIN`) by using the MPE syntax.

MPE pathnames

The traditional MPE pathname describes a file as `FILE.GROUP.ACCOUNT` where the file always comes first, the group always comes second, and the account always comes last. The following example shows a fully qualified MPE pathname for the file `MYFILE`:

```
MYFILE.PAYROLL.FINANCE
```

Only files in groups can be referred to using an MPE pathname. The group and account components of the name must refer to groups and accounts and not to hierarchical directories.

MPE name resolution

To ensure backward compatibility, file names are interpreted according to MPE syntax conventions and are qualified as follows:

- All characters are upshifted and treated as uppercase.
- Characters are validated against the accepted character set (A-Z, 0-9).
- File, group, and account name lengths are checked.
- Unqualified names are resolved relative to the current working directory. (Typically, your current working directory is your logon group so everything works as it has in the past. But, you can now change your current working directory to a hierarchical directory and still use MPE syntax to create, list, purge, etc. files and directories.)

HFS syntax

HFS syntax is similar to the file naming conventions used on open systems.

The syntax describes files by referencing the path or location leading to the file. For example, the file name `/SYS/PUB/CI` in HFS syntax is the same as `CI.PUB.SYS` in MPE syntax.

HFS syntax operates as follows:

- HFS file or directory names can include alphanumeric characters (A-Z, a-z, 0-9) and the following special characters:
  - dot (.)
  - underscore (_)
  - hyphen (-)
- File or directory names cannot begin with a hyphen (-).
- HFS syntax is **case sensitive**. So, HFS file or directory names can be in uppercase, lowercase, or mixed case.
- HFS directory or file names that are directly under the root or directly under a group or an account must be less than or equal to 16 characters.
- HFS directory or file names under HFS directories can be up to 255 characters long.

- Maximum pathname size is 1024 characters (including null terminator).

You can refer to traditional MPE file names using HFS syntax (as well as using MPE syntax). However, you must specify the name in uppercase. HFS syntax does not upshift the characters. For example, if a file named `BILLING` exists in the `PUB` group of the `MKTG` account (`BILLING.PUB.MKTG`), you can refer to it using HFS syntax as `./BILLING` if you are in the PUB group. You can also refer to it as `/MKTG/PUB/BILLING.` If you refer to the file as `./billing`, MPE/iX would not locate the file.

HFS pathnames

Files are referenced using pathnames. Pathnames consist of a list of name components. Except for the last name component, each name component in a valid pathname refers to a directory or symbolic link. The last name component may name either a directory, file, or symbolic link. The pathname's series of name components describes a path through the file system hierarchy.

HFS pathnames differ from MPE pathnames in the following ways:

- Names are separated with forward slashes (`/`), rather than dots.
- The order of the file name, group name, and account name are presented in reverse order compared to MPE syntax (`/ACCT/GROUP/FILE` versus `FILE.GROUP.ACCT`).
- **Slash** (`/`) at the beginning of a pathname indicates the root directory.
- **Dot-slash** (`./`) at the beginning of a pathname indicates the current working directory (CWD). The CWD is the directory in which you are currently working.

- Pathnames can be up to 1023 characters whereas traditional MPE names must be less than or equal to 26 characters (names can be up to 35 characters if a lockword is used). See Table 2-2 for CI restrictions.

Using these conventions, the format of the MPE pathname `MYFILE.PAYROLL.FINANCE` appears as follows in HFS syntax:

```
/FINANCE/PAYROLL/MYFILE
```

In this example, it is assumed that `MYFILE` is a file under the `PAYROLL` group and `FINANCE` account. However, `FINANCE` and `PAYROLL` need not necessarily be an account and a group as they must in MPE syntax. Using HFS syntax, `MYFILE` could be a file under the `PAYROLL` HFS subdirectory, which is under the `FINANCE` HFS directory, which is under the root directory.

Absolute and relative pathnames

Figure 2-3 presents an example using HFS pathnames. The example illustrates HFS directories and files using names of rivers, cities, and states.

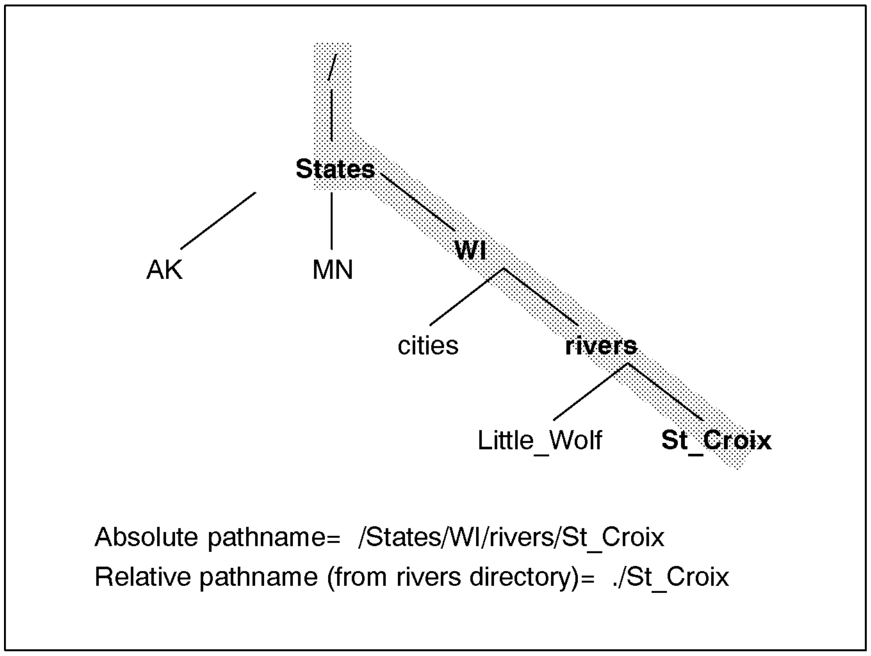

/
States
AK
MN
WI
cities
rivers
Little_Wolf
St_Croix
Absolute pathname= /States/WI/rivers/St_Croix
Relative pathname (from rivers directory)= ./St_Croix
LG200208_007

Figure 2-3. HFS Pathname Example

Files can be referenced using either absolute or relative pathnames. An **absolute pathname** begins with a forward slash (/) and is interpreted starting from the root directory. A **relative pathname** is anything without a leading slash and is interpreted starting from your current working directory. When working in the MPE/iX shell, relative pathnames need not begin with a dot-slash (./).

In Figure 2-3, the file `St_Croix` is located in the `rivers` subdirectory of the `WI` HFS directory. You can display the `St_Croix` file using its absolute pathname from anywhere on the system:

```
:LISTFILE /States/WI/rivers/St_Croix
```

If your current working directory is /States/WI, you can display the St_Croix file using its relative pathname:

```
:LISTFILE ./rivers/St_Croix
```

Relative pathnames are interpreted with respect to your current location on the system. For example, given the hierarchical directory structure shown in Figure 2-3, the above relative pathname only works if you are in the WI directory.

MPE-Escaped syntax is used by the CI and is the default for most intrinsics. It assumes MPE syntax, but allows the user to "escape" to HFS syntax by starting the name with a dot (.) or a slash (/), which are illegal first characters in MPE syntax.

Summary of MPE/iX syntax features

Maximum lengths of pathnames, numbers of characters in a name, and directory depth are increased as of Release 4.5 and 5.0. Table 2-1 summarizes the new limits that can be used when calling C library functions (available through the MPE/iX Developer's Kit) or intrinsics such as HPFOPEN.

Table 2-1.
Summary of MPE/iX Programmatic Interface Limits

Feature	HFS Syntax	MPE Syntax
Maximum directory depth	512 (`/1/2/3` ...)	3 (account, group, file)
Maximum no. of characters in a component	255	8
Maximum characters in a pathname	1023	35
File location	Under any root directory, /, account, or group	Under a group only
File referencing direction	Top-down (`/ACCT/GRP/file`)	Bottom-up (`FILE.GRP.ACCT`)
Pathname separators	Slashes (/)	Dots (.) and slashes (/) for lockwords
Case sensitivity	Yes (`FILE1` and `file1` are two different files.)	No (Lowercase automatically upshifts to uppercase.)

Table 2-2 summarizes the primary differences between HFS and MPE syntax as it operates in the command interpreter (CI). The CI imposes some constraints on component and pathname lengths, and maximum directory depth due to the length of the command buffer. (The term *component* refers to a file, account, group, directory, or fileset.)

Table 2-2. Summary of MPE/iX CI Limits

Feature	HFS Syntax	MPE Syntax
Maximum directory depth	512 (`/1/2/` ... `/512`)	3 (file, group, account)
Maximum number of characters in a component	Up to 255 for files or directories under HFS directories (Note: See Table 2-3 for more information.)	8 for accounts, groups, or files
Use of MPE syntax or HFS syntax	Initial slash (/) or dot (.) in the CI means use HFS syntax; only HFS syntax is used in the MPE/iX shell	Lack of an initial / or . in the CI means use MPE syntax
File referencing direction	Top-down (`/ACCT/GRP/file`)	Bottom-up (`FILE.GRP.ACCT`)
Pathname separators	Slashes (/)	Dots (.)
Case sensitivity	Yes (`FILE1` and `file1` are two different files.)	No (Lowercase automatically upshifts to uppercase.)

Table 2-2.
Summary of MPE/iX CI Limits (continued)

Feature	HFS Syntax	MPE Syntax
File location	Under any directory, /, account, or group	Under a group only
Maximum characters in pathname	See Table 2-3.	35 (8 times 4) +3
Lockwords	Not allowed	Allowed. `MEMO/A3` is a file named `MEMO` with a lockword called `A3`
Specifying remote environment	Not allowed	Remote environment specified using *envid*
File equations	(1) Allowed only on right side of equation; (2) Cannot backreference a file using HFS syntax	Allowed on both sides of the equation; backreferencing done by preceding name with asterisk (*)

MPE/iX commands are implemented in native mode or compatibility mode. The *MPE/iX Commands Reference Manual, Volumes I & II* (32650-60115) provides this information about each command. The maximum number of characters that each type of command can accept in a file name that is being used as a parameter to the command differs in native mode and compatibility mode. The maximum directory depth (`/1/2/3/4` ...) is the same for both modes.

Table 2-3 displays the maximum characters in a component name (such as a file or account name) and a pathname for natvie mode and compatibility mode commands. Refer to the *MPE/iX Commands Reference*

Manual, Volumes I & II (32650-60115) to learn in which mode each command is parsed.

Table 2-3. Native Mode vs Compatibility Mode Filename Lengths

Feature	Native Mode	Compatibility Mode
Maximum directory depth	512	512
Maximum characters in a component	255	■ 253 for relative pathnames (e.g., `./253chars`) ■ 254 for absolute names (e.g., `/254chars`)
Maximum characters in a pathname	279	■ 253 for relative pathnames (e.g., `./253chars`) ■ 254 for absolute names (e.g., `/254chars`)

New File Types

Release 4.5 and 5.0 of MPE/iX provide some new file types. Here is a brief definition of them. Since these interfaces are based on a commonly used standard, many textbooks are available that contain a detailed discussion of their use.

Byte-Stream Files

Byte stream files are simply a sequence (stream) of bytes. The term byte stream file is frequently used when talking about files with the byte stream record type, even though they are not a new file type. Byte stream files do not have any record structure associated with them and have a record size of 1 byte. Also, they allow reading and writing of data in arbitrary sized chunks.

Symbolic Links on MPE/iX

A symbolic link is a type of file that contains another pathname. It is a permanent file in the system directory. This concept is similar to MPE/iX *file equations*.

A symbolic link file may contain a relative or absolute path name. If a symbolic link to a relative path name is encountered during path name traversal, the contents of the symbolic link replaces the symbolic link component and is expanded into the path name being interpreted. If a symbolic link to an absolute path name is encountered, the contents of the symbolic link replaces all components up to and including the symbolic link and is expanded into the remainder of the path name.

For example, if the path name `/dir1/dir2/syml/file` is being traversed and the component *syml* is a symbolic link that points to the relative path `dir3/dir4`, then the final path name resolved is: `/dir1/dir2/dir3/dir4/file`. However, if `syml` contains the absolute path `/dir4/dir5`, then the final pathname resolved is: `/dir4/dir5/file`.

All symbolic links are interpreted in this manner except when the symbolic link is the last component of a pathname passed as a parameter to one of the following POSIX functions: `READLINK`, `RENAME`, `SYMLINK`, `UNLINK`, `CHOWN`, and `LSTAT`. With these calls, the symbolic link itself is accessed or affected.

Pipes

A *pipe* consists of two file descriptors connected such that data written to one can be read from the other in a first-in-first-out manner.

FIFOs

A FIFO is a *named pipe*. Because a FIFO has a directory name, it can be used by non-related processes to perform interprocess communication.

New and Enhanced Commands

As of Release 4.5, the MPE CI was extended to support POSIX enhancements. Several new commands are available and some existing commands have additional features.

New commands

Table 2-4 lists and briefly describes the new commands. For detailed information on each command, refer to the *MPE/iX Commands Reference Manual, Volumes I & II* (32650-60115).

Table 2-4. New MPE/iX Commands

Command Name	Description
`ALTFILE`	Alters a file's attributes such as the UID and GID of a file, directory, MPE group or account (except that the GID may not be altered for MPE groups or accounts).
`CHDIR`	Changes the current working directory to any directory, including MPE groups and accounts.
`DISKUSE`	Displays disk space usage, in sectors, for one or more directories or a directory subtree.
`NEWDIR`	Creates a hierarchical directory anywhere the user has permission.
`NEWLINK`	Creates a link to a file, group, account, directory, or root.
`PURGEDIR`	Deletes HFS directories (does not delete MPE groups or accounts).
`PURGELINK`	Removes a link.
`SETCLOCK`	Alters the system time or system time zone.
`SHOWCLOCK`	Displays information about the system date and time.

Enhanced commands

Table 2-5 lists and briefly describes each enhanced command. For detailed information, refer to Chapter 2 of the *MPE/iX Commands Reference Manual, Volumes I & II* (32650-60115).

Table 2-5. Enhanced MPE Commands

Command Name	Summary of Changes
ALTSEC	Changes access permissions for HFS directories.
ALTUSER	Incrementally adds or subtracts user capabilities and modifies the user ID (UID).
BUILD	Supports HFS syntax for permanent files. Supports byte-stream files.
COPY	Copies files with HFS names. You can copy such files to and from HFS directories or MPE groups.
FILE	Supports byte-stream files, and the *filereference* parameter has been enhanced to support HFS pathnames.
LISTACCT	Adds the FORMAT parameter to provide a choice of several display formats.
LISTEQ	Displays file equations with HFS pathnames.
LISTF	Several new formats have been added.
LISTFILE	Traverses an HFS directory tree.
LISTGROUP	Adds the FORMAT parameter to provide a choice of several display formats.
LISTUSER	Adds the FORMAT parameter to provide a choice of several display formats.
NEWACCT	Provides the GID and UID parameters to add a group ID (GID) to the group database and the user ID of the account manager to the user database.

Table 2-5.
Enhanced MPE Commands (continued)

Command Name	Summary of Changes
NEWUSER	Provides the UID parameter to add a user ID to the user database.
PLISTF (UDC)	Name changed from LISTF to PLISTF.
PRINT	Supports HFS file names.
PURGE	Supports HFS file names and wildcards.
RENAME	Supports HFS file names and renames across account boundaries.
RUN	Supports execution of HFS-named program files.
SAVE	Allows saving of HFS-named temporary domain files into permanent domain.
SHOWPROC	Shows processes that belong to a workgroup through the WG= keyword.
STORE	Stores HFS directories and files in HFS directories.
RESTORE	Restores HFS files and directories.
VSTORE	Verifies HFS files.

Enhanced Intrinsics

As of Release 4.5, many existing intrinsics have been enhanced to support new features of MPE/iX. Refer to Chapter 4 for a complete list of the enhanced intrinsics. The *MPE/iX Intrinsics Reference Manual* (32650-90028) fully describes the enhanced intrinsics.

System-Provided UDCs

Several UDCs are provided to simplify the use of the enhanced and new MPE/iX commands.

Several of the system-provided UDCs tailor the use of the `LISTFILE` command within the hierarchical file system. For example, a `PLISTF` UDC calls the `LISTFILE` command because the `LISTF` command does not list directories or files whose names are not consistent with the MPE syntax.

An additional UDC (`DISCUSE`) provides an alternate spelling of the `DISKUSE` command.

Table 2-6. System-Provided UDCs

UDC Name	Description
`DISCUSE`	Invokes `DISKUSE` to display disk space usage for one or more directories.
`FINDDIR`	Invokes `LISTFILE` to find a specified directory.
`FINDFILE`	Invokes `LISTFILE` to search for a specified file.
`LISTDIR`	Invokes `LISTFILE` to list directories.
`PLISTF`	Invokes `LISTFILE` to list files and directories.
`SH`	Invokes the MPE/iX shell.

Check with your system manager to determine whether or not these UDCs are activated on your system.

For additional information on using these UDCs, refer to Chapters 5, 6, and 7 of this manual. A complete description of each UDC is included in the *MPE/iX Commands Reference Manual, Volumes I & II* (32650-60115).

New and Enhanced Utilities

Two new utilities have been added to MPE/iX and `FSCHECK` has been enhanced.

New utility: PXUTIL

This new utility was created to maintain the POSIX-required user and group databases. When updating to or installing Release 5.0, MPE/iX automatically creates the user ID (UID) and group ID (GID) databases. The `PXUTIL` utility is available for version information and backdating capabilities.

A complete description of `PXUTIL` is included the *MPE/iX System Utilities Reference Manual* (32650-90081).

New backdating tool

The enhancements introduced as of Release 4.5 and 5.0, such as the use of hierarchical directories, files located outside groups, and byte-stream files all serve to complicate the backdating process. MPE/iX provides a program called `BDREPORT` to simplify backdating. Refer to Chapter 3 for a description of this program. Refer to the *HP 3000 MPE/iX Installation, Update, and Add-On Manual* (36123-90001) for information on backdating your system.

Utility enhancements

Table 2-7 summarizes enhancements to `FSCHECK`, `DEBUG`, and `FCOPY`. For detailed information on these utilities, refer to Chapter 3 of the *MPE/iX System Utilities Reference Manual* (32650-90081). Enhancements have also been made to `DISCUTIL` and `VOLUTIL`.

Table 2-7. Utility Enhancements

Utility Name	Summary of Changes

`DEBUG`	The following `DEBUG` commands now support HFS pathnames:
	`LIST` `STORE`
	`LOG` `SYMFILES`
	`MAP` `SYMOPEN`
	`MAPLIST` `SYMPREP`
	`REGLIST` `TXW`
	`RESTORE` `USE`
	The following `DEBUG` functions now support HFS pathnames:
	`mapsize`
	`mapva`
	The following `DEBUG` environment variables are available:
	`POSIX_FNAMES`
	`POSIX_OS`

`DISCUTIL`	`DISCUTIL` can be used to store HFS files and directories.

`FSCHECK`	The following `FSCHECK` commands now support HFS pathnames:
	`CHECKLABEL`
	`CHECKFILE`
	`DISPLAYEXTENTS`
	`PURGEFILE`
	`SYNCACCOUNTING`
	All other `FSCHECK` commands remain unchanged.

`FCOPY`	If a file exists in an HFS directory, you can use `FCOPY` to copy the contents of another file into it (if you have appropriate security permissions to do so).
`VOLUTIL`	`VOLUTIL` can be used to restore HFS files and directories from tape.

Computer-Based Training

You can learn about the new features of MPE/iX by using the POSIX Computer-Based training supplied with the MPE/iX operating system. The course is primarily for system managers who want to learn about how to manage the hierarchical file system.

MPE/iX Shell and Utilities

The POSIX 1003.2 standard is implemented on MPE/iX as the MPE/iX Shell and Utilities (commonly referred to as the **MPE/iX shell** or the **shell**). The MPE/iX shell is a user workspace that provides a set of commands and utilities in a UNIX®-like environment on MPE/iX. The MPE/iX shell is provided with the MPE/iX operating system. Refer to the *MPE/iX Shell and Utilities Reference Manual* (36431-90001) for complete information about all of the shell commands provided by the MPE/iX shell.

The MPE/iX shell is provided in addition to the MPE/iX command interpreter, not as a replacement for it. You have the option of using either the shell or the MPE command interpreter (or both) to do your work. You can invoke the shell from the CI by typing `SH` (the system-provided UDCs must be activated) or by executing `SH.HPBIN.SYS -L`. The UDC puts you into your login group and runs the shell. You can use shell commands at this point. Type `exit` to exit the shell and return to the CI.

Table 2-8 lists some tasks and associated MPE/iX shell commands that may be of interest to you. MPE commands, utilities, or features that are comparable to the shell commands are also listed. Not all of the shell commands operate in the same way as the MPE command or feature; some are rough equivalents.

Table 2-8. Selected MPE/iX Shell Commands

Task	Shell Command	Comparable MPE/iX Feature
Shorten command names using an alias	`alias`, `scripts`	command files or UDCs
Change working area	`cd`	`CHDIR`
Copy files	`cp`	`COPY`, `FCOPY`
Display all or part of text files	`cat`, `more`, `tail`	`PRINT`
Determine disk usage	`du`	`DISKUSE`
Search for text patterns in a file	`grep`, `egrep`, `fgrep`	Search commands in text editors
Terminate a process or job that's running	`kill`	`ABORTJOB`, `ABORT`
Rename files	`mv`	`RENAME`
Display current working directory	`pwd`	`SHOWVAR HPCWD`
Delete files	`rm`	`PURGE`
Delete directories	`rmdir`	`PURGEDIR`
Sort information	`sort`	Sort/Merge utility
Edit text or program files	`vi`, `ed`, `ex`	`EDIT/3000`
Display a list of who's logged on the system	`who`	`SHOWJOB`, `SHOWPROC`

MPE/iX Developer's Kit

Programmers who want to develop POSIX applications on MPE/iX or use POSIX functions can use the **MPE/iX Developer's Kit** (36430A). In addition, C library functions available through this kit provide a robust programming interface to many of the MPE/iX enhancements. The MPE/iX Developer's Kit is a separate product that must be purchased specifically. It includes the following:

- POSIX.1 Library Functions
- HP C Compiler
- HP C Library Functions

The MPE/iX Developer's Kit includes the following documents:

- *MPE/iX Developer's Kit Reference Manual, Volumes I and II* (36430-90001 and 36430-90002) describe programmatic interfaces and extensions to the C library.
- *MPE/iX Shell and Utilities User's Guide* (36431-90002) provides a command overview and tutorials on the MPE/iX shell and describes utilities such as make, yacc, lex, and awk.
- *MPE/iX Shell and Utilities Reference Manual, Volumes I and II* (36431-90001 and 36431-90003) contain complete descriptions of all MPE/iX shell commands.
- *HP C/iX Reference Manual* (31506-90005) provides a complete reference for the C programming language.
- *HP C Programmer's Guide* (92434-90002) includes information on programming in C.
- *HP C/iX Library Reference Manual* (30026-90001) describes C library routines.

- *The POSIX.1 Standard: A Programmer's Guide* (36430- 90003) by Fred Zlotnick, Benjamin/Cummings Publishing Company, Inc., 1991. A programmer's guide to using POSIX.2 C library functions to create portable programs.

3

What's New for System Administrators?

This chapter provides conceptual information about what's new as of Release 4.5 and 5.0 specifically for system administrators. You should read and be familiar with Chapter 2 before reading this chapter.

This chapter includes the following sections:

- System management considerations
- POSIX computer-based training
- Expanded search capabilities
- Backup and restore
- UDCs, JCL, command files, and programs
- HFS directories and high-availability products
- Accounting
- Applications
- POSIX-compliant applications
- Postponing POSIX
- User and Group IDs
- File Ownership
- User and group databases
- MPE/iX security components
- Backdating

System Management Considerations

System administrators should be aware of the following considerations concerning MPE/iX at Release 4.5 and 5.0:

Syntax

- The `HPPATH` system variable does not accept HFS pathnames. Consequently, you cannot include HFS pathnames in the default search path for a session.
- Remote file access does not support HFS syntax.

Directories

- The following types of files can be located only within MPE groups; they cannot be located within HFS directories:
  - UDC files
  - Databases
  - Compatibility mode files (for example, CIR, KSAM, RIO)

Security

- ACDs are automatically assigned to HFS directories and files residing within those directories. Also, directories and files in root or accounts.
- You cannot assign ACDs to the root directory, MPE accounts, or MPE groups.
- Successful logon for users without SM capability requires valid UID and GID entries in the user and group databases.
- By default, any user can create files in the `/tmp` directory. This directory is automatically created when you update to Release 4.5 or 5.0. While this behavior is desirable for POSIX-based applications, some system managers may want to restrict access to this directory.

Commands

- Frequently used MPE commands directly support HFS syntax, some do not.
- The `HELLO` and `JOB` commands only log on to a group.
- The `FILE` command allows HFS pathnames only on the right side of the equation.
- You should use the `DISKUSE` command instead of the `REPORT` command to report on disk usage of HFS directories that exist outside groups and accounts.

Groups

- Databases are supported only in MPE groups.
- Compatibility mode files can reside only in MPE groups.

Accounting

- MPE account or group-level accounting does not extend to directories residing outside of accounts.

System management tasks remain essentially the same with Release 5.0. Tasks such as system startup and shutdown, system configuration, system logging, adding user accounts and groups all work the same with some minor enhancements to access HFS features. By default, existing MPE/iX commands and intrinsics assume MPE syntax. You must deliberately activate new features to use them.

POSIX Computer-Based Training

Due to all of the changes to MPE/iX relating to the addition of POSIX functionality, a computer-based training course is available online. The course is primarily for system managers who need to manage MPE/iX systems which are updated to Release 5.0.

The training is automatically installed in the SYS account when you update to Release 5.0. To take the training, you need to log on to your system and type the following:

```
:POSIXCBT.LSN.SYS
```

Expanded Search Capabilities

MPE/iX has enhanced certain commands, such as `LISTFILE`, to support pattern matching for HFS names as well as for traditional MPE names. Table 3-1 shows examples of resolutions to various wildcard combinations that use standard MPE syntax or HFS syntax. The examples in the table are case sensitive.

Table 3-1. Examples of Wildcard Resolutions

Filename Specified	Pattern Matched
`@`	Zero or more occurrences of A-Z, 0-9.
`[ab]`	One occurrence of A or B.
`[a-c]`	One occurrence of A, B, or C.
`./@`	Zero or more occurrences of: a-z A-Z 0-9 Dot (.) Underscore (_) Hyphen (-)
`./[aB]`	One occurrence of a or B.
`./[a-c]`	One occurrence of a, b, or c.
`./@[-ac]`	Name ending in -, a, or c.

The use of a trailing `/` at the end of a pathname or use of the `TREE` option in commands such as `LISTFILE` and `STORE` provides another expanded search tool. This option causes the command to recursively search all directories at all levels under the last directory specified in the pathname. For example, the following command lists all files at all levels under the `/SYS` account:

```
LISTFILE /SYS/
```

You can further delimit your search by using the `NAME` parameter with the `LISTFILE` command. The following example displays all files, groups, and directories under the `SYS` account that begin with `OFF`:

```
LISTFILE /SYS/;NAME=OFF@
```

In addition, the following UDCs are provided to simplify searching for files and directories within the hierarchical file system:

- The `FINDDIR` UDC finds a specified directory.
- The `FINDFILE` UDC finds a specified file.

Refer to Volume 1 of the *MPE/iX Commands Reference Manual* (32650-90003) for details on using the UDCs.

Backup and Restore

MPE/iX provides the STORE/RESTORE facility and TurboSTORE II/iX, which enable you to back up files and transfer files from one MPE system to another. Both of these facilities allow you to back up and restore HFS directories and files.

As of Release 5.0, `STORE @.@.@` stores all files (and HFS directories). You can also use the following command to back up all files:

```
STORE /
```

The command backs up all accounts, groups, directories, and files on the system. In the command, the `/` does three things:

1. It specifies the root directory.
2. It tells MPE/iX to look for names using the HFS syntax (which includes all MPE syntax names).
3. It calls the `TREE` option which includes all files under the directory specified.

Note

If you use a third-party program to back up your system, you need to be sure that it is capable of storing files in HFS directories and subdirectories.

UDCs, JCL, Command Files, and Programs

Existing UDCs, JCL, command files, and programs function on MPE/iX as they do on MPE XL. By default, existing MPE XL commands and intrinsics take MPE syntax rather than HFS syntax.

MPE/iX also provides several useful UDCs listed in the section called “System-Provided UDCs” in Chapter 2. These are located in `HPPXUDC.PUB.SYS`. You need to activate these UDCs using `SETCATALOG` as you would other UDCs.

For example, to activate the UDCs system-wide without overriding any other UDCs you have set up:

```
:SETCATALOG HPPXUDC.PUB.SYS; SYSTEM; APPEND
```

You must have SM capability to use this command.

Note

If applying the `HPPXUDC.PUB.SYS` catalog, realize that there may be an impact on third-party UDCs.

HFS Directories and High-Availability Products

HFS directories under MPE groups can be located on non-system volume sets. But, HFS directories under the root and accounts must reside on the system volume.

High-availability products, such as Mirrored Disk, Autorestart, and SPU Switchover, are restricted to non-system volume sets; therefore, they cannot support HFS directories under the root or accounts. These products only support HFS directories as descendants of MPE groups.

As a workaround, symbolic links can be used to “redirect” directories from under root or an account to a group that is located on a user volume set.

Accounting

MPE/iX allows system managers to limit the amount of disk space that a user may be allocated. Disk space limitations can only be placed on MPE/iX accounts and groups. However, a limit placed on an account or group is also imposed on all hierarchical directories and files created at all levels beneath that account or group.

Users may create files outside their logon account if granted the proper access to do so. Disk usage is still accumulated to the parent account or group, regardless of who creates the file.

Users can use `chown()` in the MPE/iX shell to change the group ID of files they created outside their logon accounts to their logon account's group ID. This puts the file in the administrative domain of the user's account manager rather than in the domain of the account manager for the account in which the file is located. The disk space for the file accrues to the account in which it resides. Account managers may need to cooperate with each other or with the system administrator to manage disk space when files are being created across account boundaries. See Chapter 9 for additional information on user identification and security.

No accounting limits exist for files or directories that don't reside below MPE groups or accounts. Also, because the `REPORT` command does not show disk usage outside of the MPE account structure, you need to use the `DISKUSE` command to do so.

Note It is suggested that access to root and directories under root be restricted.

Applications

Because MPE/iX is backwards and forward compatible, the behavior of existing MPE applications will remain the same on systems that do not take advantage of the hierarchical file system.

POSIX-Compliant Applications

You can run POSIX-compliant applications on MPE/iX. Follow the application instructions for installing and running the application.

Postponing POSIX

If you run third-party software that is not POSIX-compliant and you have no immediate interest in POSIX features, you have several options:

1. You can update to Release 5.0 and take advantage of the new `:SETCLOCK` command or wildcarded `:PURGE`, but not use the POSIX extensions.

   If most of your users are trapped into applications at logon, they will not notice any difference. More importantly, they cannot create any HFS objects, which means the system manager does not really need to be POSIX knowledgeable.

   You can take comfort in knowing that users need to take deliberate action to build a POSIX-named object. Remember that the command `:BUILD a` still creates the file uppercase "A" in your logon group (as long as you have not changed your CWD). Production jobs and programs will behave the same.

   You can limit access to the shell and utilities by adding ACDs to all program files in the `HPBIN.SYS` group. You can also restrict the use of the .2 shell by adding a lockword.

2. You can update to Release 5.0 and purge all of the FOS HFS files and the `HPBIN.SYS` group. This will save you some disk space, but will not prevent a user from creating an HFS object.
3. You can update to Release 5.0 and buy a commercially available software product that disables the HFS-related features of MPE/iX.

User and Group IDs

Each MPE/iX user has an associated **user ID (UID)**. The UID is a string (in the form *user.account*) and a corresponding integer value. Additionally, one or more users can be organized into groups (distinct from MPE groups) to simplify file sharing. Each group has an associated **group ID (GID)**. The GID is a string (in the form *account*) with a corresponding integer value.

UIDs and GIDs are used in conjunction with other security mechanisms to control access to objects. **Objects** are entities that contain or receive information, such as files, directories, and devices.

A **process** is a program that is currently being executed. Every process has a UID, one or more GIDs, an optional file mode creation mask (specifies which permissions should not be granted), and a set of capabilities (for example, SM, AL, GL, ND, SF, etc.). MPE/iX Release 5.0 only supports one GID per process.

When files or directories are created, they are assigned their parent directory's GID and the UID of the process creating them. File and directory UIDs and GIDs can be changed using the `chown()` function available through the MPE/iX shell.

File Ownership

Prior to Release 4.5, MPE has used the creator name, an unqualified user name, to track file ownership. The system only recorded file creators (not the creators of accounts or groups). As of Release 4.5, files and HFS directories can be created outside the logon account if the user has the appropriate access to do so. Therefore, unqualified user names are no longer sufficient for indicating object ownership.

As of Release 4.5, file ownership for all newly created files or renamed files is indicated by a fully qualified user name. The fully qualified user name is called the **file owner**. The concept of the file owner is similar to the file creator. Note however that the file creator could never be changed for a file whereas the file owner can be changed.

User and Group Databases

UIDs and GIDs are stored in two databases: `HPUID.PUB.SYS` holds UIDs and related user information in a **user ID database**, and `HPGID.PUB.SYS` holds GIDs and related information in a **group ID database**.

These databases are privileged files that are transparent to users. The databases are automatically updated when you add, change, or delete users on the system by using the following commands:

```
ALTACCT
ALTUSER
NEWACCT
NEWUSER
PURGEACCT
PURGEUSER
```

The system automatically creates and maintains the user and group databases when you update to Release 5.0.

The complete user and group databases can be rebuilt using the `RESTORE` command with the `DIRECTORY` option. Refer to the section "Reloading Hierarchical Directories" in Chapter 8 for more information.

MPE/iX Security Components

This section describes some existing security features then introduces security enhancements and their implications for system administrators.

Access control definitions

MPE/iX continues to support **access control definitions (ACDs)**. ACDs are ordered lists of pairs (access permissions and user specification) that specify access to objects. An ACD takes precedence over certain other security features, such as lockwords and the file security matrix.

Files located outside of MPE groups and HFS directories are automatically assigned ACDs when they are created. By default, RACD (read ACD) is assigned to all users and only the owner can access the file or directory. The ACD can be modified using the `ALTSEC` command but the ACD cannot be deleted.

When files are renamed to a group outside the original account, they are automatically assigned ACDs. When a file located in an MPE group has its group ID (GID) changed to the GID of another account, an ACD is automatically assigned. The ACD can be modified using the `ALTSEC` command but it cannot be deleted.

If you are unfamiliar with ACDs, refer to Chapter 9 "Handling Security on MPE/iX" of this manual. For more information on manipulating ACDs, refer to the `ALTSEC` command in the *MPE/iX Commands Reference Manual, Vol. I (32650-90003)*.

Access modes

ACD pairs control the access and manipulation of HFS directories and the files within them. MPE/iX has enhanced ACDs to support four new ACD **access modes**. The ACD access modes are as follows:

Permissions common to files and directories

RACD	Copy or read the ACD.
NONE	Deny access.

File permissions

R	Read a file.
W	Write to a file.
L	Lock a file.
A	Append to a file.
X	Execute a file.

Directory permissions

CD	Create directory entries.
DD	Delete directory entries.
RD	Read directory entries.
TD	Traverse directory entries.

User specifications

The following new ACD user specifications are provided:

- `$OWNER` specifies users whose UID maches the file owner of the object. `$OWNER` enables file owners to voluntarily limit their access to an object. For example, file owners can grant themselves read-only access to a file to guard against accidentally modifying the file. The `$OWNER` user specification is the only way for file owners to limit their access to an object.

- `$GROUP` specifies users with a GID that matches the current group ID of the object. `$GROUP` permits dynamic reference to the GID of an object. This is useful because GIDs of files and directories can be changed programmatically or using `chown` in the MPE/iX shell. When the GID of a file is changed, it is not necessary to modify an ACD to correct file sharing.
- `$GROUP_MASK` restricts the access granted by ACD entries other than `$OWNER` and `@.@`. When an ACD contains a `$GROUP_MASK` entry, a user is granted a specific access mode only if it is listed in the ACD entry the user matches (in the form *user.account*, *@.account*, and `$GROUP`) and in the `$GROUP_MASK` entry.

You can use traditional user specifications to describe individuals or groups of users:

- *username.accountname* specifies a single user
- *@.accountname* specifies all users associated with the *accountname* account.

Capabilities

SM and AM capability are checked before ACDs or the file access matrix. Users with SM capability have unrestricted access to all file system objects.

Users with SM capability can create files outside of the logon account/group structure because they have implied CD access. Those without SM capability can only create files in directories where they explicitly have CD permission. Users must also have SF capability to save files in directories and SAVE access to save files in an MPE group.

Account managers may not have total access to all objects in their account. Having AM capability enables a process to access file system objects if the GID of an object (GID represented by an account name) matches the GID (logon account) of the process. What this means is that there may be cases where the GID of a file or directory within an account has been changed programmatically or using `chown()` in the MPE/iX shell so that an AM for that account cannot access it, or the file or directory was created by a user with a different GID.

Lockwords

A file's creator can assign or remove a file lockword. Lockwords can only be assigned to files, not to directories. Lockwords can only be assigned to files in MPE groups.

All users are required to provide lockwords for files protected by active lockwords. Lockwords must be supplied by being embedded in MPE syntax file names or in response to lockword prompting.

There is no way to specify a lockword using HFS syntax. Any attempt to open a file with a lockword using HFS syntax results in a lockword violation. The user is not prompted for the lockword.

Although system managers can assign ACDs to any file or directory in the system, they must supply the lockword for any lockword-protected files before they can assign an ACD. Once the file has an ACD, the ACD supercedes the lockword.

Restricting Access to /tmp

Because any user can build files in `/tmp`, you can restrict access by using the `ALTSEC` command.

Creating files and directories

Users can create files or directories in any HFS directory which they can traverse and to which they have been granted create directory entries access. Only users with SM capability can create files in MPE groups outside the logon account or in the root directory. Users can create files in MPE groups in their logon account and in other groups where they have SAVE access. A user must have SF capability to create a file or directory. The MPE group must have SAVE access assigned to it before files and directories can be created at any level under it.

Renaming files and directories

Users with sufficient access can rename or move a file between directories. The file's creator is no longer the only user able to do this. Only the file creator can perform a rename operation if the lockword of the file is being changed.

To rename a file across directory boundaries, a user must have DD access to delete the old directory entry and CD access to create the new directory entry. TD access is also needed to all name components that make up the source and target pathnames. SAVE access is required to rename a file from an MPE group in one account to an MPE group in another account.

Deleting files or directories

To delete files or directories in HFS directories, users must have DD access to the parent directory. During the design of the HFS, it was decided that any user that could create a directory in a group should also be able to purge that directory. Since all that is required to build a directory is SAVE access to the group, that is also what is required to purge the directory. The directory must be "empty", meaning that it contains only the . and .. entries. For historical compatibility, to delete files in MPE groups, users must have WRITE access to the file.

Backdating

Occasionally it is necessary to reinstall an earlier version of MPE/iX after having updated a system to the latest release. This is called *backdating* a system. The *HP 3000 MPE/iX Installation, Update, and Add-On Manual* (36123-90001) provided with the MPE/iX software includes the information required for backdating a system.

Note

Backdating a system is not a trivial task and should be done only if absolutely necessary. You should contact the Response Center before backdating. Response Center engineers may be able to help you solve the problem without having to backdate the system.

The introduction of the hierarchical file system at Releases 4.5 and 5.0 allows for the possible presence of hierarchical directories, files located outside groups, byte-stream files, FIFO files, symbolic link files, and device link files on MPE/iX systems. These new features could make it difficult to backdate MPE/iX Release 5.0 to an earlier release and could prevent the earlier version of MPE/iX from operating correctly.

MPE/iX provides a program called `BDREPORT` to simplify backdating. You must back up the entire system before running this program and backdating your system.

`BDREPORT` identifies any operating system incompatibilities (such as files and HFS directories existing outside of MPE groups) and produces two script files called `BDSCRP1` and `BDSCRP2`. The script files contain the commands necessary to delete files and directories that would not be compatible with an earlier release of MPE/iX.

Once the incompatibilities are identified, you can store the files and directories onto tape. You can then execute the scripts to correct other incompatibilities, such as files whose file labels, ACDs, or transaction management logs that are not compatible with the earlier release.

Executing the `BDSCRP1` script file does the following:

- Purges the incompatible files and directories
- Modifies file labels if required
- Backdates transaction management for all mounted user volume sets

Executing the `BDSCRP2` script does the following:

- Stores the directory structure
- Backdates the user and group databases using `PXUTIL`
- Backdates the system volume set transaction management logs

After the incompatibilities are identified and handled using this process, you can backdate the system to the earlier release. You will then need to restore the user information that was backdated and any files you want to retain from the STORE tape.

4

What's New for Programmers?

This chapter provides application developers with information about new MPE/iX features available through MPE/iX system intrinsics.

Summary of New and Enhanced Features

This section provides a quick summary of new and enhanced features accessible through MPE/iX system intrinsics. Refer to later sections in the chapter for more detailed information about these features.

New features

- Hierarchical directory files
- MPE/iX byte-stream files
- Renaming a file across account boundaries
- Append mode (versus append access)

Enhanced features

- Security features
- Object ownership
- Closing a file
- Listing file information
- Terminating file name strings
- Accessing a file by its file name
- The intrinsics listed in Table 4-1.

Table 4-1. Enhanced MPE/iX Intrinsics

Intrinsic Name	Summary of Changes
FCHECK	Provides additional file system error codes.
FCLOSE	Supports hierarchical directories and byte-stream files.
FCONTROL	Supports hierarchical directories and byte-stream files.
FFILEINFO	Enhanced and new item numbers return information about hierarchical directories and byte-stream files. Supports HFS syntax.
FGETINFO	Enhanced parameters return information about byte-stream files but only limited information about hierarchical directories. Does not support HFS syntax.
FLABELINFO	Enhanced and new item numbers return information about hierarchical directories and byte-stream files. Supports HFS syntax.
FLOCK	Supports byte-stream files. Hierarchical directories cannot be locked.
FOPEN	Supports HFS syntax. Creates and opens byte-stream files and opens hierarchical directories. Cannot create hierarchical directories.
FPARSE	Does not support HFS syntax.
FPOINT	Supports byte-stream files.
FREAD	Supports byte-stream files.
FREADDIR	Supports byte-stream files.
FREADLABEL	Supports byte-stream files.

Table 4-1.
Enhanced MPE/iX Intrinsics (continued)

Intrinsic Name	Summary of Changes
`FREADSEEK`	Supports byte-stream files.
`FRENAME`	Supports HFS syntax. Supports renaming of files across account boundaries.
`FSPACE`	Supports byte-stream files.
`FWRITE`	Supports byte-stream files.
`FWRITEDIR`	Supports byte-stream files.
`FWRITELABEL`	Supports byte-stream files.
`HPACDINFO`	Supports HFS syntax. Provides new and enhanced security features.
`HPACDPUT`	Supports HFS syntax. Provides new and enhanced security features.
`HPCICOMMAND`	Supports up to 512 chars in command buffer.
`HPDEVCONTROL`	Supports setting data compression.
`HPDEVCREATE`	Allows creation of FIFO and device link files.
`HPFOPEN`	Supports HFS syntax. Creates and opens both hierarchical directories and byte-stream files.
`HPPIPE`	Allows the creation of pipe files.
`PRINTFILEINFO`	Supports HFS syntax. Supports both hierarchical directories and byte-stream files.
`STACKDUMP`	Limited support of HFS syntax. Special considerations for file name terminating characters.

Hierarchical Directory Files

The following sections provide information about hierarchical directory files of interest to application developers who wish to use MPE/iX system intrinsics to manage their applications within the hierarchical directory structure.

The traditional MPE directory structure

A directory is a repository of information about objects on a computer system. Objects can be either files or directories. Directory services are a set of routines that manipulate directory objects and retrieve information about the objects found in a directory.

Prior to MPE/iX Release 4.5, the MPE/iX directory structure organized the file system as a four-level hierarchy:

- The root directory. This directory was never exposed to the user.
- Accounts, or account directories, immediately under the root directory. Only group directories were allowed under account directories.
- Groups, or group directories, at the level immediately under account directories. Only files are allowed under group directories.
- Files at the level immediately under group directories.

This traditional MPE directory structure was implemented to meet the demands of the commercial business community and continues to serve as an excellent directory structure.

Beginning with MPE/iX Release 4.5, the directory structure has been greatly enhanced with the introduction of the MPE/iX hierarchical directory structure. The challenge Hewlett-Packard faced was to provide an “open” directory structure while preserving the traditional MPE directory structure.

This has been accomplished by successfully integrating MPE/iX hierarchical directory structure within and around the traditional MPE directory structure, thus providing the benefits of both directory structures to existing and new users.

Note

All existing applications provided on HP 3000 Series 900 computer systems continue to function exactly as they have in the past. In the future, more and more applications will be available that take advantage of the new MPE/iX hierarchical directory structure. This includes both POSIX-conformant applications as well as existing applications enhanced to take advantage of MPE/iX "Open" features.

MPE/iX hierarchical directory structure features

The new MPE/iX hierarchical file system directory services organize the file system as a file hierarchy. The HFS file system hierarchy conceptually resembles a tree. In the hierarchy, files are grouped together as leaf nodes in name contexts called directories.

Files and directories are referred to using a pathname that describes their location in the file system hierarchy relative to either a process's root directory (absolute pathname) or current working directory (relative pathname).

A process's current working directory may change during the lifetime of a process. Files may have exactly one pathname relative to the system's root directory (the absolute pathname). Each file name reference is known as a link.

MPE/iX hierarchical directory services present an integrated view of the file system. There exists a common root for all directories (exposing the traditional MPE root directory to users).

Traditional MPE accounts and groups are considered in the HFS context to be directories. However, they are special directory structures that include hierarchical directory behavior while continuing to serve within the MPE framework.

Following are special features of the MPE/iX root directory and MPE/iX account and group directories to ensure both backward compatibility and forward benefit.

Root directory features

- The MPE/iX root directory (/) cannot be renamed, copied, or purged.
- Only users with SM capability can create objects directly under the root directory.
- Access permissions for the root directory are RD (read directory) and TD (traverse directory) for all users and CD (create directory entries) and DD (delete directory entries) for none. Attempts to remove or change access permissions of the root directory result in an error.
- Names of files and hierarchical directories created directly under the root directory are restricted to 16 characters in length.
- The root directory is restricted to the MPE/iX system volume set.
- The root directory does not contain explicit dot (.) and dot-dot (..) directories; however, dot and dot-dot directory behavior is supported.
- The user ID (UID) and group ID (GID) associated with the root directory cannot be modified.

Account directory features

- MPE/iX accounts can be created directly under the root directory only by a user with SM capability using the MPE/iX `NEWACCT` command. MPE/iX accounts cannot be created, renamed, copied, or purged through the MPE/iX shell.

- GID = *account name*
- Access permissions for an MPE/iX account are RD & TD for all users and CD & DD for none. Attempts to use the `ALTSEC` command or `chmod` MPE/iX shell command to remove access permissions of an MPE/iX account result in an error.
- When an MPE/iX account name is a component in an HFS pathname, it must be specified in uppercase.
- MPE/iX accounts are restricted to the MPE/iX system volume set.
- MPE/iX accounting limits for disk space apply to both hierarchical directories and files located at all levels under MPE/iX groups.
- An MPE/iX account does not contain explicit dot and dot-dot directories; however, dot and dot-dot directory behavior is supported.
- The user ID (UID) and group ID (GID) associated with an MPE/iX account cannot be modified.
- Names of files and directories created under accounts are limited to 16 characters.

Group directory features

- MPE/iX groups can be created directly under MPE/iX accounts only by a user with SM capability or a user with AM capability who is a member of that account (whose GID matches the GID of the account), using the MPE/iX `NEWGROUP` command. They can be modified using the `ALTGROUP` command. MPE/iX groups cannot be created, modified, or purged through the MPE/iX shell.

- Default access permissions for MPE/iX groups are RD & TD for all users and CD access for none. Attempts to remove access permissions of an MPE/iX group result in an error. An MPE/iX group must have MPE/iX save access assigned to it before files and directories can be created at any level under it. For more information about MPE/iX save access, refer to the manual *Manager's Guide to MPE/iX Security* (32650-90474).
- When an MPE/iX group name is a component in an HFS pathname, it must be specified in uppercase.
- GID inheritance from `ACCOUNT`.
- Names of files and hierarchical directories created directly under MPE/iX groups are restricted to 16 characters in length.
- MPE/iX groups (and, indirectly, all files and hierarchical directories at all levels under them) can optionally be assigned to a user volume set.
- MPE/iX accounting limits for disk space apply to both hierarchical directories and files located at all levels under MPE/iX groups.
- An MPE/iX group does not contain explicit dot and dot-dot directories; however, dot and dot-dot directory behavior is supported.
- The user ID (UID) and group ID (GID) associated with an MPE/iX group cannot be modified.

Hierarchical directory features

- Hierarchical directories cannot be copied. Hierarchical directories can be purged only when they are empty (containing only the dot and dot-dot directory entries).
- HFS directories can be renamed through the POSIX.1 `rename` function, or through the shell `MV` command.

The renaming of a directory must be within the same disk space accounting domain.

- Users can define and modify access permissions for hierarchical directories.
- The group ID (GID) of a hierarchical directory is inherited from its parent directory. The user ID (UID) of a hierarchical directory is inherited from the user who created it. The UID and GID of a hierarchical directory can be modified using the `ALTSEC` command or the MPE/iX shell `chown` command.
- Hierarchical directories (and all objects under them) that are not under MPE/iX groups are restricted to the MPE/iX system volume set.
- Names of hierarchical directories and files located directly under either the system's root directory or MPE/iX accounts or groups are restricted to 16 characters in length.
- Names of files and hierarchical directories created directly under a hierarchical directory are restricted to 255 characters in length.
- Hierarchical directories contain explicit dot and dot-dot directory entries.

MPE/iX hierarchical directory services can access any file in the file system hierarchy including files in the traditional MPE directory structure using HFS syntax. For example, both of the following fully qualified file name specifications refer to the same file.

```
MYFILE.MYGROUP.MYACCT

/MYACCT/MYGROUP/MYFILE
```

Hierarchical directory files

Beginning with MPE/iX Release 4.5, the directory file type is available to allow an MPE/iX file to reproduce POSIX directory file behavior.

The POSIX standard defines a directory to be a special file that contains directory entries of all files and directories located directly beneath it. Users can create directories and read directory entries but cannot directly write to directory files. Instead, the system creates a new directory entry each time a file is created and placed in that directory, and it removes the entry when the file either purged or moved to a different directory.

Note

In this manual, the term “hierarchical directory” is used to refer to an MPE/iX disk file whose file type is directory, and its record type is the hierarchical directory record type.

Creating a hierarchical directory

You use the `HPFOPEN` intrinsic to create a hierarchical directory. The `FOPEN` intrinsic can open an existing hierarchical directory but cannot create one because the *foptions* parameter cannot be extended to specify a file type of directory or a record format type of hierarchical directory.

The following table defines the file and access attributes imposed on a hierarchical directory when it is created. These attributes cannot be changed during the life of the hierarchical directory.

Table 4-2. HFS Directory File Attributes and/or Access Options

Attribute Name	Default
File type	Directory (DIR) disk file. Must be explicitly specified in order to create a hierarchical directory.
Dynamic locking	Disabled. Any value other than `0` will result in error.
Exclusive	Shared. Any attempt to open with exclusive `access(1)` will result in error.
Nowait I/O	Disabled. Any attempt to open `nowait(1)` will result in error.
Record format	Hierarchical directory. All other specifications are ignored.
Domain	Create as a permanent file. All other specifications are ignored.
Carriage control	Specifying `cctl` will result in error.
Access type	Directory read access. Must be explicitly specified to create a hierarchical directory.
Multirecord	Multirecord access (MULTI). All other specifications are ignored.
ASCII/binary	ASCII. All other specifications are ignored.
Remote environment	Remote access to hierarchical directories is not supported. Specifying this option results in an error.
File size	Two gigabytes. Specifying a file size will result in error.
Copy mode	Ignored.
Record size	32 bytes. Any value specified will be ignored.
File code	Specifying any value will result in error.
Number of buffers	Ignored.
User labels	Not allowed. Attempting to specify this option results in an error.
File equations	Only applicable with MPE namespace directories.

The dot (.) and dot-dot (..) directories

When a hierarchical directory is first created, two special directory files are placed in the directory:

- The **dot (.)** directory entry is an alternative way to specify the directory itself without having to use a formal directory file name.
- The **dot-dot (..)** directory is an alternative way to specify the directory's parent directory without having to use a formal directory file name.

The dot and dot-dot directories provide additional navigation aids to a process. Applications using the dot and dot-dot directories to express a directory or parent directory need not be concerned with their absolute location on the system.

The dot and dot-dot directories can be opened just like any other hierarchical directory. File information intrinsics can be used to return information about these directories. However, the dot and dot-dot directories cannot be modified by a user. In addition, these two directories cannot be explicitly purged except by purging the directory they live in.

Note

The dot and dot-dot directories are not found explicitly under the root directory or under MPE/iX accounts and MPE/iX groups; however, dot and dot dot behavior is supported.

Opening a hierarchical directory

Both `HPFOPEN` and `FOPEN` can be used to open an existing hierarchical directory.

Record selection and data transfer

The `HPFOPEN` intrinsic provides the "read directory" access option when creating or opening a directory, but there are no system intrinsics that allow you to either read from or write to a directory. Note that POSIX interfaces exist to read directory entries from directories. The system itself manages a directory file, managing read access during pathname resolution and write access when new files are either created or purged in the hierarchical directory.

The `FCONTROL` intrinsic item number 5 "rewind" option allows you to rewind a hierarchical directory entry pointer to the first entry of the directory.

Getting hierarchical directory information

The following file information intrinsics provide support for hierarchical directory files:

`FFILEINFO`	Item numbers 3, 52, 53, 54, and 55 have been enhanced to return information about hierarchical directories. Item numbers 80 through 91 are new.
`FGETINFO`	`FFILEINFO` is the recommended intrinsic for returning information on a hierarchical directory (or any file, for that matter). The *foptions* parameter of `FGETINFO` cannot return the file type or record format of a hierarchical directory.
`FLABELINFO`	Item numbers 6, 7, 8, 13, 20, and 21 have been enhanced to return information about hierarchical directories. Item numbers 37 through 49 are new.
`PRINTFILE-INFO`	Enhanced to display information about an open hierarchical directory. For more information, refer to "Listing File Information" later in this chapter.

Closing or purging a hierarchical directory

The `FCLOSE` intrinsic has been enhanced to allow you to close a hierarchical directory, but the directory must be closed in the permanent file domain. For more information, refer to the section "Closing a File" later in this chapter.

You cannot purge a hierarchical directory that has entries in it other than the dot and dot-dot directory entries. You must first purge all objects under a hierarchical directory before you can purge that directory.

The only exception to this rule is when you use the `PURGEGROUP` command from the MPE/iX CI to purge an MPE/iX group directory, or the `PURGEACCT` command to purge an MPE/iX account directory (and all groups under it). When this occurs, all hierarchical directories located under the purged groups are automatically purged, whether or not they contain entries.

New File Types

Release 4.5 and 5.0 of MPE/iX provide some new file types. Here is a brief description of them. Since these interfaces are based on a commonly used standard, many textbooks are available that contain a detailed discussion of their use.

Symbolic Links

Symbolic links are permanent MPE/iX files that can be created, removed, renamed, and archived.

Creating symbolic links

Symbolic links can be created by one of the following two ways:

1. POSIX C-library function `symlink`
2. MPE/iX command `NEWLINK`. For example:

   ```
   :NEWLINK ./syml, /SYS/PUB/SYSSTART
   ```

   This creates the file `syml` in the current working directory as a symbolic link to the pathname `/SYS/PUB/SYSSTART`.

Removing symbolic links

Symbolic links can be removed in the following ways:

1. POSIX C-library function `unlink`
2. MPE/iX command `PURGELINK`

   ```
   :PURGELINK ./syml
   ```

   This purges the file `syml`.

Renaming symbolic links

Symbolic links can be renamed by calling the POSIX C-library function `rename`.

The MPE/iX command `RENAME` does not rename the symbolic link itself, it renames the file pointed to by the symbolic link.

Archiving symbolic links

Symbolic links can be stored and restored to your MPE/iX system by using the MPE/iX `STORE` and `RESTORE` commands like any other file on the system.

MPE/iX `LISTFILE` command

This command displays information about the symbolic link file and its contents. For example: If the file

```
/SYS/PUB/syml
```

is a symbolic link to `/SYS/PUB/NL`, then

```
LISTFILE /SYS/PUB/syml,5
```

displays information about the file `syml` and in addition, it displays:

```
SYMLINK TARGET: /SYS/PUB/NL
```

Pipes

A *pipe* provides a method to take output from one process and use it as input to another process. The `HPPIPE` intrinsic creates a new file type object called *pipe* for this purpose. It provides a one-way flow of data. After creating a pipe file type object, the `HPPIPE` intrinsic returns both read-access and write-access file numbers. Related processes can access the pipe.

Data can be written to the write-end and read from the read-end of the pipe. The data is accessed in a first-in-first-out manner, or FIFO.

A pipe file does not have a name associated with it and is not inserted into any directories upon creation. After the final close of a pipe file, the file is deleted. Pipe files are treated as new files.

FIFOs

A FIFO is similar to a pipe. It is a one-way flow of data, with the first byte written to it being the first byte read from it. Unlike pipes, a FIFO has a name associated with it, allowing unrelated processes to access a single FIFO. FIFOs are also called "named pipes."

The program `MKNOD.PUB.SYS` has been modified to support the creation of a new file type object called *fifo* to be used for this purpose. It creates a FIFO file and inserts it into the directory.

The `HPFOPEN` intrinsic has been modified to support the opening of the FIFO file. After the final close of a FIFO file, the data remaining in the FIFO is deleted, but the FIFO container remains and can be reopened and used again. `FOPEN` can open FIFOs as well.

MPE/iX Byte-Stream Files

Beginning with MPE/iX Release 4.5, the byte-stream record format is available to allow an MPE/iX file to reproduce POSIX byte-stream behavior.

Note

In this manual, the term "byte-stream file" is used to refer to an MPE/iX standard ASCII disk file with a record format of byte- stream.

The standard file used in a POSIX environment is the byte-stream file. Conceptually, a POSIX byte-stream file has no system-defined record structure. A user can write any number of bytes from a user buffer to a byte-stream file starting at the current file offset. By default, byte-stream files created by `HPOPEN` have a file limit of 2 Gigabytes. Byte-stream files created by `FOPEN` have a default file limit of 1023 bytes. The size can be defined by the user.

Likewise a user can read any number of bytes from a byte-stream file to the user's buffer starting at the current file offset. In short, a byte-stream file is a file that stores data in the form of a stream of bytes. It is the responsibility of the programmer to provide meaning and structure to the data.

Creating a byte-stream file

You use either the `HPFOPEN` or `FOPEN` intrinsics to create a byte-stream file. (Byte-stream files can also be created using POSIX.1/iX library functions.) You can also use the `BUILD` command, for example:

```
:BUILD X;REC=,,B;DISC=100000
```

where `B;` indicates byte-stream record format.

The following table defines the file and access attributes imposed on an MPE byte-stream file when it is created. These attributes cannot be changed during the life of the byte-stream file.

Table 4-3.
Access Attributes of Byte-Stream Files

Attribute Name	Default
Domain	Must explicitly specify "create as a permanent file" if the file is being created under a hierarchical directory.
Record format	Byte-stream. Must be explicitly specified in order to create a byte-stream file.
Carriage control	No carriage control (NOCCTL). All other specifications are ignored.
File type	Standard disk file (STD). All other specifications are ignored.
Multirecord	Multirecord access (MULTI). All other specifications are ignored.
Copy mode	The file is always accessed as its own file type. All other specifications are ignored.
Record size	One byte. All other specifications are ignored.
Remote environment	Remote access to byte-stream files is not supported. Specifying this option results in an error.
File size	Two gigabytes. Care should be taken when resetting the file limit of a byte-stream file. POSIX does not support MPE/iX file limits. To minimize the impact of this unsupported behavior, the file limit of a byte-stream file is set to the maximum possible value.
Block factor	One byte per block. All other specifications are ignored.

Table 4-3.
Access Attributes of Byte-Stream Files
(continued)

Attribute Name	Default
Fill character	ASCII null character. Unlike other record formats, the file system does not pad byte-stream files except when the `FPOINT` intrinsic is called to set the record pointer beyond existing data in the file. If data is later written at this point, the resulting gap is padded with ASCII null characters.
Inhibit buffering	Inhibit buffering (NOBUF). All other specifications are ignored.
ASCII/binary	ASCII. All other specifications are ignored.

Note

If you are using the `FOPEN` intrinsic to create a byte- stream file, the number of bits available in the record format field (bits 8:2) of *foptions* parameter is inadequate to specify byte-stream record format.

To create a file with a byte-stream record format, you must set *foptions* bits 8:2 to "01" (variable) and *foptions* bit 1:1 to "1" (record format extension).

User-defined labels can be attached to a byte-stream file. For these files, the behaviors of both the `FREADLABEL` and `FWRITELABEL` intrinsics remain unchanged.

Opening a byte-stream file

MPE/iX provides two mechanisms to open and access a byte-stream file. The behavior of the byte-stream file differs depending on how you open it. The two mechanisms are:

- The open byte-stream file behaves as a byte-stream file only if you explicitly specify "byte-stream" in the

`HPFOPEN` record format option (item 77). The behavior of byte-stream files is described in the following sections.

- The open byte-stream file behaves as a variable-length record format file if you do not explicitly specify "byte-stream" in the record format option. (The default MPE system behavior.)

Opening a byte-stream file as a buffered variable-length record format file allows existing applications to access a byte-stream text file in a known and expected manner without having to perform any code revisions.

For example, many text editors supported today can access either fixed-length record format files or variable-length record format files. By allowing a byte-stream file to be opened as a variable-length record format file, these text editors can read from and write to byte-stream files without concern for compatibility problems that occur when trying to access a file with an unknown record structure.

Record selection and data transfer

Both the blocking factor and the record size of a byte-stream file are equal to one. The record pointer is defined by its offset from the beginning of a file of one-byte records. The offset of the first record, or byte, is zero. When a byte-stream file is first created, the record pointer points to byte zero.

Conceptually, there should be no limit to the number of bytes a user can transfer to and from a byte-stream file. In fact, processes performing data transfer using POSIX.1/iX library functions can transfer up to two gigabytes of data in one operation. However, the *length* parameter used by MPE/iX data transfer intrinsics (`FREAD`, `FREADDIR`, `FWRITE`, `FWRITEDIR`) to specify the number of bytes to transfer limits the value to 32,768 (this value must be passed as a negative-signed integer value to indicate byte transfer).

Sequential access. Byte-stream files are always opened for multirecord access. Because data transfer to and from byte-stream files usually occurs in blocks of bytes, data transfer to and from byte- stream files usually occurs using sequential access. Sequential access is the typical form of byte-stream data transfer.

The behavior of both the `FWRITE` and `FREAD` intrinsics are straightforward with respect to byte-stream files. Data transfer to and from byte-stream files is usually accomplished by transferring a number of bytes to or from the file.

After a successful data transfer, the record/byte pointer is set to the next unread/unwritten byte. Carriage controls are ignored when accessing byte-stream files.

Care should be taken when accessing a file using sequential access methods. Because there is no system-imposed record structure in a byte-stream file, applications that require some form of record structure in order to sequentially access user-defined records in the file must define and manage their own record blocking and deblocking.

Random access. The behavior of both the `FWRITEDIR` and `FREADDIR` intrinsics are straightforward with respect to byte-stream files. When performing random access on a byte-stream file, remember that the logical record number passed in the *lrecnum* parameter corresponds to the desired byte offset from the beginning of the file (byte offset 0).

Successful data transfers set the record/byte pointer to the next unread/unwritten byte. Carriage controls are ignored when randomly accessing byte-stream files.

Care should be taken when accessing a file using random access methods. Because there is no system-imposed record structure in a byte-stream file, applications that require some form of record structure in order to randomly access user-defined records in the file must define and manage their own record blocking and deblocking.

Append access and append mode. A byte-stream file opened for append access using the `HPFOPEN`/`FOPEN` "access type" option performs in a straightforward manner. When the file is opened, the record/byte pointer is set to the end of the file. Subsequent writes to the file using `FWRITE` appends data to the end of the file, ensuring that existing data cannot be overwritten.

Beginning with MPE/iX Release 4.5, item number 51 of the `FCONTROL` intrinsic can be used to enable a new type of append access called "append mode". Append mode is available only for byte-stream files.

Append mode differs from append access in the following ways:

- Append mode can be dynamically enabled and disabled while a byte-stream file is open. Append access is established when opening a file and cannot be changed dynamically for a file number.
- Append mode affects all processes sharing a logical record pointer. Append access is a process-local attribute affecting only the access of the process opening the byte-stream file. (The `HPFOPEN`/`FOPEN` MULTI and GMULTI options permit logical record pointers to be shared.)
- A byte-stream file's logical record pointer can be manipulated using `FPOINT` or `FREADDIR` when append mode is enabled. Append access prohibits logical record pointer operations. For example, calls to `FPOINT` and `FSPACE` result in an error for files opened for append access.

Update access. Update access is not allowed on byte-stream files. If you attempt to set the *access type* option of `HPFOPEN/FOPEN` to "update" when opening a byte-stream file, an error occurs. Likewise, a call to the `FUPDATE` intrinsic on a byte-stream file results in an error.

Moving the file pointer. Use the `FPOINT` intrinsic to move the record pointer from its current offset to any offset in a byte-stream file. `FPOINT` also allows you to set the record pointer beyond existing data in the file. If data is later written at this point, the resulting gap is padded with the fill character, which is defaulted to the ASCII null character, but may be set to any character at creation time. This case is the only circumstance where the file system pads a byte-stream file.

Use the `FCONTROL` intrinsic with a control code of 5 to "rewind" the record pointer of a byte-stream file to the first record/byte, a file offset of zero.

Use of the `FSPACE` intrinsic on a byte-stream file is not allowed. An attempt to do so results in an error. Instead, use the `FPOINT` intrinsic.

Note The affect of moving the record pointer of a shared byte-stream file may be different than expected if append mode is enabled for that file. For more information, refer to the discussion of the new append mode feature in the section called "Append access and append mode" earlier in this chapter.

Renaming a File

Prior to MPE/iX Release 4.5, files could not be renamed across account boundaries. A reason for this restriction was to ensure file security. This restriction also made it so that disk space would be accounted to the account in which the user was located. File ownership specified in the file's label was limited to only the creator name in the form *username*, a character string representing the name of the user who created the file (for example, `LINDA`). A creator name was only unique within a single MPE account, not across the whole system. A different account may have the same user name to specify a different user.

For example, `LINDA.FINANCE` has complete access to all files in the `FINANCE` account where the file's creator specifies `LINDA`. In addition, `LINDA.MARKETNG` has complete access to all files in the `MARKETNG` account where the file's creator specifies `LINDA`. If a file created by `LINDA.FINANCE` was allowed to be renamed to the `MARKETNG` account, the file system would allow `LINDA.MARKETNG` to have total access to that file, believing that `LINDA` in account `MARKETNG` was the creator. This is considered a security breach.

Beginning with MPE/iX Release 4.5, all newly created files, renamed files, and copied files have file ownership specified in the file label in the form *username.accountname*. This enhancement of file ownership from creator (in the form *username*) to owner (in the form *username.accountname*) ensures the uniqueness of file ownership across the whole system. This enhancement of file ownership corresponds to the new feature of a user ID (UID) associated with each user on the system.

Using the example specified above, the file label of a file created by `LINDA.FINANCE` that is renamed to the `MARKETNG` account specifies the creator/owner to be `LINDA.FINANCE`. The file system is able to distinguish owner `LINDA.FINANCE` from `LINDA.MARKETNG` and does not allow `LINDA.MARKETNG` creator/owner access to that file.

File labels of files existing on your system prior to MPE/iX Release 4.5 that have not been copied or renamed continue to specify ownership using only the creator name (in the form *username*). Since these files remain within the MPE account structure (that is, directly under MPE groups), either standard MPE file system security features or ACDs continue to ensure security for these files.

When a file is renamed across account boundaries, the file owner can continue to access the file as the owner only if MPE/iX security provisions allow him/her access to that file. (However, the renamed file still belongs to the original file group and is still managed by the original account manager.)

For example, if a process being executed by `LINDA.FINANCE` were to call the `FRENAME` intrinsic to rename a file `PAYROLL` to `/MARKETNG/PUB/directory1/PAYROLL`, the process (whose UID is currently associated with `LINDA.FINANCE`) must have either SM capability assigned to the user associated with the process's UID or the following access rights:

- MPE save files (SF) capability assigned to the user associated with the process's UID (in this case, `LINDA.FINANCE`).
- Delete directory entry (DD) access to the source file's parent directory (specified in the ACD associated the directory).
- Traverse directory (TD) access to all parent directories of the target file (specified in the ACD associated with each directory).

- Create directory entry (CD) access to the target file's parent directory (specified in the ACD associated with `directory1/`).
- Standard file system security provisions or the ACD associated with the source file allows the user write access to the source file if it lives in a group. Write access to the file is only required for files in MPE groups. It is part of the definition of DD access for groups.

For additional restrictions on renaming a file using the `FRENAME` intrinsic or the `RENAME` command, refer to the appropriate descriptions located in the *MPE/iX Intrinsics Reference Manual* (32650-90028) and *MPE/iX Commands Reference Manual, Vol. II* (32650-90374), respectively.

Enhancements to MPE/iX File System Security Features

File system access control has been enhanced to accommodate new hierarchical file system features so that MPE/iX can control access to files created outside MPE groups and to hierarchical directories.

Application developers need to understand the concepts described in the following sections in order to effectively use new MPE/iX security features.

Object ownership

In past releases, MPE/iX has used the creator name, a user name in the form *username*, to track file ownership. The creator name for the root directory, MPE groups, and accounts was not recorded. Only files were assigned creator names. For example, if a user `JOE` in his logon account `FINANCE` created a file named `MYFILE`, the creator name associated with that file was `JOE`. Of course, if there was another user `JOE` in another account `PAYROLL`, any files he created also had the creator name

`JOE` associated with them. This did not cause security problems because neither `JOE` could create files outside their own logon account.

Beginning with MPE/iX Release 4.5, files and hierarchical directories can be created outside the logon account. For example, if given the proper access rights, `JOE.FINANCE` can create a file in the same directory that `JOE.PAYROLL` can. Using only the creator name to determine ownership, MPE/iX cannot determine which `JOE` is the creator of this file. For this reason, unqualified user names are no longer sufficient for indicating object ownership across the whole system.

Beginning with MPE/iX release 4.5, file ownership for all newly created, copied, or renamed files is indicated by a fully qualified user name in the form *username.accountname*. This fully qualified user name is referred to as the **file owner** and is associated with a user ID (UID).

The file creator was a static value for the lifetime of a file. However, the file owner can be changed during the lifetime of a file.

File owners are assigned to all newly created files and directories. The file owner of the root directory is `MANAGER.SYS`. MPE account and MPE group directories created before installation of the new FOS release lack file owners since older releases of MPE/iX did not initialize ownership information.

Directories with uninitialized file ownership information appear to have a file owner of "0" when displayed by `LISTFILE`. The system reserves the zero UID value for use by MPE/iX. Zero UID values cannot be assigned to users, files, or directories.

Object ownership for MPE groups, accounts, and the root directory are new concepts. The existing access control policy for these directory types is based solely upon appropriate privilege. Account managers did not retain any additional access to MPE groups they had created if their AM capability was removed by their system manager.

Starting with MPE/iX Release 4.5, the ability to create or delete entries in the root directory, MPE groups, and MPE accounts is no longer based solely on appropriate privilege. Directory file owners are granted all access to the directories they own.

Sharing objects

Prior to MPE/iX Release 4.5, MPE accounts provided the basis for file sharing. All file user types other than the ANY file user type were members of the logon account. Beginning with MPE/iX Release 4.5, files created under the root directory or below some combination of hierarchical directories below the root directory are not within an MPE account. File sharing on MPE/iX has been enhanced using the concept of the file group ID (GID).

When files and directories are created, they are assigned their parent directory's file group ID (GID). MPE accounts are assigned a unique GID when they are created. The HPGID database records this association of MPE account and file GID.

Uninitialized file group information appears as a file GID of "0" when displayed by `LISTFILE`. The GID database interfaces reserve zero GID values for use by MPE/iX. Users, files, and directories cannot be assigned zero GID values.

Closing a File

The following sections describe modifications to the behavior of the `FCLOSE` intrinsic beginning with MPE/iX Release 4.5.

Closing shared files

Prior to Release 4.5, if a conflict occurred between the disposition specifications of multiple `FCLOSE` intrinsic calls on a shared file, the disposition specification that had the lower positive-integer value always took precedence when the file was finally closed.

Beginning with Release 4.5, this behavior is modified if one of the processes sharing the file invokes the POSIX.1/iX library function `unlink()` to purge the file from the directory. Upon successful completion of the `unlink()` call, the file is moved from the permanent file domain to the new file domain. When this occurs, the behavior of the `FCLOSE` intrinsic on the shared file may change.

Care must be taken only in an environment where processes using the `FCLOSE` intrinsic share files with processes calling the `unlink()` C library function available through the MPE/iX Developer's Kit.

Closing directory files

Directory files exist only in the permanent file domain. Attempting to perform an `FCLOSE` specifying a domain disposition of anything other than "no change" or "close as permanent" on an open directory file results in an `FCLOSE` error. The directory file remains open.

Listing File Information

The `PRINTFILEINFO` intrinsic has been enhanced to display file attributes and access options introduced with MPE/iX Release 4.5, including new record formats, file types, and directory types.

The file creator field of the `PRINTFILEINFO` display has been enhanced to reflect the new concept of file ownership (in the form *username.accountname*). The file creator display field (identified by `ID IS` ...) has been moved to the line following its past position and given the new identifier `FILE OWNER:` ... . Even files whose labels express only the creator name (in the form *username*) display the fully qualified owner name.

Beginning with MPE/iX Release 4.5, file names expressed using HFS absolute pathnames (pathnames beginning from the root directory) may contain 1024 characters. The `PRINTFILEINFO` intrinsic prints a wider display (79 characters instead of 50 characters) if called to display information on a file whose pathname exceeds 32 characters in length. Pathnames greater than 62 characters in length are wrapped to following lines.

Following is an example of a `PRINTFILEINFO` display of a file whose pathname exceeds 62 characters in length.

```
+-F-I-L-E---I-N-F-O-R-M-A-T-I-O-N---D-I-S-P-L-A-Y-----------+

!  FILE NAME IS /users/linda/official/finances/payroll/san-
diego/user-identifi !
!               cation
!
!  FOPTIONS: NEW,BINARY,FORMAL,F,NOCCTL,FEQ
!
!            NOLABEL
!
!  AOPTIONS: INPUT,NOMR,NOLOCK,DEF,BUF,NOMULTI
!
!            WAIT,NOCOPY
!
!  DEVICE TYPE: 3      DEVICE SUBTYPE: 8
!
!  LDEV: 62       DRT: 8         UNIT: 0
!
!  RECORD SIZE: 128    BLOCK SIZE: 128   (WORDS)
!
!  EXTENT SIZE: 128    MAX EXTENTS: 8
!
!  RECPTR: 0           RECLIMIT: 1023
!
!  LOGCOUNT: 0            PHYSCOUNT: 0
!
!  EOF AT: 0           LABEL ADDR: %00000000000
!
!  FILE CODE: 0        ULABELS: 0
!
!  FILE OWNER: LINDA.FINANCE
!
!  PHYSICAL STATUS: 0000000000000000
!
!  ERROR NUMBER: 0     RESIDUE: 0
!
!  BLOCK NUMBER: 0            NUMREC: 1
!
+-----------------------------------------------------------+
```

Figure 4-1. Enhanced PRINTFILEINFO Display

For additional information, refer to the `PRINTFILEINFO` intrinsic description located in the *MPE/iX Intrinsics Reference Manual* (32650-90028).

Terminating File Name Strings

Many MPE/iX intrinsics that access a file by its file name require that you terminate the file name string specified in the *formaldesignator* parameter with a nonalphanumeric character that is not reserved by MPE/iX. Prior to MPE/iX Release 4.5, you could use the nonalphanumeric characters, such as "-" and "_" to terminate a file name string.

Beginning with MPE/iX Release 4.5, the hyphen "-" and underscore "_" characters can be interpreted as part of the file name if the system is interpreting the name using HFS syntax. (The system could then attempt to interpret invalid data on the stack as part of the file name.) Therefore, these two characters should no longer be used by the following intrinsics to terminate a file name string:

- `CATOPEN`
- `FLABELINFO`
- `FOPEN`
- `FPARSE`
- `FRENAME`
- `STACKDUMP`

The above mentioned characters can still be used to terminate MPE syntax names, but this is not recommended practice, because these characters will not terminate an HFS-Escaped name. The recommended practice is to terminate a file name string with an ASCII null character.

For additional information about these intrinsics, refer to the appropriate intrinsic descriptions located in the *MPE/iX Intrinsics Reference Manual* (32650-90028).

Accessing a File by Its File Name

MPE/iX intrinsics that access a file by its file name require that you pass the file name string in the *formaldesignator* parameter. Some of these intrinsics have additional parameters that either pass or return file name strings. Most, but not all, parameters have been enhanced to accommodate HFS syntax.

Enhanced intrinsic parameters

The following MPE/iX intrinsics have been enhanced to accommodate a file name string that uses either MPE syntax or HFS syntax.

Table 4-4. Intrinsic Parameter Enhancements

Intrinsic	Enhancement
`FFILEINFO`	Item number 80 (new)
`FLABELINFO`	*formaldesignator* parameter and item number 38 (new)
`FOPEN`	*formaldesignator* parameter
`FRENAME`	*formaldesignator* parameter
`HPACDINFO`	Item number 1
`HPACDPUT`	Item numbers 1, 2, and 15
`HPFOPEN`	Item numbers 2, 41, 51, and 52 (*filereference* parameter only)
`STACKDUMP`	*formaldesignator* parameter. However, pathname must resolve to an MPE group.

In the above intrinsics, file names specified using HFS syntax must conform to HFS syntax rules, described in Chapter 2.

For additional information about these intrinsics, refer to the appropriate intrinsic descriptions located in the *MPE/iX Intrinsics Reference Manual* (32650-90028).

Special enhancements to HPFOPEN

The `HPFOPEN` intrinsic provides an optional exception to the rule for specifying file names using HFS syntax. `HPFOPEN` item number 41 enables you to pass a value that indicates how `HPFOPEN` should interpret a file name passed in item numbers 2 or 51. The following table describes item number 41:

Table 4-5. HPFOPEN Item Number 41

Value	Meaning
0 MPE-Escaped	Default. If the first character is either a dot "." or a slash "/", use HFS syntax to interpret the pathname. Otherwise, use MPE syntax to interpret the file name.
1 MPE only	Use only MPE syntax to interpret the file name. The name must be in the MPE name space.
2 HFS	Use HFS syntax to interpret the pathname.

A dot (".") is not required to indicate a relative pathname (a pathname relative to the current working directory), although it is allowed. File name strings are interpreted literally as HFS pathnames (that is, no upshifting occurs, and HFS nonalphanumeric characters are allowed).

For example, if the process's current working directory pathname is `/dir1/dir2/dir/` and the specified pathname is `dir4/dir5/Myfile`, `HPFOPEN` prepends the current working directory pathname to the specified pathname prior to pathname resolution and searches for the file using the pathname `/dir1/dir2/dir/dir4/dir5/Myfile`.

For additional information, refer to the `HPFOPEN` intrinsic description located in the *MPE/iX Instrinsics Reference Manual* (32650-90028).

Restricted intrinsic parameters

The following MPE/iX intrinsics have not been enhanced to accommodate a file name string that uses HFS syntax. File names must be specified using only MPE syntax.

Table 4-6. Restricted Intrinsics

Intrinsic	Restriction
`CATOPEN`	File name string passed in *formaldesignator* parameter must use MPE syntax
`FFILEINFO`	Item number 1
`FGETINFO`	*formaldesignator* parameter
`FLABELINFO`	Item number 1
`FPARSE`	*formaldesignator* parameter

For additional information about these intrinsics, refer to the appropriate intrinsic descriptions located in the *MPE/iX Intrinsics Reference Manual* (32650-90028).

5

Managing Accounts, Groups, and Users

This chapter is primarily for system administrators who manage accounts, groups, and users. However, the information it contains is also useful to programmers who may need to understand user and group IDs.

Background

HP 3000 computers have traditionally been designed for use in an office setting where users work in separate departments. All of the users in a particular department are typically grouped together and placed in one account. Users log on to an account and group to do their work. One or more users can work in an account.

Users store their files in groups in accounts and are identified by a *user.account* string such as `MANAGER.PAYROLL`. The *user.account* associated with a job or session does not change for the life of the job or session. The user and his or her files cannot travel outside of the account boundary. Files could be copied but not renamed across account boundaries.

As of Release 4.5, MPE/iX identifies each user by a unique user ID (UID) as well as a *user.account* string. Users are arranged into groups and are identified by a group ID (GID). Users that share one account will share the same GID. Owners of files on MPE/iX are identified by the file creator field. Refer to the sections called "User and Group IDs" and "File Ownership" in Chapter 3 for more information.

The data associated with the user is stored in separate user and group databases. This means that on MPE/iX, users with proper security permissions can move outside of account boundaries. Users can be allowed to have more flexibility to move around to different directories within the hierarchical file system.

Setting Up Group and User Databases

The UID and GID databases are a required part of any operating system that implements POSIX standards. These databases contain user identities to authorize user and system interaction.

- User ID database (UID) in `HPUID.PUB.SYS` and `HPUIDNX.PUB.SYS`
- Group ID database (GID) in `HPGID.PUB.SYS` and `HPGIDNX.PUB.SYS`

The databases are automatically created when you install or update to MPE/iX Release 4.5 or later. The databases are privileged files whose contents are not visible. As system manager, you need to be aware of their existence.

MPE/iX requires that all processes have an associated UID and GID. Each user is assigned a unique user ID (UID) which the system maintains. When a user logs on, MPE/iX uses the authenticated *user.account* logon string to query the databases for the UID and GID. The IDs are stored in a run- time process table.

If no entry is found in the UID database, MPE/iX checks whether the user has SM capability. If the user does not have a UID and does not have SM capability, the user cannot log on to the system.

Group ID

Note

The term **group** in this context is distinct from an **MPE group**. To clarify, this document refers to groups under accounts as MPE groups.

The group database defines members of a file sharing group. The database also maps numerical group IDs to POSIX.1 group names in the file called `HPGID.PUB.SYS`. The GID is a unique number that identifies the group. MPE/iX automatically assigns a group ID when you create a new account unless you specify one. The group database is set up automatically when you install or update MPE/iX.

User ID

The user database maps numerical user IDs to login names in the file called `HPUID.PUB.SYS`. The UID is a unique number that identifies the user. MPE/iX automatically assigns a user ID when you create a new account or add a user unless you specify one.

Maintaining the group and user databases

It is not necessary to perform any special procedures to maintain the user and group databases. When you add, modify, or purge users, groups, and accounts, the user and group databases are modified accordingly. A utility called `PXUTIL` exists to provide version information and backdating capabilities.

PXUTIL

PXUTIL has been created to provide the commands VERSION, BACKDATE, and PURGE along with current commands of HELP, QUIT, and EXIT. The UPDATE command has been obsoleted due to the redesign of the UID/GID databases. The following describes these new commands:

1. VERSION

   The VERSION command displays the versions of the database files, the number of valid records in the databases, and the next UID and GID numbers to be used.

```
:PXUTIL.PUB.SYS
PXUTIL> version

UID DATABASE:
   HPUID.PUB.SYS Version:    A.01.00
   HPUIDNX.PUB.SYS Version:  A.01.00
   Valid Records:            368
   Uid Counter:              531

GID DATABASE:
   HPGID.PUB.SYS Version:    A.01.00
   HPGIDNX.PUB.SYS Version:  A.01.00
   Valid Records:            122
   Gid Counter:              276
```

2. BACKDATE

   The BACKDATE command purges the UID/GID databases for backdating purposes. The user information must be saved on tape first via :STORE;;DIRECTORY. A warning message to this effect is displayed with an option to cancel the process. The following example demonstrates this.

```
:PXUTIL.PUB.SYS
PXUTIL> backdate

You MUST do a :STORE ;;DIRECTORY
BEFORE running this pre-backdating operation
to preserve user information.  Continue (Y/N)?
```

3. PURGE

   The PURGE command purges old UID/GID database files created from a UID/GID system boot error (HPUIDOLD.PUB.SYS, HPUIDONX.PUB.SYS, HPGIDOLD.PUB.SYS, and HPGIDONX.PUB.SYS).

4. HELP

   The HELP command displays instruction on the utility.

5. QUIT

   The QUIT command exits the utility.

6. EXIT

   The EXIT command exits the utility.

Setting Up Accounts and Adding Users

Once the user and group databases are set up, you add accounts and users just as you would on traditional MPE systems. MPE/iX automatically updates the user and group databases as needed. You do have an additional option to specify group or user IDs when setting up accounts. You can also specify user IDs when adding users.

The tasks of creating accounts and users are described in *Performing System Management Tasks* (32650-90004).

Setting up accounts

To set up an account, use the NEWACCT command and its options. Refer to the *MPE/iX Commands Reference Manual, Vol. I* (32650-90003) for the complete syntax of NEWACCT.

You must have system manager capabilities to set up a new account. You must specify the account name and identify an account manager. You also have the option of specifying a user ID number and group ID number, or a unique number is automatically assigned. If you specify user and group ID numbers, you should assign numbers over 100. MPE/iX reserves numbers less than 100 for system-defined IDs.

For example, to create a new account called PAYROLL for the payroll department:

```
:NEWACCT PAYROLL,MANAGER;PASS=PAYUS2;UID=150;GID=120;
```

The new account, PAYROLL, contains one user (MANAGER), who is the account manager. The password for the account is PAYUS2. The user ID of 150 identifies the account manager MANAGER.PAYROLL. The group ID of 120 identifies the payroll account. Also, the account automatically contains one group: PUB.

If you omit the UID and GID parameters, MPE assigns a unique UID to the account manager, MANAGER.PAYROLL, and a unique GID to the account, PAYROLL.

If you assign a UID or GID that is already in use, an error message is displayed.

Adding users

To add a user to an account, use the `NEWUSER` command and its options. Refer to the *MPE/iX Commands Reference Manual, Vol. I* (32650-90003) for the complete syntax of `NEWUSER`.

Only the system manager or the account manager for the account can add new users. You must include a user name. If you do not include an account name, the user is added to the account that you are currently in.

For example, you can create a new user in the `PAYROLL` account as follows:

```
:NEWUSER BANKS.PAYROLL;PASS=ALEX;HOME=SALES;UID=120
```

This example creates a user called `BANKS.PAYROLL` in the `PAYROLL` account with a home group `SALES`. The user ID is 120.

6

Managing Directories

This chapter describes what a directory is and defines related terminology. Then it includes step-by-step procedures on how to manage directories on MPE/iX, including:

- Creating directories
- Listing directories
- Changing the current working directory
- Determining space used by directories
- Repairing directories
- Deleting directories

What Is a Directory?

A **directory** is a special kind of file that contains entries that point to other files and directories. Directories, like MPE groups, help applications and users to organize files in a logical manner on the system. A directory contained within another directory is also called a **subdirectory**. A directory that contains other directories is called a **parent directory**. Directories, subdirectories, and files form a structure for the file system. A **directory entry** associates a file name with a file.

MPE/iX has four types of directories:

- root directory
- accounts
- MPE groups
- hierarchical directories

The structure of the HP 3000 file system traditionally allowed for a three-level hierarchy made up of accounts, groups, and files. Files could exist only below MPE groups. The directory structure on MPE/iX is hierarchical (like a tree structure). You can create directories in groups and accounts that can, in turn, hold files and subdirectories.

The hierarchical directory structure has a common root, called the **root directory** and is denoted by a leading slash (/). The root is the foundation of the HP 3000 directory structure. In traditional MPE terms, the root is the parent of all accounts. Files, directories, and accounts may be located in the root directory.

Note You can put files under accounts in addition to being able to put them under groups, directories, and root.

Figure 6-1 shows an example hierarchical directory structure.

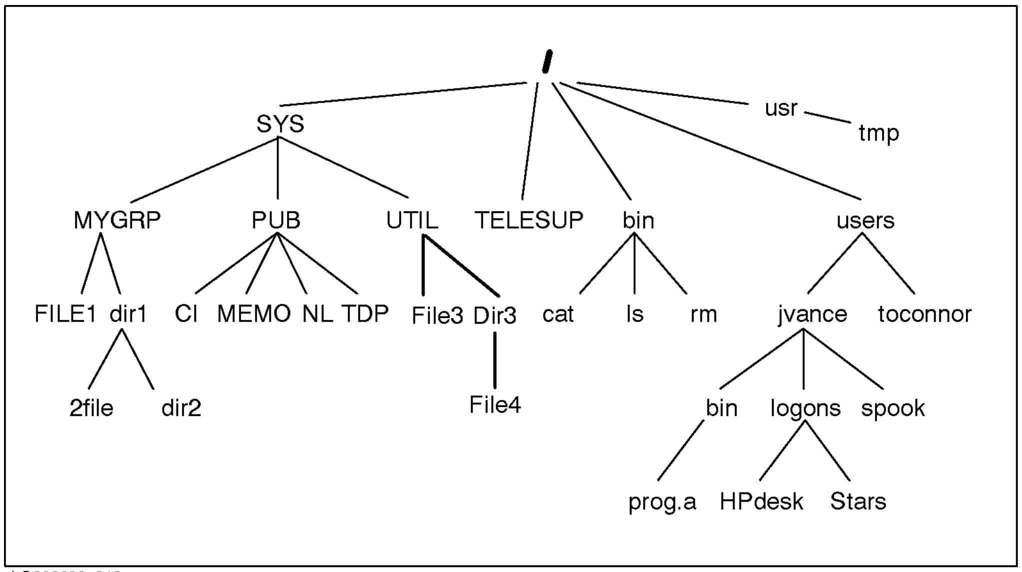


Figure 6-1. Example of Hierarchical Directory Structure

Pathnames

A **pathname** describes the route (or path) to a file. It is composed of the file name prefixed by any directory names. A slash (/) separates each part of a pathname. The maximum pathname length is 1023 characters. Although pathnames can be 1023 characters, other system limits restrict pathname length. For example, the command interpreter limits the number of characters you can type in a command to 511 characters.

While the system does not support pathnames of greater than 1023 characters, it does not explicitly prevent their creation. Although this limit is large enough to cover any practical usage, the system will respond in a degraded fashion to any object with a path longer than

1023 characters. For example, the `STORE` utility will not be able to back up these files. The offending objects will have to be removed "by hand" by setting your CWD far enough down the path so that a 1023 character relative name can reference and purge the files and/or directories.

When the pathname begins at the root directory, it is called an *absolute pathname.* An example of an absolute pathname is:

```
/SYS/MYGRP/dir1/2file
```

The example describes the location of the file `2file.` It is in the directory `dir1,` which is a subdirectory of the `MYGRP` group; `MYGRP` is in the `SYS` account, which is a subdirectory of the root directory (/).

An absolute pathname is similar to a fully qualified MPE file name in that the names of all the components are listed. The fully qualified MPE file name `TDP.PUB.SYS` translates to the absolute pathname `/SYS/PUB/TDP`.

Note All MPE syntax names must be specified in uppercase.

Other pathnames are relative to a process's current working directory. (The current working directory is the directory where you are working.) So if you specify the name `./jvance/spook`, MPE looks for a subdirectory called `jvance` in your current working directory, then for a file called `spook` in that directory.

Access to Directories

You can use access control definitions (ACDs) to grant the following accesses to directories:

CD - Create directory entries
DD - Delete directory entries
RD - Read directory entries
TD - Traverse directory entries
RACD - Read ACD
NONE - No access

Note By default, directories allow Read ACD privilege to all users on the system (RACD:@.@). You assign or change directory ACDs using the `ALTSEC` command.

You can use the `LISTFILE` command to display directory ACDs. Refer to Chapter 9 for more information about ACDs and to the *MPE/iX Commands Reference Manual, Vol. I* (32650-90003) for detailed syntax and descriptions of `ALTSEC` and `LISTFILE`.

The separation of creating and deleting directory entry permissions could be used to allow a user to create files in a directory but not be able to purge them.

The root directory, accounts, and groups cannot be assigned ACDs. However, all users are granted access equivalent to read directory entries (RD) access and traverse directory entries (TD) access to root, all accounts, and all MPE groups.

Initially, only the system manager has create directory entries (CD) and delete directory entries (DD) access to root. Save access for an MPE group implies CD and DD permission. A user can create or delete a directory in an MPE group if the group grants Save access to the user.

The only users that can create files or directories in accounts are users with SM capability, or users with AM capability in their own accounts.

Once a file or directory is created under the root or an account, the security for that object can be changed to allow broader access, if desired.

Creating and Naming Directories

This section describes the following topics:

- Conventions for naming directories
- Security when creating directories
- Creating directories in your current working directory
- Creating directories in another group

Conventions for naming directories

Directory names follow the same rules as file names. Directory names can include the special characters - (hyphen), _ (underscore), and . (dot); they can be uppercase, lowercase, or a combination of the two. The maximum name length for a directory created in the root directory, an MPE group, or an account is 16 characters. Hierarchical directories can contain names that are up to 255 characters long. They cannot begin with a - (hyphen).

The file names dot (.) and dot-dot (..) have special meaning. Dot (.) is the identity function, or shorthand for the directory itself, wherever that directory is. For example:

```
/A/B/./C
```

The . entry does not refer to the CWD, but to the `B` directory, since that is the location you are at in the path.

A leading dot meaning the CWD is a result of the fact that a non-absolute name is a relative name (relative names are always relative to the CWD), therefore, the leading dot names the CWD.

The name dot-dot refers to thedirectory that is the next level up, also called the *parent directory*.

Refer to Tables 2-1, 2-2, and 2-3 in Chapter 2 for additional naming restrictions.

Security when creating directories

You can create directories in any directory, group, or account to which you have CD access.

You must have the following privileges to create directories:

- Create directory entries (CD) access for the parent directory
- Save files (SF) capability
- Traverse directory entries (TD) access to the parent directory

Access to directories is controlled by ACDs. Refer to Chapter 9 for information on using ACDs to change directory access.

Creating directories in your current working directory

To create a directory in your current working directory:

- Use the `NEWDIR` command.

For example:

```
:NEWDIR CPROGS
```

This command creates a directory called `CPROGS` in the directory in which you are currently working.

If you typed the name in lowercase and did not preface it with a `./` or `/`, MPE/iX converts it to uppercase:

```
:NEWDIR cprogs
```

MPE creates a directory called `CPROGS` in your current working directory. Note that because of the upshifting that occurs, executing this command has the same result as executing the command with `CPROGS` in uppercase.

If you want to use HFS syntax for naming a directory, you need to preface the name with ./ or /. For example, to create a directory with a lowercase name:

```
:NEWDIR ./cprogs
```

This example creates the directory cprogs in your current working directory.

Creating directories in another group

To create a directory in a group other than your logon group, use the full pathname of the directory in the NEWDIR command line.

For example, if you have TD access to /PRG/BOB and CD access to BOB on your system, you can type the following command:

```
:NEWDIR /PRG/BOB/cprogs
```

This command creates a new directory called cprogs in the group called BOB in the PRG account.

If you try to create a directory in a directory that you don't have CD access to, you receive a message telling you that you need CD permission to create new directories.

Listing Directories

The LISTFILE command traverses hierarchical directories and accepts pathnames. Because MPE/iX file names can be long, the output format displays any names with the HFS syntax as the last item, wrapping the ends of long file names onto the next line, if necessary.

To list only the directories that are within the current working directory, use the LISTDIR system-provided UDC. The LISTDIR UDC invokes the LISTFILE command.

For example:

```
:LISTDIR
```

This example lists all directories in your current working directory. Note that it does not list subdirectories that may exist below any of the directories located there.

LISTDIR can be used to show directories anywhere on the system:

```
LISTDIR /
```

Listing Files in Directories

Use the LISTFILE command to list files in directories.

To list all of the files (including any directories) in the current working directory:

```
:listfile ./@

PATH=/OFFICIAL/GP1

FILE1  dir2/ file1
```

To list only files named using MPE syntax:

```
:LISTFILE @
```

Deleting Directories

Use the PURGEDIR command to delete a hierarchical directory. MPE-Escaped naming rules apply.

You cannot use PURGEDIR to delete an account, an MPE group, the root directory, or a file. You will receive an error message if you try. Although you can specify (.) and (..) in the PURGEDIR command, you cannot delete these directories.

You must have traverse directory entries (TD) access to the directory you want to delete and any other directories referenced in the pathname. You must have delete directory entries (DD) access to the parent directory of the directory you are deleting.

Deleting empty directories

To delete a directory that is empty:

- Use the PURGEDIR command. For example:

```
:purgedir /MYACCT/MYGRP/dir1
Directory "/MYACCT/MYGRP/dir1" to be purged? (Yes/No)y
```

  This example deletes dir1.

If the directory that you are trying to delete is not empty, you receive an error message and it is not deleted. In the following example, dirwithfiles contains files.

```
:purgedir /MYACCT/MYGRP/dirwithfiles
Directory "/MYACCT/MYGRP/dirwithfiles" to be purged?
(Yes/No)y
Directory is not empty. No PURGEDIR done.
Consider using the ;TREE option. (CIERR 994)
```

Deleting directories that are not empty

To delete a directory that is not empty:

- Use the `PURGEDIR` command with the `TREE` option, or specify a pathname ending in `/`. For example:

  ```
  :purgedir /MYACCT/MYGRP/dir1 ;TREE
  ```

  or

  ```
  :purgedir /MYACCT/MYGRP/dir1/
  ```

  Both of these examples perform the same action: they delete `dir1` and any files or directories in `dir1`.

  You must have TD and DD access to `dir1` and any directories in it to delete them. Specifying a slash at the end of a pathname or using the `TREE` option makes this a recursive deletion that starts from the bottom of the tree and works its way up the tree.

  If any of the files in the directory (or directories) are open when you execute the command, you receive an `IN USE` warning message. The directory that contains the open file and the open files are not deleted. All other files and directories are deleted.

Deleting using wildcards

To delete a directory and the files or directories it contains using wildcards:

- Use the `PURGEDIR` command. For example:

  ```
  :purgedir /MYACCT/MYGRP/@
  ```

  This example deletes all directories rooted to `/MYACCT/MYGRP`.

  To delete all empty directories under the CWD (Current Working Directory) with `TMP` in their name:

  ```
  :purgedir @TMP@
  ```

  To delete all directories under the CWD with names beginning with `TMP`, and all objects below these directories:

```
:purgedir TMP@; TREE
```

To delete all directories under the CWD with names ending with TMP, and all objects below these directories:

```
:purgedir ./@TMP/
```

When wildcards are specified with *dir_name*, then RD access is required to the parent directory of each wildcard component. If the purge is multilevel, then TD, RD, and DD accesses are necessary to each directory below *dir_name*.

Changing Your Current Working Directory

Use the CHDIR command to move your CWD from one directory to another. You must have created the directory that you want to move to or have appropriate permissions to access an existing directory. For example, if you want to change to /SYS/PUB, you must have traverse directory entries (TD) access for both SYS and PUB.

You can also use CHDIR to return to your logon directory.

For example, if you created a directory called memos in your logon group (ALEXB) in your account (FOX) and your current working directory is /FOX/ALEXB, you can move to it as follows:

```
:chdir ./memos
```

The name ./memos is a relative pathname. MPE interprets it relative to the current working directory. You can move back up to the MPE group level by specifying the full pathname:

```
:chdir /FOX/ALEXB
```

Or you can use a couple of shortcuts:

```
:chdir ..
```

Here .. represents the next level up or the parent directory. You can also most simply type

```
:chdir
```

to move back to your initial working directory (logon group).

```
:chdir /SYS/PUB/CMDFLS
```

This example changes to the directory `CMDFLS` in the `PUB` group in the `SYS` account.

Showing Your Current Working Directory

MPE/iX provides a CI variable called `HPCWD` that keeps track of your current working directory (CWD). As you move around in the file system, it is useful to display your current location.

To find out your CWD, type the following command:

```
:showvar hpcwd
HPCWD = /SYS/PUB/Directory1/Directory2
```

The pathname of the directory is presented from the root directory down to the current location.

You can change your prompt so that it shows your current working directory at all times by including the variable in the prompt. Refer to the section "Change your prompt to show the current working directory" in Chapter 1.

Listing Space Used by Directories

Use the DISKUSE command to display the amount of disk space used by a directory. You must have traverse directory entries (TD) and read directory entries (RD) permissions to directories you want to report on.

Note

The REPORT command does not provide detailed line items for directories below accounts.

For example:

```
:DISKUSE /CYC
             SECTORS
        TREE       LEVEL   DIRECTORY
                   BELOW
         950         200   /CYC/
```

This command displays the disk space used by directory /CYC. Each of the columns contains information about the directory:

TREE	contains the total number of sectors used by the directory listed. This includes space used by the directory itself, all files in the directory, and all space used by subdirectories and files in the subtree below the directory.
LEVEL BELOW	shows the number of sectors allocated to all files and directories one level below the named directory. It does not include the space used by the directory itself or by the rest of the subtree.
DIRECTORY	displays the name of the directory in HFS format (for example, /MYACCT/MYGROUP).

If you add up all of the tree'd items you will sum to the total.

You can also report the disk space used by a directory and all subtrees below it. If you specify a directory ending with a / and use HFS syntax or use the TREE option, the command reports the amount of space used by it and all directories in it.

For example:

```
:DISKUSE /CYC/CELESTE/
          SECTORS
     TREE      LEVEL   DIRECTORY
               BELOW
      160+       128   /CYC/CELESTE/dir1/subdir1/
      384        224   /CYC/CELESTE/dir1/
      192+         0     (files directly below specified directory)
      608        224   /CYC/CELESTE/(32+)
```

This command displays the disk space used by the directory /CELESTE and all the subdirectories: dir1 and dir2. Including the TREE option has the same effect as including a slash at the end of a pathname. For example:

```
:DISKUSE /CYC/CELESTE ;TREE
```

The output is the same as that for the last example.

Sometimes the number of sectors under the TREE column are followed by a plus sign. This means that the amount refers to directories that are one level below the target directory.

The DISCUSE system-provided UDC works exactly like the DISKUSE command except that it warns you to use the DISKUSE command in the future.

Refer to the *MPE/iX Commands Reference Manual, Vol. I* (32650-90003) for details on the syntax of the `DISKUSE` command and the `DISCUSE` UDC.

Using Symbolic Links

You have a lot of flexibility in naming directories and files by using symbolic links. What if you want to move a set of files from one location to another without affecting the normal processing of any application that accesses those files. To do this, you can move the actual files to the new location, and in the old location create symbolic links with the same name specifying the new location of the files.

Creating symbolic links

Suppose that files `file1`, `file2`, and `file3` originally existed under the MPE group `PXGROUP` of account `DEVELOP`. You have decided to link the files to the `/users/denis/bin/FILES` directory. To ensure that all of the applications that access those files will still function properly, you can create symbolic links to those files in the old directory `/DEVELOP/PXGROUP`.

Use the `NEWLINK` command to create the symbolic links.

```
:rename ./file1, /users/denis/bin/FILES/file1
:rename ./file2, /users/denis/bin/FILES/file2
:rename ./file3, /users/denis/bin/FILES/file3
:chdir /DEVELOP/PXGROUP
:newlink ./file1, /users/denis/bin/FILES/file1
:newlink ./file2, /users/denis/bin/FILES/file2
:newlink ./file3, /users/denis/bin/FILES/file3
```

From this point on, anytime an application accesses these files by their old names, the symbolic links will redirect the file system to the new location of the files.

Note

This only applies to commands that operate on the target of the links and not the links themselves. For example, `PURGELINK` and `STORE` operate on the link itself and not the target files.

You can use this method to install newer versions of these files in another location without overlaying their current version, and changing the symbolic links to point to the newer version. If at any time you need to access the old version, you can point the symbolic links to that version of the files.

Deleting symbolic links

Use the `PURGELINK` command to delete a symbolic link.

```
:purgelink ./file1, /users/denis/bin/FILES/file1
```

The above example removes the symbolic link file `file1` in the CWD.

Renaming symbolic links

Symbolic links can be renamed by calling the POSIX C-library function `rename()`.

The MPE/iX `RENAME` command does not rename the symbolic link itself, it renames the file pointed to by the symbolic link. You can use the shell's `MV` command to rename a link.

Archiving symbolic links

Symbolic links can be stored and restored to your MPE/iX system by using the MPE/iX `STORE` and `RESTORE` commands like any other file on the system.

7

Managing Files

A **file** is an object that can be written to and read from. A file has certain attributes including access permissions and file type.

Directories are also files. They are special files that contain directory entries. A directory entry associates a name with an object.

Traditionally, MPE has restricted file creation to MPE groups within a user's logon account. On MPE/iX, authorized users can create files within and outside of their logon accounts.

You can create files in MPE groups and accounts, directly under the root directory, and in hierarchical directories.

MPE/iX File Naming Conventions

MPE/iX has only one file system but it supports two naming conventions. In MPE/iX, files can be named using the traditional MPE conventions or according to the new expanded HFS syntax. Some of the files can be named using either naming convention.

In the HFS syntax, a **file name** can be up to 255 characters. The valid characters are uppercase (A-Z) and lowercase letters (a-z), integers (0-9), the dot (.), the underscore (_), and the hyphen (-); the first character cannot be a hyphen. File names are case sensitive. The pathname limit is 1023 characters.

Note

Although the file name components can be up to 255 characters, other system limits restrict file name length. The command interpreter limits the number of characters that you can type in a command to 511 characters. Names of files directly under root or directly under a group or account are limited to 16 characters. Refer to Tables 2-1, 2-2, and 2-3 in Chapter 2 for additional naming restrictions.

A pathname identifies a file. It has a beginning slash (/), followed by the names of directories separated by slashes that lead to the file location. For example, the pathname for a file called `USERINFO` in the `PUB` group of the `SYS` account would be `/SYS/PUB/USERINFO`.

Table 7-1 shows sample file names and MPE/iX's interpretation of the names. In the table, `./` refers to the current working directory using the default MPE-Escaped syntax.

Table 7-1. MPE-Escaped Syntax File Name Resolution Examples

File Name Used	MPE/iX CI Interpretation
`a.b.c`	`A.B.C` or `/C/B/A`
`a.b`	`A.B.`*logonacct* or `/logon/B/A`
`a`	`A` or `./A`
`./a`	`./a`
`./a.b`	`./a.b`
`.a`	`./.a`
`..a`	`./..a`
`/a/b/c`	`/a/b/c`

To refer to files named using HFS syntax, you need to preface the name with a dot (`.`) or a slash (`/`). For example, to refer to the file `prog.src` in your current working directory, you need to refer to it as `./prog.src`. The name `/env_stats` refers to a file called `env_stats` in the root directory.

Byte-Stream Files

MPE/iX has been enhanced to support **byte-stream files**. Byte-stream files do not have a system-defined record structure. Information is stored as a "stream of bytes." By convention, a newline character divides lines of data. This file structure is compatible with the file structure on systems that use byte-stream files, such as HP-UX.

MPE utilities act on byte-stream files as if they are variable-length record format files; the utilities then save byte-stream files as variable-length record format files. This is also true for some MPE applications. To successfully treat byte-stream files as variable-length record format files, the byte-stream files must include newline characters at regular intervals.

Editing byte-stream files

MPE/iX emulation allows all tools to see Variable and Fixed files as a byte-stream. Therefore, the Shell can now manipulate the data in these two types of files as though they were byte-stream files. A program which does not purge the file can modify the byte-stream file with the emulator and get a byte-stream file.

To edit byte-stream files, you can use one of the following methods:

- Use the *vi* editor in the MPE/iX shell.
- Use editors such as `HPEDIT`.

- Convert the byte-stream file into a record-oriented file (for example, using `FCOPY`).

Editing byte-stream files using the vi editor

The `vi` editor is a standard UNIX editor that the POSIX .2 shell provides. Refer to the "VI Tutorial" in the *MPE/iX Shell and Utilities User's Guide* (36431-90002) or to *The Ultimate Guide to the vi and ex Text Editors* (97005-90015) for information on how to use `vi`.

Creating Files with HFS Syntax

To create a file with an HFS name or a file in an HFS directory, you can use the `BUILD` command. For example,

```
:BUILD ./.a
```

or

```
:BUILD .a
```

Note that any file moved into an HFS directory is assigned an ACD automatically if it doesn't already have one.

Most often, files using HFS syntax are created and managed by HFS-aware applications using programmatic interfaces.

Renaming Files

You can change the names of files using the `RENAME` command. When renaming a file across directories, you need to have CD permission to the file's new parent directory, DD permission to the file's old parent directory, and TD permission to all directories.

You can use either HFS or MPE syntax in the `RENAME` command. File names using HFS syntax must begin with a dot (`.`) or slash (`/`).

For example:

```
:rename /users/public/hearing.fil,./hearing.doc
```

This example renames the file `/users/public/hearing.fil` to `hearing.doc` in the current working directory.

```
:rename /dir2/doc/print.es, MYFILE.PUB.SYS
```

This example renames the file `/dir2/doc/print.es` to `MYFILE` in the `PUB` group of the `SYS` account.

```
:rename SURVEY.PUB, ./jan/survey
```

This example renames the file `SURVEY` in the `PUB` group and in the logon account, to `survey` in the `jan` subdirectory of the current working directory. Note that the directory `jan` must already exist for this example to work properly.

The following security considerations apply when renaming files:

- When renaming files, if a file has an ACD, the ACD remains the same for the renamed file.
- If you rename a file with no ACD to another group in the same account it will not be assigned an ACD.
- If you rename a file having no ACD from one MPE account to another account, to an HFS directory, to the account, or to root, the system assigns an ACD

that is an interpretation of the file access matrix in effect for the original file. For example, @.@ access is interpreted as any user +RACD; $GROUP and $GROUP MASK access is assigned to any user, AC user, and RACD; $OWNER receives all access + RACD.

Table 7-2 summarizes file security changes that occur at the file level when you rename files on MPE/iX. You must have the appropriate file access permission to rename files.

Table 7-2. Resulting Security When Renaming Files

From	To	Resulting Security
`file1.group1.acct1`	`file2.group1.acct1`	Same as original file.
`file1.group1.acct1`	`file2.group2.acct1`[2]	Same as original file. Note that the group security may be different.
`file1.group1.acct1`	`file2.group2.acct2`[1, 2]	If `file1` has no ACD, an ACD is assigned based on the file security matrix of the original location. If `file1` has an ACD, the ACD is not changed.
`file1.group1.acct1`	`/ACCT1/GROUP1/dir1/file2`[1, 2]	If `file1` has no ACD, an ACD is assigned based on the file security matrix of the original location. If `file1` has an ACD, the ACD is not changed.
`/ACCT1/GROUP1/dir1/file1`	`/ACCT1/GROUP1/dir2/file2`[2]	Same (original file has an ACD that is not changed).
`/ACCT1/GROUP1/dir1/file1`	`/ACCT2/GROUP2/dir2/file2`[2]	Same (original file has an ACD that is not changed).
`/file1`	`/file2`	Same (original file has an ACD that is not changed).

1. Some users who can access `FILE1` in `ACCT1` may not be able to access the file in its new location.
2. To access a file, you need TD entries access. So some users previously able to access the file may not be able to access the file in the new location.

For example, you may want to rename a file named `STATS` and move it into a hierarchical directory called `/SYS/PUB/FY92`. Assuming you are in `PUB.SYS`, you can perform the following steps to accomplish this.

- List the file and check what type of security it has.

```
:LISTFILE STATS.PUB.SYS.4
```

- Create the `FY92` directory in `PUB.SYS`.

```
:NEWDIR FY92
```

- Rename the file into the `FY92` directory.

```
:RENAME STATS,./FY92/STATS
```

- List the file again and see what the security looks like.

```
:LISTFILE ./FY92/STATS,4
```

The following figure shows how the file looks after it is first created.

```
:listfile stats,4
************************************
FILE: STATS.GROUP.ACCT

ACCOUNT ------  READ : ANY
               WRITE : AC
              APPEND : AC
                LOCK : ANY
             EXECUTE : ANY

GROUP --------  READ : ANY
               WRITE : GU
              APPEND : GU
                LOCK : ANY
             EXECUTE : ANY
                SAVE : GU

FILE ---------  READ : ANY           FCODE: 0
               WRITE : ANY         **SECURITY IS ON
              APPEND : ANY           NO ACDS
                LOCK : ANY
             EXECUTE : ANY

FOR OPERATOR.SYS: READ, WRITE, EXECUTE, APPEND, LOCK
```

After you rename the file into an HFS directory called /ACCT/GROUP/dir1, executing a LISTFILE command from that directory shows that an ACD has been assigned to the file:

```
:chdir /ACCT/GROUP/dir1
:listfile ./stats,4
************************************
FILE: /ACCT/GROUP/dir1/./stats

ACCOUNT ------  READ :
               WRITE :
              APPEND :
                LOCK :
             EXECUTE :

GROUP --------  READ :
               WRITE :
              APPEND :
                LOCK :
             EXECUTE :
                SAVE :

FILE ---------  READ :              FCODE: 0
               WRITE :            **SECURITY IS ON
              APPEND :              ACD EXISTS
                LOCK :
             EXECUTE :

FOR OPERATOR.SYS: READ, WRITE, EXECUTE, APPEND, LOCK, RACD
```

You can display the ACD using the `ACD` or `-2` option of the `LISTFILE` command:

```
:listfile ./stats,ACD
PATH= /ACCT/GROUP/dir1/

--------------ACD ENTRIES-----------------FILENAME
$OWNER                : R,W,X,A,L,RACD    stats
$GROUP_MASK           : R,X,L,RACD
$GROUP                : R,X,L,RACD
@.@                   : RACD
```

Listing Files

You can use the `LISTFILE` command to list files named using MPE and HFS syntax. Because files named using HFS syntax can be long, the format used when listing these files with the `,2` and `,1` options shows the pathname as the last item on the line. Names are listed in alphabetical order.

The following examples illustrate using the `LISTFILE` command to list files using HFS syntax.

Note

You can also use the `PLISTF` system-provided UDC to list files using the same parameters as provided by the `LISTFILE` command. The system-provided UDCs must be cataloged to be available to you.

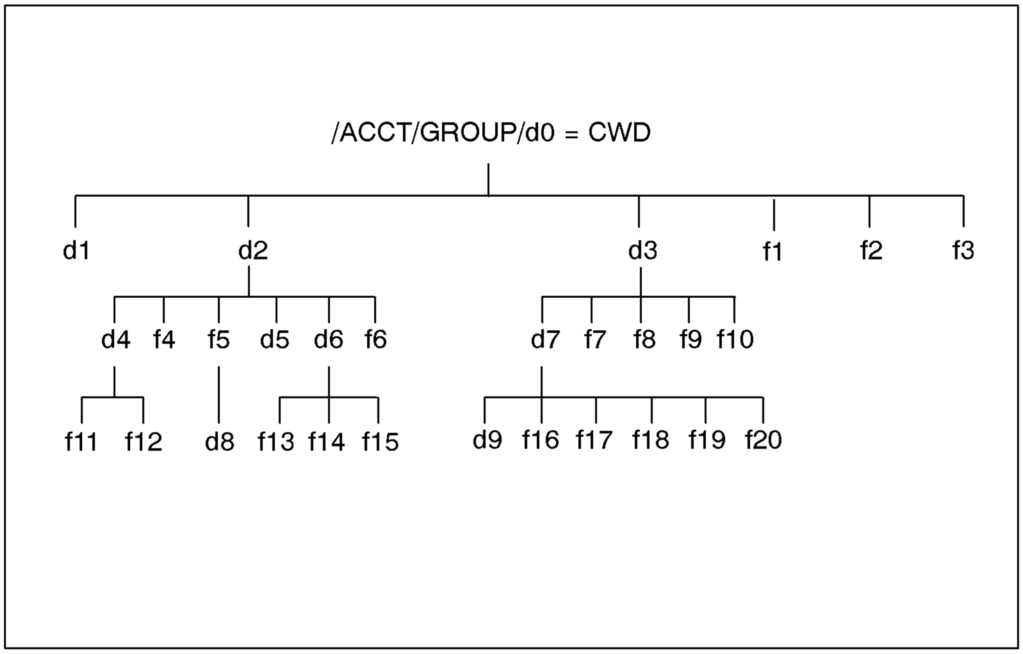
The following example lists all files and directories in the current working directory using the `PLISTF` UDC. The UDCs must be activated for you to use them. If you get a FILE NAME MISSING error message when you execute the following command, the UDCs are not available to you. Talk to your system manager about activating them.

```
hello me.official,gp1
plistf ./@

 PATH= /OFFICIAL/GP1/

 FILE1  dir2/ file1
```

Figure 7-1 illustrates an example hierarchical directory structure. In this figure, directory names are shown as the character `d` plus a number (for example, `d0`), and file names are shown as the character `f` plus a number (for example, `f1`). The examples following Figure 7-1 assume the directory structure shown. They also assume that the current working directory (CWD) is `/ACCT/GROUP/d0`.



LG200208_011

Figure 7-1. Example HFS File System

In the first example, the HPPROMPT variable is set to show the current working directory, the user changes directories using the CHDIR command, and requests a listing of all files one level below the CWD.

```
:hello manager.acct,group

:setvar hpprompt "!!hpcwd:"
/ACCT/GROUP:chdir ./d0
CWD is "/ACCT/GROUP/d0".
/ACCT/GROUP/d0:listfile ./@


 PATH= /ACCT/GROUP/d0/

 d1/ d2/ d3/ f1  f2  f3
/ACCT/GROUP/d0:
```

The next example also requests a listing of all files one level below the CWD using `FORMAT=2` (DISC) option.

```
/ACCT/GROUP/d0:<user|listfile ./@,2|
 PATH= /ACCT/GROUP/d0/

 CODE  ------------LOGICAL RECORD-----------  ----SPACE----
FILENAME
         SIZE  TYP        EOF      LIMIT R/B  SECTORS #X MX

          16W  DBH          4   67107839   1       64  2  *  d1/
          16W  DBH          4   67107839   1       64  2  *  d2/
          16W  DBH          4   67107839   1       64  2  *  d3/
          80B  FA          12         12   1       16  1  1  f1
          80B  FA          12         12   1       16  1  1  f2
          80B  FA          12         12   1       16  1  1  f3
```

The next example, the user requests a listing of all entries one level below the group by specifying the absolute pathname.

```
/ACCT/GROUP/d0:<user|listfile /ACCT/GROUP/@,2|
 PATH= /ACCT/GROUP/

 CODE  ------------LOGICAL RECORD-----------  ----SPACE----
FILENAME
        SIZE  TYP        EOF      LIMIT R/B  SECTORS #X MX

         16W  DBH          4   67107839   1       64  2  *  *d0/

/ACCT/GROUP/d0:
```

In the next example, the user specifies the `NAME` parameter to request a listing of all entries with a name beginning with a lower case "d". The `FORMAT=6` (QUALIFY) option is used to show the absolute pathname of all HFS entries.

```
/ACCT/GROUP/d0:listfile /;name=d@;format=6
 /ACCT/GROUP/d0/
 /ACCT/GROUP/d0/d1/
 /ACCT/GROUP/d0/d2/
 /ACCT/GROUP/d0/d2/d4/
 /ACCT/GROUP/d0/d2/d5/
 /ACCT/GROUP/d0/d2/d5/d8/
 /ACCT/GROUP/d0/d2/d6/
 /ACCT/GROUP/d0/d3/
 /ACCT/GROUP/d0/d3/d7/
 /ACCT/GROUP/d0/d3/d7/d9/
```

The next example illustrates the use of the `OBJECT=ACCT` parameter to show all accounts on the system.

```
/ACCT/GROUP/d0:listfile/;name=@;seleq=[object=acct];format=6
 /ACCT/
 /SYS/
 /TELESUP/
 /TEST/
```

The next example illustrates the `OBJECT=GROUP` parameter to show all groups on the system.

```
/ACCT/GROUP/d0:listfile/;seleq=[object=group];format=qualify
 /ACCT/GROUP/
 /ACCT/PUB/
 /SYS/ALINE925/
 :
 /TELESUP/PUB/
 /TEST/PUB/
 /TEST/SPOOL/
 /TEST/SPOOLSTD/
 /TEST/TEMPLATE/

/ACCT/GROUP/d0:
```

The next example illustrates the use of the `OBJECT=DIR` parameter to show all directories on the system.

```
/ACCT/GROUP/d0:listfile/;seleq=[object=dir];format=qualify
 /ACCT/
 /ACCT/GROUP/
 /ACCT/GROUP/d0/
 /ACCT/GROUP/d0/d1/
 /ACCT/GROUP/d0/d2/
 /ACCT/GROUP/d0/d2/d4/
 /ACCT/GROUP/d0/d2/d5/
 /ACCT/GROUP/d0/d2/d5/d8/
 /ACCT/GROUP/d0/d2/d6/
 /ACCT/GROUP/d0/d3/
 /ACCT/GROUP/d0/d3/d7/
 /ACCT/GROUP/d0/d3/d7/d9/
 /ACCT/PUB/
 /SYS/
 /SYS/ALINE925/
 /SYS/ALINK925/
:
 /TELESUP/PUB/
 /TEST/PUB/
 /TEST/SPOOL/
 /TEST/SPOOLSTD/
 /TEST/TEMPLATE/

/ACCT/GROUP/d0:
```

The next example illustrates a summary listing (format option 1) of all files in subdirectory `d3`.

```
/ACCT/GROUP/d0:listfile ./d3/@,1
 PATH= /ACCT/GROUP/d0/d3/

 CODE  ------------LOGICAL RECORD-------  FILENAME
         SIZE  TYP        EOF      LIMIT

          16W  HBD          4   67107839  d7/
          80B  FA          12         12  f10
          80B  FA          12         12  f7
          80B  FA          12         12  f8
          80B  FA          12         12  f9

/ACCT/GROUP/d0:
```

The next example illustrates a detail listing (format option 3) of all files in subdirectory `d3`.

```
/ACCT/GROUP/d0:<user|listfile ./d3/@,3|
 ********************
 FILE: /ACCT/GROUP/d0/./d3/d7/

 FILE CODE : 0                  FOPTIONS: DIRECTORY
 BLK FACTOR: 1                  OWNER   : **
 REC SIZE: 32(BYTES)            GROUP ID: **
 BLK SIZE: 32(BYTES)            SECURITY--READ    :
 EXT SIZE: 0(SECT)                        WRITE   :
 NUM REC: 4                               APPEND  :
 NUM SEC: 64                              LOCK    :
 NUM EXT: 2                               EXECUTE :
 MAX REC: 67107839                      **SECURITY IS ON
                                FLAGS   : NO ACCESSORS
 NUM LABELS: 0                  CREATED : TUE, JUL 21, 1992,  2:20 PM
 MAX LABELS: 0                  MODIFIED: TUE, JUL 21, 1992,  2:23 PM
 DISC DEV #: 1                  ACCESSED: WED, JUL 22, 1992, 12:05 PM
 SEC OFFSET: 0                  LABEL ADDR: **
 VOLCLASS  : MPEXL_SYSTEM_VOLUME_SET:DISC
 ********************
&vellipsis;
 FILE: /ACCT/GROUP/d0/./d3/f9

 FILE CODE : 0                  FOPTIONS: ASCII,FIXED,NOCCTL,STD
 BLK FACTOR: 1                  OWNER   : **
 REC SIZE: 80(BYTES)            GROUP ID: **
 BLK SIZE: 80(BYTES)            SECURITY--READ    :
 EXT SIZE: 13(SECT)                       WRITE   :
 NUM REC: 12                              APPEND  :
 NUM SEC: 16                              LOCK    :
 NUM EXT: 1                               EXECUTE :
 MAX REC: 12                            **SECURITY IS ON
 MAX EXT: 1                     FLAGS   : NO ACCESSORS
 NUM LABELS: 0                  CREATED : TUE, JUL 21, 1992,  2:21 PM
 MAX LABELS: 0                  MODIFIED: TUE, JUL 21, 1992,  2:21 PM
 DISC DEV #: 2                  ACCESSED: TUE, JUL 21, 1992,  2:21 PM
 SEC OFFSET: 0                  LABEL ADDR: **
 VOLCLASS  : MPEXL_SYSTEM_VOLUME_SET:DISC

/ACCT/GROUP/d0:
```

The next example illustrates the use of the `FORMAT=-3` option to show the owner. You must be the owner, or have AM or SM capability to use this option. Notice that the owner is now displayed as a fully-qualified user name.

```
listfile /ACCT/GROUP/@,-3

/ACCT/GROUP/d0:<user|listfile /ACCT/GROUP/@,-3|
 ********************
 FILE: /ACCT/GROUP/d0/

 FILE CODE : 0                  FOPTIONS: DIRECTORY
 BLK FACTOR: 1                  OWNER   : MANAGER.ACCT
 REC SIZE: 32(BYTES)            GROUP ID: ACCT
 BLK SIZE: 32(BYTES)            SECURITY--READ    :
 EXT SIZE: 0(SECT)                        WRITE   :
 NUM REC: 4                               APPEND  :
 NUM SEC: 64                              LOCK    :
 NUM EXT: 2                               EXECUTE :
 MAX REC: 67107839                      **SECURITY IS ON
                                FLAGS   : 1 ACCESSOR,SHARED
 NUM LABELS: 0                  CREATED : TUE, JUL 21, 1992,  1:10 PM
 MAX LABELS: 0                  MODIFIED: TUE, JUL 21, 1992,  2:16 PM
 DISC DEV #: 2                  ACCESSED: WED, JUL 22, 1992, 11:40 AM
 SEC OFFSET: 0                  LABEL ADDR: $000000E1
$0009A220
 VOLCLASS  : MPEXL_SYSTEM_VOLUME_SET:DISC

/ACCT/GROUP/d0:
```

The next example illustrates the use of the `FORMAT=4` (SECURITY) option to display the security matrix for all objects one level below the group (in this case, `d0`).

```
/ACCT/GROUP/d0:listfile /ACCT/GROUP/@,4
 ********************
 FILE: /ACCT/GROUP/d0/

 ACCOUNT ------  READ :
                WRITE :
               APPEND :
                 LOCK :
              EXECUTE :

 GROUP --------  READ :
                WRITE :
               APPEND :
                 LOCK :
              EXECUTE :
                 SAVE :

 FILE ---------  READ :                FCODE: 0
                WRITE :              **SECURITY IS ON
               APPEND :                ACD EXISTS
                 LOCK :
              EXECUTE :

 FOR MANAGER.ACCT: RACD, TD, RD, CD, DD
/ACCT/GROUP/d0:
```

The next example illustrates the use of the `FORMAT=-2` (ACD) option to display the access contol definition (ACD) for file `f4` in subdirectory `d2`. Note that all users (@.@) have read ACD (RACD) access for this file, which implies that no one has any other access to the file.

```
/ACCT/GROUP/d0:listfile ./d2/f4,-2
  PATH= /ACCT/GROUP/d0/d2/

  -----------ACD ENTRIES--------------- FILENAME

  @.@               : RACD            f4

/ACCT/GROUP/d0:
```

Displaying File Contents

You can display file contents using the PRINT command.

The following command prints the last 10 records of the file called print.myjob in doc of posix of the current working directory.

```
print ./posix/doc/print.myjob,,-10
```

File Equations and UDCs

As of Release 4.5 and 5.0, the operational components of MPE appear unchanged from a system management perspective. Existing UDCs, command files, and programs function identically on MPE/iX as they do on earlier releases.

However, if you plan to create files and directories using HFS syntax, you should review any UDCs and file equations that you have set up to be sure that they will not conflict with HFS syntax. For example, if name qualification is performed by a UDC or CI script, be sure that longer names (if used) will be acceptable in the commands.

Volume Set Usage

Files can reside on a user volume set. You must mount the volume set and set it up as the home volume set using an MPE group. HFS files on the UV can only exist as a descendent of an MPE group.

Determining File Types

Before you try to work with a file created using HFS syntax, you may need to determine what type of file it is. Use the `LISTFILE` command with the `,2` option to list compete information about a file. The `TYP` field indicates the type of file it is and whether it is a directory.

Using `LISTFILE, 2` produces output such as the following:

```
<user |listfile /OFFICIAL/GP1,2|
  PATH= /OFFICIAL/

 CODE  ------------LOGICAL RECORD-----------  ----SPACE----
FILENAME
         SIZE  TYP        EOF      LIMIT R/B  SECTORS #X MX

           0   GBD          0          0   0       64  2  *  GP1/
```

The `TYP` field uses letters to represent types of files. In the example shown here, the `TYP` field contains `GBD` indicating that it is a group directory.

Table 7-2 defines letter codes that can appear in the `TYP` field.

Table 7-3. File and Directory Types

TYP Code	**Description**
First Column	*Record Type*
A	Account directory
B	Byte-stream "records"
F	Fixed-length records
G	Group
H	Hierarchical directory
R	Root directory
U	Undefined-length records
V	Variable-length records
Second Column	*ASCII or Binary Format*
A	ASCII
B	Binary
Third Column	*CCTL Indicator*
C	CCTL (carriage control)

Table 7-3.
File and Directory Types (continued)

TYP Code	**Description**
Fourth Column	*File Type*
D	Directory (root, account, group, or hierarchical)
K	KSAM XL files
L	Symbolic link file
M	Message files
O	Circular files
R	RIO files
S	Spool file
d	Device link file
f	FIFO file
s	Streams

Note

If the file was not created with the CCTL option, the File Type indicators will be displayed in the third column instead of the fourth.

Copying Files

You can use the `COPY` command to copy any MPE/iX files. If you are granted sufficient access, you can copy files outside your current logon account.

You can also copy files to and from directories by using HFS syntax with COPY. Files named using HFS syntax must begin with (.) or (/). Remember that HFS syntax is case sensitive.

If you copy a file that has an ACD assigned to it, the ACD is copied along with the file. You can change the ACD on the copied file so it has the ACD you want by using the `ALTSEC` command. Refer to the `ALTSEC` command in the *MPE/iX Commands Reference Manual, Vol. I* (32650- 90003).

If you copy a file that has no ACD into an HFS directory, the file is automatically assigned an ACD. By default, the ACD assigned is `RACD:@.@` (read ACD access for all users on the system).

Note

The default ACD assigned when copying files into an HFS directory may not match the original file's access matrix. You may want to use the `ALTSEC` command to change the ACD of the copied file to set up the desired security provisions on the file.

The `TO=` parameter of the `COPY` command has always allowed `.GROUP` to be specified as a shortcut for `sourcefile.GROUP`. To remain consistent, a leading (.) in the `TO=` parameter only is *not* interpreted as an HFS name.

Table 7-4 summarizes file security changes that occur at the file level when you copy files on MPE/iX.

Table 7-4. Resulting Security When Copying Files

From	To	Resulting Security[1, 2]
`FILE1.GROUP1.ACCT1`	`FILE2.GROUP1.ACCT1`	Same as original file
`FILE1.GROUP1.ACCT1`	`FILE2.GROUP2.ACCT1`[3]	If `FILE1` has no ACD, the copied file takes on the security assigned to `GROUP2`. If `FILE1` has an ACD, it is copied to `FILE2`.
`FILE1.GROUP1.ACCT1`	`FILE2.GROUP2.ACCT2`[3]	If `FILE1` has no ACD, the file security matrix remains the same. Note that the group security may be different for `GROUP2`. If `FILE1` has an ACD, `FILE2` is assigned the same ACD.
`FILE1.GROUP1.ACCT1`	`/ACCT1/GROUP1/dir1/file2` [3]	If `FILE1` has no ACD, an ACD is assigned (usually `RACD:@.@`). If `FILE1` has an ACD, `file2` is assigned the same ACD.
`/ACCT1/GROUP1/dir1/file1`	`/ACCT1/GROUP1/dir2/file2` [3]	Same (original file has an ACD that is the same as that of the copied file)
`/ACCT1/GROUP1/dir1/file1`	`/ACCT2/GROUP2/dir2/file2` [3]	Same (original file has an ACD that is the same as that of the renamed file)
`/file1`	`/file2`	Same (original file has an ACD that is the same as that of the copied file)

1. The copied file takes on the GID of the parent directory and the UID of the person doing the copy.
2. The ACD may not change on the copied file, but the way it is evaluated may change if the UID and/or the GID of the file changes and the ACD has a `$OWNER` or `$GROUP` entry, respectively.
3. To access a file, you need TD entries access. So some users previously able to access the file may not be able to access the file in the new location.

Copying a file from an account to a directory

To copy a file from an account to a directory:

- Use the `COPY` command. For example:

```
:COPY LETTER.PUB.SYS, ./xfer
```

The file named `LETTER.PUB.SYS` is copied to `xfer` in the current working directory. The file `xfer` will only be recognized in lowercase. If you try to list it using `LISTFILE xfer`, MPE/iX displays an error

(`Non-existent file`). To list it, you need to specify HFS syntax:

```
:LISTFILE ./xfer

PATH= /SYS/PUB/

./xfer
```

Copying a file from a directory to an account

To copy a file from a directory to an account:

- Use the `COPY` command:

  ```
  :COPY ./xfer, LETTER.PUB.ACCT
  ```

The file named `xfer` (in lowercase) in the current working directory is copied to `LETTER.CUB.ACCT` (in uppercase) in the current account.

Copying a file from one directory to another

To copy a file from a directory to a second directory:

- Use the `COPY` command:

  ```
  :COPY /dir0/file1,/dir1/file2
  ```

The file `file1` in `dir0` in the root directory is copied to `file2` in `dir1` also in the root directory. These files are in directories outside of accounts and groups.

Deleting Files

You can use the `PURGE` command to delete files from the system. You must have DD access to a file's parent directory to delete a file.

For example, to delete the file called `print.doc` in the directory named `DOC` which resides in the directory called `posix` in your current working directory, enter:

```
:PURGE ./posix/DOC/print.doc
```

Storing Files

To back up all files on the system, enter:

```
:store /
```

Refer to Chapter 8 for detailed information on system backup.

Using Symbolic Links

Symbolic links are permanent MPE/iX files that can be created, removed, and renamed.

Creating symbolic links

You can use the NEWLINK command to create a symbolic link to a file, group, account, root, or another directory. Suppose that you log on as follows:

```
:hello dmarcon, engineer.develop, pxgroup
```

You can create a symbolic link SYMLN and associate it with the target file TAXINFO.FEB92.PAYROLL as follows:

```
:newlink symln, taxinfo.feb92.payroll
```

or

```
:newlink ./SYMLN, /PAYROLL/FEB92/TAXINFO
```

This creates a file called SYMLN under group PXGROUP of account DEVELOP. Assuming that you (that is, dmarcon) have the appropriate access to the target file (i.e., taxinfo), you can use the PRINT command to examine the contents of the target file as follows:

```
:print symln
```

instead of

```
:print taxinfo.FEB92.payroll
```

Deleting symbolic links

You can use the PURGELINK command to remove a symbolic link. This command takes the name of the symbolic link and deletes it without affecting the target of that link. For example, if you (that is, dmarcon) wanted to remove the symbolic link SYMLN, you would do the following:

```
:purgelink symln
```

or

```
:purgelink ./SYMLN
```

Renaming symbolic links

Symbolic links can be renamed by calling the POSIX C-library function of rename().

The MPE/iX command RENAME does not rename the symbolic link itself, but the file pointed to by the symbolic link.

Archiving symbolic links

Symbolic links can be stored and restored to your MPE/iX system by using the MPE/iX STORE and RESTORE commands like any other file on the system.

8

Backing Up and Recovering MPE/iX

Enhancements to the file system allow files to be stored at every level of the directory hierarchy, such as below the root directory, within directories and subdirectories, and within directories subordinate to groups. Changes were made to the `STORE` and `RESTORE` commands so that all of the directories and files on the system can be backed up.

This chapter generally describes the changes to the `STORE` and `RESTORE` commands as a result of the introduction of the hierarchical file system. It is especially important for system administrators to be aware of these changes. For the complete syntax of the `STORE` and `RESTORE` commands, refer to the *MPE/iX Commands Reference Manual, Vol. II* (32650-90374).

Backing Up MPE/iX Files

Valid file names that use MPE syntax are also valid in HFS syntax; however, the converse is not necessarily true. For example, the MPE file name `STORE.PUB.SYS` maps to `/SYS/PUB/STORE` in the HFS syntax; but the file name `/tmp/store` is not valid in the MPE name space (because HFS syntax is case sensitive but MPE syntax is not).

File name mapping for full backups

To ensure that all files on a system get backed up, certain MPE syntax names get mapped to HFS syntax. Anytime the filename, group, and/or account part of an MPE syntax name is equal to `@`, the name will be mapped to an equivilent HFS name. This mapping is only done for inclusion file sets, it is not done for exclusion (negative) file sets.

Specifying MPE/iX files

You can store or restore file sets that use MPE or HFS syntax by using the `STORE` or `RESTORE` commands, respectively. You can also use an indirect file. Following are the rules for specifying MPE/iX file sets in the `STORE` and `RESTORE` commands:

- Separate multiple file sets with a ",".
- Specify exclusions from a file set by including a "-" symbol before the file set (negative file set).
- Since "-" is a valid HFS file name character, make sure that it is preceded by a blank so that it is recognized as a negative file set symbol in an HFS syntax file set.

You can use wildcard characters to specify the MPE/iX file set that is to be stored. After the file set is syntactically verified, it is expanded into the list of files to be stored.

Use the `STORE` and `RESTORE` syntax to specify the following HFS file sets:

- all files directly under a specified directory (horizontal cut)
- all files and directories at all levels below and including a specified directory (a recursive scan)
- hierarchical directory structure (the entire file system)

The pathname component at every level in the file set pathname expands to a horizontal cut (matching the wildcard) at that level. In addition, if the file set has a trailing "/", then all of the directory matches at the last component of the specified set are scanned recursively.

`STORE` and `RESTORE` also provide two overriding options: `TREE` and `NOTREE`. These options override the trailing slash on the file set. The `TREE` option forces every file set to be scanned recursively from the specified level, including any levels beneath it; the `NOTREE` option forces every file set to be scanned for a horizontal cut only.

You can also exclude certain files from the file set by preceding the name with a minus sign. An unlimited number of exclusion file sets may be specified.

Table 8-1 shows examples of valid MPE/iX file sets for `STORE` and `RESTORE`.

Table 8-1. Example File Sets for STORE/RESTORE

File Set	Contents
`?@.@.@`	All files and directories directly under MPE groups whose names conform to MPE syntax
`./file1`	`file1` in the current working directory
`./@ or @`	All files and directories directly below the current working directory
`?@`	All files and directories in the current working directory whose names conform to MPE syntax
`./[aA]@`	All files or directories in the current working directory whose names begin with "a" or "A"
`./dir1/@`	All files or directories located in the subdirectory `dir1` in the current working directory
`/SYS/PUB/@ or @.PUB.SYS`	All files or directories in directory `/SYS/PUB`
`/@/@/@`	All files or directories three levels below the root directory

Table 8-1. Example File Sets for STORE/RESTORE (continued)

File Set	Contents
`./dir1/a@,` `./dir1/@a`	From the current working directory, all files and directories in `dir1` beginning or ending with "a"
`./dir1/@ -` `./dir1/a@`	From the current working directory, all files and directories in `dir1` except those beginning with "a"
`/@/@/@ -` `/SYS/PUB/@`	All files and directories exactly three levels beneath the root directory except the files and directories in `/SYS/PUB`
`/ or @.@.@`	All files and directories on the system (beginning with the root directory and moving down through the entire file system)
`./dir1/`	From the current working directory, all files and directories beneath and including `dir1`
`./dir1/a@, ./dir2/`	From the current working directory, all files and directories beginning with "a" directly beneath `dir1`; and all files and directories at all levels beneath and including `dir2` ofthe current working directory
`./d1/ -` `./d1/dir2/@`	All files and directories at all levels beneath and including `d1`, except for those files and directories that are in subdirectory `dir2`
`./dir@/ -` `./dir1/dir2/`	From the current working directory, all files and directories beneath directories matching dir@ (such as `dira`, `dirb`, etc.), except the files and directories under subdirectory `dir1/dir2` (or subdirectory `dir2` of `dir1` of the current working directory)
`/ - @.@.@`	All files and directories that are outside of MPE syntax

Backing up the file system

To back up the entire hierarchical file system including all MPE-named files and HFS-named files, you can still use the following command:

```
:STORE @.@.@
```

You can also use:

```
:STORE /
```

This is the HFS syntax to perform the same task.

It is recommended that `STORE /` should be used in place of the command `STORE @.@.@` when performing a full system backup, although `STORE @.@.@` will be mapped to include all files on the system.

Additional backup examples

The following example backs up all of the files and directories in the account called `MKTG` and sends a listing of the backed up files and directories to `LP`. In the example, including the slash after the account name (as in `/MKTG/`) is the same as specifying the `TREE` option with the `STORE` command. It causes the `STORE` command to store all files and HFS directories in the `MKTG` account and its subtree.

```
FILE TAPEDEV; DEV=TAPE
STORE /MKTG/; *TAPEDEV; SHOW=OFFLINE
```

The following example produces a complete system image on the backup media and sends a listing of the backed up files to `LP`.

```
FILE TAPEDEV; DEV=TAPE
STORE /; *TAPEDEV; SHOW=OFFLINE; DIRECTORY
```

The file set “/” specifies the root directory, which is scanned recursively. This backs up all files on the system.

The `DIRECTORY` option saves the entire hierarchical directory structure along with the MPE/iX account and group information and UID/GID data. The directory information is used to recreate the hierarchical directory structure when these files and directories are regenerated using the RESTORE command.

The following example performs the same complete system image on the backup media:

```
FILE TAPEDEV; DEV=TAPE
STORE @.@.@; *TAPEDEV; SHOW=OFFLINE; DIRECTORY
```

The DATE>= option can be used to obtain an incremental backup of the files changed or modified since some_date as illustrated in the next example.

```
FILE TAPEDEV; DEV=TAPE
STORE /; *TAPEDEV; SHOW=OFFLINE; DIRECTORY; DATE>=some_date
```

Storing old files

The DATE<= and PURGE options can be used to archive files not accessed since some_date, as shown in the following example.

```
FILE TAPEDEV; DEV=TAPE
STORE /; *TAPEDEV; SHOW=OFFLINE; DATE<= some_date;PURGE
```

The PURGE option removes only the files that match the date and file specifications and does not remove the directory nodes.

What gets stored

For a horizontal cut (all files in a directory) or for the recursive scan (all files at all levels below the specified directory), the files and directories are stored together. However, if the DIRECTORY option is used with the STORE command, the directories in the specified file set are stored separately from the files.

The directories are collected and stored at the start of the media. This portion of the tape is used to regenerate the hierarchical directory structure as required at RESTORE time. (The root/account/group and user information is stored in a file to maintain backwards compatibility.)

Note that you must have SM or OP capability to use the DIRECTORY option.

Note

The default file set is "@". With the STORE command, if the DIRECTORY option is specified, the default file set is empty.

An unqualified file set such as "@" or "a@" is always expanded relative to the current working directory.

Storing of Symbolic Links

As of Release 5.0, MPE/iX supports symbolic links. STORE treats symbolic links in the following manner:

1. When the symbolic link is the final component of a pathname, the link will be stored and not the file to which the link points.

```
:NEWLINK /ACC/SLINK,/usr/jdoe/file
:STORE /ACC/SLINK;*tape;show
```

If you perform these commands, the symbolic link `/ACC/SLINK` is stored and the file `/usr/jdoe/file` is not stored.

2. When the symbolic link is an intermediate component of a pathname, the link will be traversed and files will be stored using absolute pathname.

```
:NEWDIR /usr/jdoe/dir
:NEWLINK /DENIS,/usr/jdoe/dir
:BUILD /DENIS/file1
:BUILD /DENIS/file2
:STORE /DENIS/@;*tape;show
```

The files `file1` and `file2` are stored and identified with the following names:

```
/usr/jdoe/dir/file1
/usr/jdoe/dir/file2
```

They can only be restored using these pathnames.

3. Intermediate symbolic links only match *literally*. The symbolic link is not expanded before checking for name matches.

```
:NEWDIR /usr/jdoe/dir
:NEWLINK /DENIS,/usr/jdoe/dir
:NEWLINK /JAIME,/usr/jdoe/dir
:BUILD /DENIS/file1
:BUILD /DENIS/file2
:BUILD /DENIS/file3
:BUILD /JAIME/file4
:STORE /DENIS/@ -/DENIS/file1 -/usr/jdoe/dir/file2 -/JAIME/file4;*t;show
```

If you perform the example above, the files `/usr/jdoe/dir/file3` and `/usr/jdoe/dir/file4` are stored. The files `/usr/jdoe/dir/file1` and `/usr/jdoe/dir/file2` are not stored. `/usr/jdoe/dir/file4` is stored because `/DENIS/file4` does not match `/JAIME/file4`. Therefore, `/JAIME/file4` excluded a file *not* in the store set.

Restoring MPE/iX Files

You can restore an MPE/iX file set by using the `RESTORE` command. The file set name syntax and scanning rules for `RESTORE` are exactly the same as for `STORE`. (Refer to "How to specify MPE/iX files" earlier in this chapter.)

The `CREATE` option of the `RESTORE` command has been enhanced to provide the `CREATE=PATH` option. This option is used to create the required paths for a `RESTORE`.

Note

The default file set for `RESTORE` continues to be file set `@[.@[.@]]` depending on whether the user has NONE, AM, or SM capabilities. If the `DIRECTORY` option is specified, the default file set is empty.

What Gets Restored

At `RESTORE` time, only files are restored by default. The directory matches are also restored if either the `DIRECTORY` or `CREATE=PATH` options is specified. If the `RESTORE` command specifies a file set including only MPE-named files, then only MPE-named files can be restored. If a `RESTORE` command specifies an HFS-named file set, then all files and directories are restored from the media. As with `STORE`, specifiying the wildcard character `@` for one filename portion of an MPE-named file will cause it to be mapped to include both HFS and MPE named files.

Reloading Hierarchical Directories

You can regenerate the hierarchical directory structure and the UID and GID databases by restoring the directory information stored on the media.

The following rules should be followed when regenerating the hierarchical directory structure:

- You need to specify the `DIRECTORY` option with the `RESTORE` command to regenerate the entire hierarchical directory structure. The entire directory structure is regenerated before restoring any files if the information is available on the media (that is, if the `DIRECTORY` option was used at `STORE` time). You must have either SM or OP capability to use the `DIRECTORY` option.
- If you use the `CREATE` option with the `RESTORE` command, all directories matching the specified file set are restored.
- If you use the `CREATE` option with the `RESTORE` command and the media is missing a directory that is required to restore a file, then the required directory is created with default security provisions.
- If you do not use the `CREATE` option with the `RESTORE` command, an error message is printed for every file that cannot be restored due to a nonexistent directory component.

9

Handling Security on MPE/iX

MPE/iX system security required enhancements to support the hierarchical file system. The new security features are integrated so that they cannot be used to bypass traditional MPE/iX security. All existing MPE security features continue to work as they have.

This chapter describes enhancements to security that support the hierarchical file system. The major areas affected by the changes are:

- User identification
- Directory access
- File access

These concepts are described in some detail in the beginning of the chapter. The latter part of the chapter presents tasks involving system security.

The information in this chapter is written for all system users who plan to use the new features of MPE/iX. It is particularly relevant to system administrators who need to control users' access to MPE accounts, groups, directories, and files.

User Identification

Users on MPE/iX are now identified by a user ID (UID). The UID is a string (in the form *user.account*) with a corresponding integer value. Each MPE account has a group ID (GID) associated with it. The GID is a string (in the form *account*) and also has a numerical value assigned to it. UIDs and GIDs were added to file and process structures to more easily identify object owners and file sharing groups, respectively. (Refer to Chapter 5 for more information about UIDs and GIDs.)

In addition to the UIDs and GIDs, users are identified as follows:

Table 9-1. User Categories

Category	Conditions
File Owner	The user whose UID matches the object's UID (matched by *user.account* or `$OWNER` in ACDs).
File Group Member	Any user whose GID matches the GID of the object (also called *@.account* or `$GROUP` in ACDs). This *group* is a new file sharing concept that should be distinguished from MPE groups (that is, group directories). By default, when a user creates a file or directory, it is assigned the parent directory's GID.
File Group Class Member	Any ACD entry except `$OWNER` and `@.@`.

Access Control Definitions (ACDs)

MPE/iX file system access is controlled by using access control definitions (ACDs) or the file access matrix. Refer to the *Manager's Guide to MPE/iX Security* (32650-90474) and *User's Guide to MPE/iX Security* (32650-90472) manuals for information on restricting file access by using the file access matrix and more details on ACDs. This chapter briefly introduces ACDs because of their relevance to controlling access to files and directories created outside of MPE groups.

Because ACDs are now required in some cases, it becomes increasingly important that you understand the MPE/iX ACD facility.

Note

ACDs are the main method of controlling access to files, hierarchical directories, and devices. ACDs are automatically assigned to hierarchical directories and to files existing outside of MPE groups.

What is an ACD?

ACDs are ordered lists of pairs. The pairs are made up of access permissions and user specifications that control access to **objects**. Objects are passive entities that contain or receive information, such as files, directories, and devices. Each entry in the ACD specifies object access permissions granted to a specific user or group of users. In addition to being granted access to an object protected by an ACD, users can also be granted access to read the ACD itself.

ACDs can be applied to any MPE/iX files using the `ALTSEC` command. This command was enhanced to support directories. If a file has an ACD, this method of specifying access to the file takes precedence over other security features, such as lockwords and the file access matrix. ACDs cannot be placed on root, account, group, or directories.

Access modes

ACDs control the ability to access and change hierarchical directories and the files within them. MPE/iX has enhanced the `ALTSEC` command to support access to directories. The available ACD access modes are as follows:

R	Read a file.
W	Write to a file.
L	Lock a file.
A	Append to a file.
X	Execute a file.
CD	Create directory entries.
DD	Delete directory entries.
RD	Read directory entries.
TD	Traverse directory entries.
RACD	Copy or read the ACD associated with the object.
NONE	Deny access.

You use the `ALTSEC` command to alter access modes for files, hierarchical directories, logical devices, or device classes. For more information about ACD access modes, refer to the `ALTSEC` command in the *MPE/iX Commands Reference Manual, Vol. I* (32650-90003).

User specifications

Beginning with MPE/iX Release 4.5, the MPE/iX access control definition (ACD) facility provides three new user specifications. In addition to specifying a user (*user.account*) or set of users (@.*account*) in a file or directory ACD, you can also use the following designators:

`$OWNER`	Specifies the file owner. The file owner is granted the access permissions specified by `$OWNER`. A user is a file owner if the user's UID matches the UID of the file.
`$GROUP`	Specifies the file group members of the file or directory. If the user's GID matches the GID of the file, the user is granted the access permissions specified by `$GROUP`.
`$GROUP_MASK`	Restricts all ACD entries except for `$OWNER` and @.@. In this case, if a user matches a *user.account* entry, `$GROUP` entry, or *@.account* entry, the user is granted an access permission if it appears in both the matching ACD entry and `$GROUP_MASK`. An ACD with a `$GROUP_MASK` entry must also have a `$GROUP` entry. `$GROUP_MASK` is provided to integrate the POSIX definition of security with the more robust security provided by MPE/iX ACDs.

These new user specifications modify the manner in which the file system checks access permissions when an ACD is associated with a file.

Required ACDs

Prior to Release 4.5, the MPE/iX ACD facility provided an optional security facility to replace MPE/iX standard file system security features. In the current release, ACDs are required on the following system objects:

- All hierarchical directories
- All files under hierarchical directories
- All files directly under MPE/iX groups where the file GID does not match the GID of the account and group in which the file is located.

How do ACDs work

The way ACDs are evaluated has changed as of Release 4.5 as a result of security enhancements. When you attempt to access a file, the system checks access permissions in the following order of precedence:

1. Do you have SM capability? If so, you are granted all access to the file.
2. Do you have AM capability and does your GID match the GID of the file? If so, you are granted all access to the file.
3. Are you the owner of the file (your UID matches the UID of the file)?
   a. If there is no ACD associated with the file, you are given all access permissions to the file and the checking ends.
   b. If there is an ACD associated with the file and there is no `$OWNER` entry, you are given all access permissions to the file and the checking ends.
   c. If there is an ACD associated with the file and that ACD contains the `$OWNER` entry, you are restricted to the access permissions assigned to `$OWNER`. (Since you are the file owner, you can always modify the ACD if you need more access permissions than provided by the `$OWNER` entry.)

   If you are not the owner of the file, the system performs the check described in step 4.
4. Is there an ACD assigned to the file? If there is no ACD assigned to the file, the system performs the checking described in step 5. If there is an ACD, the system performs the checking in the following order (from more specific to less specific):
   a. Does your UID match a specific user name entry (for example, `ALEX.TECHNLGY`). If so, you are granted the access permissions assigned to that entry unless a `$GROUP_MASK` entry exists. If the

`$GROUP_MASK` entry exists, the resulting access permissions are only those that are in both the *user.account* entry and the `$GROUP_MASK` entry. No further checking is performed.

   b. Does your GID match the GID of the file? If so, and a `$GROUP` entry exists, you are granted the access permissions assigned to that entry unless a `$GROUP_MASK` entry exists. If the `$GROUP_MASK` entry exists, the resulting access permissions are only those that are in both the `$GROUP` and the `$GROUP_MASK` entries. No further checking is performed.

      If you match the `$GROUP` entry and your GID matches the account portion of an *@.account* entry, you are granted the access permissions assigned to either ACD entry prior to `$GROUP_MASK` evaluation.
   c. Does your GID match the *account* portion of an *@.account* entry? If so, you are granted the access permissions assigned to that entry unless a `$GROUP_MASK` entry exists. If the `$GROUP_MASK` entry exists, the resulting access permissions are only those that are in both the `@.account` and the `$GROUP_MASK` entries. No further checking is performed.
   d. Does an `@.@` entry exist? If so, you are granted the access permissions assigned to that entry. No further checking is performed.
   e. If you match no ACD entries (or if the access mode assigned to you is NONE), you are granted no access to the file, and no further checking is performed.

5. If there is no ACD, the system uses the file access matrix to check for access permissions.

ACD examples

You assign ACDs using the ALTSEC command. In addition, files outside of MPE groups are automatically assigned ACDs.

Following is an example of an ACD that could be assigned to a text file:

```
NONE:JIM.DOE,@.ACCT;R,W,X,L:@.PAYROLL;R:@.@
```

The ACD pairs in this example set up the following access controls on the text file:

- Deny JIM.DOE and all users in the ACCT account access to the file.
- Allow read, write, execute, and lock access to users in the PAYROLL account.
- Allow read access to everyone else.

Notice that in cases of contradictions, the most specific ACD pair takes precedence. So even though all users are assigned read access (R:@.@), JIM.DOE cannot access the file because he is specifically assigned no access (NONE:JIM.DOE).

If the ACD in the above example had a $GROUP_MASK entry (for example, R,X:$GROUP_MASK), then the users in the PAYROLL account would only have read and execute access. The entire ACD would read as follows:

```
NONE:JIM.DOE,@.ACCT;R,W,X,L:@.PAYROLL;R:@.@;R,X:$GROUP_MASK
```

An example of an ACD for an HFS directory (dir1) follows:

```
CD,DD,RD,TD,RACD:@.ACCT;TD:@.@
```

The ACD pairs in this example set up the following access controls on dir1:

- Allow all users in the ACCT account the ability to create, delete, traverse, and read directory entries in dir1, and to read the ACD.
- Allow everyone else the ability to traverse dir1 only.

Controlling access to files and directories

New access attributes for ACDs have been added to support security for hierarchical directories. The access attributes associated with directories are as follows:

- CD—create directory entries
- DD—delete directory entries
- TD—traverse directory entries
- RD—read directory entries

Users need appropriate permission to access a directory and its contents. For example, the owner of a directory can grant *create directory entries (CD)* access to other users. Users can only create files or other directories within a directory if they have CD access to the directory.

RD access and TD access differ as follows. If a user wants to use `LISTFILE` to list the files in a directory, the user needs RD permission for that directory. But, if a user wants to access a file such as `/users/jeff/address`, the user needs to have TD permission for all the directories in the path; that is, `/`, `users`, and `jeff` in this case.

By default, all users can read the contents of and traverse the root directory, all MPE accounts, and all MPE groups. However, to create or delete the contents of a file, you must have the appropriate access permission to open the file itself.

Because the root, accounts, and MPE groups are special types of directories on MPE/iX, you cannot control access to them using ACDs. You cannot apply TD, DD, CD, or RD to MPE groups or accounts. You need to use existing mechanisms. For example, use the `ALTGROUP` command to change save access permissions for MPE groups.

Object creation

Creating an object, which is creating an entry for a file or directory within a directory, requires that a process have TD and CD access to the object's parent directory and SF capability. For an MPE group, SAVE access is equivalent to CD access (see "SAVE access in MPE groups").

Users with SM capability can create files and directories anywhere on the system. Users without SM capability can create files and directories outside their logon account in any directory that they can traverse to and to which they have been granted CD access.

Object deletion

To delete a file or subdirectory from a directory, you must have DD access to the directory. For files in MPE groups, you only need WRITE access to the file. For directories in MPE groups, you only need SAVE access to the MPE group. For more information, refer to "Deleting Directories" in Chapter 6 and "Deleting Files" in Chapter 7.

File renaming

Any user with the proper access can rename a file. To rename a file within the same directory or from one HFS directory to another, you must have both DD and CD access. DD is required to delete the old entry from the directory where the file resides, and CD is required to create the new directory entry. For more information, refer to "Renaming Files" in Chapter 7.

You can rename a file from one directory to another if you have DD access to the directory in which the file is located and CD access to the directory where you want the renamed file to reside.

Users with SM capability can rename files anywhere on the system. To rename a file from an MPE group in one account to an MPE group in another account, you must have SM capability.

If you rename a file that does not have an ACD from an MPE group to a directory that is not an MPE group, an ACD is automatically generated for it. This is required because the security matrix cannot protect the file any longer.

If you rename a file (that does not have an ACD) from an MPE group to another MPE group outside the original account, an ACD is automatically generated for it, because the file's GID would no longer match the parent group's GID and would not be protected by the file access matrix.

File owner

A file (or directory) owner has complete access to the file unless the user is restricted by a `$OWNER` ACD entry. Now that there is a `$OWNER` ACD entry, you can restrict the file access of the file owner.

For example, `MGR.PAYROLL` is the creator (owner) of the file `MYFILE`. On Releases 3.0 and 4.0, the owner's access cannot be restricted by an ACD or the file access matrix. So on Release 3.0 and 4.0 systems, `MGR.PAYROLL` still has all the access permissions on this file even if an ACD pair specifies only read permission (`R:MGR.PAYROLL`). As of Release 4.5, the access of the owner can be restricted by using the `$OWNER` ACD entry. Assigning `(R:$OWNER)` restricts the owner to having read permission only. However, a file owner can always modify the ACD, thus removing any restrictions specified by `$OWNER` when they are no longer necessary.

SAVE access in MPE groups

Create directory entries (CD) access and delete directory entries (DD) access to all MPE groups is governed by appropriate privileges or SAVE access. (A complete definition of appropriate privilege appears later in this chapter.) SAVE access for an MPE group implies CD and DD permission for directory entries. That is, a user can create or delete a directory in an MPE group if the group grants SAVE access to the user. However, you still need write access to a file, in addition to `SAVE` access, to be able to delete it from an MPE group. For more information, refer to "Creating and Naming Directories" in Chapter 6.

CWD and file security

You can change your current working directory (CWD) to any directory (including an MPE account, an MPE group, the root directory, or an HFS directory) as long as you have TD access to the directories in the path to the directory. This means that you can change your CWD to any MPE group on the system because all users have RD and TD access to the root directory, all accounts, and all MPE groups, by default.

It is important to note that changing your CWD to a new MPE group (using the `CHDIR` command) does not make you a GU user of the new group. GU is based on your logon group and account; this can only be changed using `CHGROUP`. If you attempt to access a file in the new group, you may not be able to access it. If the new group is in your logon account, you are allowed account level privileges (AC) in the new group. If the new group is not in your logon account, you are allowed the access privileges given to any user (ANY). No password check is done when you change your CWD. This is unlike `CHGROUP` which does a password check.

Appropriate Privilege

Appropriate privilege means that the user has sufficient capabilities to perform an operation even if the user is not explicitly granted the necessary access. The user's capabilities grant the correct access to the directory or file.

Appropriate privilege does not override file lockwords, privileged files, privileged file codes, or write-protected files.

System manager capability

Having SM capability provides appropriate privilege and allows the system manager (or those having SM) to override the file access matrix or ACD on any file or directory.

Users with SM capability can create files and directories anywhere on the system. Users with SM capability can also rename files anywhere on the system. To rename a file from an MPE group in one account to an MPE group in another account, you must have SM capability.

Account manager capability

If all objects in an account have the same GID, the traditional MPE model remains in effect. A user having AM capability for the account can access all of the files and directories within the account.

It is possible for objects within an account to have different GIDs if, for example, files are renamed or if the GID is changed programmatically. In this case, having AM capability will not be sufficient privilege to gain access to those files. The GID of the user with AM has to match the GID of the file or directory to allow access to it.

Execute (X) Access

The hierarchical file system does not provide a way to distinguish files containing executable scripts from other files. However, the POSIX standard requires that file permission bits should be checked to verify that execute access has been granted to at least one of the file classes as an indication that a file contains executable statements.

On MPE/iX, when all access would normally be granted to a user, X access is handled as a special case. Users with appropriate privilege are granted X access only if the file has an executable file code (PROG, SL, NMPRG, or NMXL), if the file access matrix assigns X access to at least one user class, or if the file has an ACD that assigns X access to at least one user.

The file owner is granted X access only if the `$OWNER` ACD entry grants X access. If the `$OWNER` entry does not exist, the file owner is granted X access if the file has an executable file code or at least one user is granted X access by the file access matrix or an ACD.

A RELEASEd file grants X access to all users.

These rules do not affect other uses of X access on the system, and they are backwards compatible with the use of X access on releases before Release 4.5. Users with appropriate privilege still get X access to files with executable file codes. X is also used to grant `STREAM` access to `JOB` files. Users with appropriate privilege can still stream these files because they have R access to the files.

Tasks Involving System Security

The following sections describe tasks relating to system security such as listing ACDs, assigning ACDs, changing ACDs, and copying ACDs.

Listing ACDs for files and directories

Because ACDs supersede other security mechanisms, it is useful to be able to determine whether or not a directory or file has an ACD assigned to it and, if so, what it is. Any directories or files residing outside of traditional MPE groups are automatically assigned ACDs when they are created. You can list ACDs by using the `LISTFILE` command with the `-2` (also called `ACD`) option.

The following example shows how to list the ACD associated with the directory called `letters.` Notice that the user named `JONES` in the `OFFICE` account has RD (read directory entries) access to the `letters` directory. All other users on the system have both RD and TD (traverse directory entries) access to `letters.`

```
listfile /dir0/letters,-2
 PATH=/dir0/

 ------------ACD ENTRIES-------------- FILENAME

     JONES.OFFICE        : RD           letters/
     @.@                 : RD,TD
```

In the next example, the directory `GRP` is assigned the default ACD. All users can read the ACD assigned to the directory. Only the creator and the system manager can change it. Also, note that `-2` is replaced with the textual equivalent `ACD`.

```
listfile /OFFICE/GRP,ACD
 PATH=/OFFICE/

 ------------ACD ENTRIES-------------- FILENAME

 @.@               : RACD              GRP/
```

In the next example, the file `assets` has an ACD assigned to it. The ACD pairs are listed from the most specific (such as a particular user in a particular account) to the least specific (all other users in all other accounts). User `ZONIS` in the `OFFICE` account has R (read) access to the file `assets.` Other users in the `OFFICE` account have both R and W (write) access to the file. And all other users in other accounts have R, W, and X (execute) access to the file.

```
listfile /OFFICE/GRP/assets,-2
 PATH=/OFFICE/GRP/

 ------------ACD ENTRIES-------------- FILENAME

 ZONIS.OFFICE        : R               assets
 @.OFFICE            : R,W
 @.@                 : R,W,X
```

The next example shows how you can list the ACDs for all of the files in the `GRP` directory. It shows the ACD on the file `assets` as in the previous example and lists the ACDs on the other two files in the directory.

```
listfile /OFFICE/GRP/@,-2
 PATH=/OFFICE/GRP/

------------ACD ENTRIES------------ FILENAME

ZONIS.OFFICE      : R               assets
@.OFFICE          : R,W
@.@               : R,W,X
ZONIS.OFFICE      : R               bills
WILKE.OFFICE      : R,W
@.@               : R,W,X
SMITH.OFFICE      : R               goods
@.OFFICE          : R,W,X
```

Changing access to files and directories

Because access to MPE/iX files and hierarchical directories is controlled by ACDs, system users may want to change the defaults assigned when files or directories are created.

For the purpose of selectively restricting access to files with ACDs, users can be classified into three groups:

- Individual users
- Specific groups of users
- All other users

Assigning ACDs

For example, you may want to assign ACD permissions to restrict access to a sensitive file so that only you and your manager can read it. You may also want to restrict access to a sensitive directory so that only certain members of a group can create files in it.

Use the `ALTSEC` command to change access permissions to a file or hierarchical directory. System managers can assign ACDs on any file or directory in the system. They must supply the lockword for any lockword-protected files before they can assign an ACD, however. Once the file has an ACD, the ACD supersedes the lockword.

You can use the `ADDPAIR` option with the `ALTSEC` command to add ACD pairs to an object that already has an ACD. (You must use the `NEWACD` option to assign ACDs to files having no ACDs.)

For example, to assign a new ACD that gives all users on the system total access to the file `NUMBERS`:

```
:ALTSEC NUMBERS;NEWACD=(R,W,L,A,X,RACD:@.@)
```

The file `SUMMARY` has an ACD (`RACD:@.@`). You want to grant read and write access to users in your account:

```
:ALTSEC SUMMARY;ADDPAIR=(W,R:@.ACCT)
```

Replacing ACDs

You can replace the current ACD by using the `REPACD` option with the `ALTSEC` command.

All users in the `MKTG` account currently have RD and TD access to the directory `van`. The users can only move through `van` and read the names of files in it. Instead, you want to grant all users in `MKTG` greater access to the contents of the directory. You want them to be able to create directory entries, delete directory entries, read directory entries, traverse directory entries, and to be able to read the ACD.

For example,

```
:ALTSEC ./van;REPACD=(CD,DD,RD,TD,RACD:@.MKTG)
```

This option is useful when you want to change the default ACDs assigned to HFS directories and to files outside of MPE groups.

Deleting ACDs

You can only delete optional ACDs on files in MPE groups that can be protected by the file access matrix.

Users in the `ACCT` account have read access to the file `/ACCT/PUB/dir1/summary` and all other users have read ACD access to the file (`R:@.ACCT;RACD:@.@`). If you decide that the users in `ACCT` should no longer have read access to the file, you can delete previously assigned ACD pairs (but you cannot delete the entire ACD):

```
:ALTSEC /ACCT/PUB/dir1/summary;DELPAIR=(@.ACCT)
```

The above example deletes read access to file `summary` for all users in `ACCT` but still allows all users (including those in `ACCT`) RACD access to the file.

You try to specify the following command to delete the ACD pair that matches `@.@`, which is the only ACD pair left on the file:

```
:ALTSEC /ACCT/PUB/dir1/summary;DELPAIR=(@.@)
```

Because this file is located in an HFS directory, it is required to have ACDs and cannot be protected by the file access matrix. You receive an error message and the ACD will not be deleted:

```
Cannot delete ACDs from objects where file matrix security
does not apply. (CIERR 7330)
```

If the file `REPORT` is a file in an MPE group, its GID matches the GID of its parent group, and its ACD is not required, you can use the following command to delete all ACD pairs:

```
:ALTSEC REPORT;DELACD
```

Copying ACDs

You can copy ACD pairs from one file to another or from one directory to another. This is particularly useful if you assign a complex set of ACDs to one file or directory and you want to assign the same set to another file or directory.

Note

You can only copy an ACD from one file to another or from one directory to another. You can't copy an ACD from a directory to a file or vice versa.

For example, you can copy the ACD from directory `dir1` to another directory `dir2`:

```
:ALTSEC ./dir2;COPYACD=./dir1
```

You can also copy ACDs between devices. The following example copies the ACD associated with ldev 5 to all devices in the device class `TERM`:

```
:ALTSEC TERM,DEVCLASS;COPYACD=5,LDEV
```

Glossary

absolute pathname
A pathname that begins with the root directory, such as `/SYS/PUB/TDP`. See also *pathname* and *relative pathname*.

access control definition (ACD)
Security feature that controls access to files and directories. Consists of a list of access permissions and user specifications. (For example, `R,W,X:@.PAYROLL` gives all users in the `PAYROLL` account read, write, and execute access to the file or directory that is assigned this ACD.) ACDs are applied to files or directories by using the `ALTSEC` command. By default, all files existing outside the traditional MPE account/group structure and all directories are assigned ACDs when they are created.

access mode
A type of access permitted to a file, such as write, read, or execute access.

appropriate privilege
Having sufficient capabilities to perform an operation on MPE/iX. SM capability always provides appropriate privilege to system administrators.

byte-stream file
A type of file that does not have a system-defined record structure. Information in a byte-stream file is stored as a stream of bytes.

case sensitivity

HFS file names can be saved in uppercase or lowercase letters. The file named `./FILE1` does not refer to the same file as `./file1` or `./File1`.

current working directory

The directory in which you are working and from which relative pathnames are resolved. See also *directory* and *relative pathname*.

directory

A special kind of file that contains entries that point to other files. It acts like a container for files and other directories. On MPE/iX, root, accounts, and groups are special types of directories.

dot (.)

The identity function of a directory. The dot `(.)` entry of any directory refers to the directory itself.

dot-dot (..)

Convention that signifies the parent directory in HFS syntax. See also *current working directory* and *HFS syntax.*

file

An object that can be written to, read from, or both. A file has certain attributes including access permissions and file type.

file owner

The person whose user identity matches that of the file. The `$OWNER` ACD entry can restrict the file access of the file owner. The file owner is similar to the file creator. The command b`LISTFILE filename -3` displays the fully qualified user ID (`user.account`) of the file owner.

file name

A name of a file that can be in MPE syntax (`FILE.GRP.ACCT`) or HFS syntax (`/ACCT/GRP/FILE1`). Each syntax has different restrictions on file name length and the characters that can compose the name. See also *MPE syntax* and *HFS syntax*.

group

For POSIX compatibility, refers to a group of related users. This is distinct from MPE groups, which are special types of directories existing directly below accounts.

group ID database

A system database that contains the group name, group ID, and user names for all groups.

group ID (GID)

A number that determines group access privileges. (On MPE/iX, the string representation of the GID is an account name, e.g. it is equivalent to the string `@.account`).

HFS syntax

Expanded MPE/iX syntax that is case sensitive and allows users to address multiple levels in the hierarchical file system. An MPE-escaped syntax name beginning with `.` or `/` automatically signifies HFS syntax to MPE/iX.

Some additional rules are as follows:

- Names of directories directly under root, a group, or an account may have up to 16 characters.
- Names of directories or files not directly under the root, group, or an account can be up to 255 characters.
- `/` separated path to file
- 1024 total character limit

- Names of directories and files can contain the following special characters: hyphen (-), dot (.), or slash (/), but may not have a leading hyphen (-).

hierarchical file system (HFS)
A file system that is tree structured and can contain files at many different levels. This file organization is obtained through the use of directories, which can contain files and other directories.

MPE/iX Developer's Kit
An MPE/iX product that supports programmers who want to port POSIX-compliant applications to MPE/iX. It emulates a UNIX-like environment by providing the MPE/iX shell and utilities, HP C compiler, HP C library functions, and POSIX.1 library functions.

MPE/iX shell
A command interpreter that provides a set of commands and utilities in a UNIX-like environment on MPE/iX. The MPE/iX shell is a POSIX.2-compliant shell that is similar to the UNIX Korn shell. It is invoked from the MPE CI.

MPE syntax
Rules that determine the file name length, special characters, and conditions for files, groups, and accounts. Account, group, and file names can be up to 8 characters. Characters are always converted to uppercase. Characters must be alphanumeric. This is the syntax current MPE/iX users are used to using (for example, `LEDGER.PUB.SYS`).

parent directory
The directory which contains the current object.

pathname

A way of identifying the path to any MPE/iX file. For example, you can refer to `FILE1.PUB.SYS` using the pathname `/SYS/PUB/FILE1`. Notice that pathnames are top- down rather than bottom-up as MPE syntax.

POSIX

Portable Operating System Interface. A set of standards that address various areas of operating system technology. The POSIX standards describe functions of an operating system interface that applications use to become "POSIX-compliant." The main point of POSIX is to facilitate software portability and minimize porting costs.

process

A program currently being executed.

PXUTIL utility

A utility designed to support the POSIX-required user and group databases. It allows you to backdate all user information while backdating a system.

relative pathname

A pathname that is interpreted from the current working directory. For example, `./dir1/longfilename` refers to the file `longfilename` in directory `dir1` in the current working directory.

root directory

Also called (and designated by) a slash or `/`. It is a system directory; all files, accounts, groups, and directories connect back to the root directory. All accounts on MPE/iX are direct descendants of the root directory.

shell
A command interpreter similar to the MPE CI. See also *MPE/iX shell*.

signal
The notification of an event occurring on the system.

slash (/)
Another name for the root directory. See also *root directory*.

subdirectory
A directory that is contained within another directory is sometimes referred to as a subdirectory.

system-provided UDCs
Several UDCs have been added to the system to simplify using the new features of MPE/iX. The UDCs are `DISCUSE`,`FINDDIR`, `FINDFILE`, `LISTDIR`, `PLISTF`, `LISTFTEMP` and `SH`. They are located in the file `HPPXUDC.PUB.SYS`. The system manager must activate these UDCs in order for the UDCs to be available to users.

user ID database
A system database that contains the user name, user ID, and group ID for each user on the system.

user ID (UID)
A user ID determines owner access privileges to files and directories. Its string representation is in the form of `user.account`.

Index