INTERACTIVE UNIX* System V/386 Release 3.2 Guide for New Users

INTERACTIVE product family

INTERACTIVE
A Kodak Company

First printing (March 1990)

INTERACTIVE Systems Corporation
2401 Colorado Avenue
Santa Monica, California 90404

INTERACTIVE UNIX System V/386
Release 3.2
Guide for New Users

CONTENTS

INTERACTIVE UNIX
Operating System Primer

CONTENTS

INTERACTIVE UNIX*
Operating System Primer

INTRODUCTION

What Will I Learn From This Primer?

This primer is an introduction to the basic components and features of the INTERACTIVE UNIX System V/386 Release 3.2 Operating System, Version 2.2. It was written to provide you with the information you need to start using the system. It is not intended as a technical reference manual. If you require detailed documentation, refer to the "Documentation Roadmap" in the *INTERACTIVE UNIX System V/386 Release 3.2 Operating System Guide*.

In this primer you will learn how to:

- Execute and stop INTERACTIVE UNIX Operating System commands.
- Interpret INTERACTIVE UNIX System error messages.
- Create directories to organize documents such as memoranda, letters, reports, and tables.
- Use the INTERACTIVE UNIX System command interpreter, the shell.

To reinforce what you read in this primer, try each example on your system.

New terms introduced in italics can be found in the glossary at the end of this primer.

Documentation References

Throughout this guide, the following full documentation titles will be referenced in shortened versions as follows:

Full Title	*Shortened Version*
INTERACTIVE UNIX System V/386 Release 3.2 Operating System Guide	INTERACTIVE UNIX Operating System Guide
INTERACTIVE UNIX System V/386 Release 3.2 User's/System Administrator's Reference Manual	INTERACTIVE UNIX System User's/System Administrator's Reference Manual
INTERACTIVE UNIX System V/386 Release 3.2 User's Guide	User's Guide
INTERACTIVE Software Development System Guide and Programmer's Reference Manual	INTERACTIVE SDS Guide and Programmer's Reference Manual

GETTING STARTED

Before you can begin this section, you or your system administrator must install your system and set up your *login account*. If the power to your computer is off, you must *boot* (bring up) the system before you can *log in*.

Booting the System

Before you can log in to your computer, you must boot the *operating system*. The operating system is the group of programs and utilities that manages the computer's resources, processes user requests, and runs application programs. You must boot the system at the *console*, or main, terminal, which is the terminal directly attached to the computer unit. Use this procedure:

1. Make sure that there is no diskette in the disk drive.
2. If you have turned off your computer, turn it on again. The INTERACTIVE UNIX Operating System is automatically booted from the fixed disk.
3. If the power to your computer is turned on and your system has been previously shut down (refer to the section "Shutting Down the System" in this primer), press any key to reboot your system.
4. `Booting the UNIX System...` appears on your screen and the computer begins to boot. The screen then clears and a number of messages appear. The system is ready for you to log in when you see the message `Console Login:`.

Logging In

To use the INTERACTIVE UNIX System, you must turn on your computer or terminal and log in. If you are logging in to the console terminal, your screen will look like this:

```
Console Login:
```

If you are using a terminal other than the console terminal, your screen will look like this:

```
login:
```

Type the user identification name (ID) assigned to you (this is also called your login), then press ENTER. Your user ID can contain a maximum of eight characters and can consist of uppercase letters, lowercase letters, and numerals. By convention, most user IDs on UNIX Systems consist of lowercase letters.

After a user ID is entered, the system may request a password:

```
login: tony
Password:
```

If you have been assigned a password by your system administrator, when `Password:` displays, type your password and press ENTER.

☛ The system does not display the password you type on the screen.

If you have not been assigned a password, your screen will look similar to this:

```
login: tony

*** Welcome to the UNIX Operating System. ***

$
```

If you accidentally log in using uppercase letters, the system assumes that your terminal (console included) is not capable of handling lowercase letters. All input and output for the remainder of the session will be shown in uppercase letters. If this happens, log off, make sure your CAPS-LOCK key is not engaged, and log in again.

When you have completed the login procedure, the system displays a *prompt* on the screen. The prompt is a symbol, usually a dollar sign (`$`) or a percent sign (`%`), but because this character can be changed by your system administrator, your prompt may be different. The prompt indicates that the system is ready to receive information. You may enter a command or run an application when the prompt is displayed on your screen.

When you interact with the operating system, the commands you type are processed by a *command interpreter*, which passes your commands to the operating system for processing and delivers the results to you. The INTERACTIVE UNIX System command interpreter is called the *shell*, and the prompt that is displayed is usually

called the *shell prompt*. You will learn more about the shell in the section "USING THE SHELL" in this primer.

Your Home Directory

Each time you log in, the system places you in your *home directory*. Your home directory serves as your personal work area.

A *directory file* (such as your home *directory*) can contain both files and other directories. It is similar in function to a file cabinet: a file cabinet has several drawers (directories), each of which can contain many folders (subdirectories), each holding one or more documents (files). A *file* is a collection of data stored under an assigned name. For example, a letter, a sales report, or payroll data can all be stored in files. You will learn more about directories and files in the section "INTERACTIVE UNIX SYSTEM DIRECTORY STRUCTURE" in this primer.

Setting a Password (`passwd`)

If you were not assigned a password when your login account was set up, the system will probably request that you choose one the first time you log in. A password keeps unauthorized users from tampering with your files by assuring that no one can log in to your account on the computer. Once your password is set, only you, or someone who knows your password, can access your account.

COMMAND NAME	`passwd`
FORMAT	`passwd`
DESCRIPTION	Set or change your login password. The program prompts for the old password (if any) and prompts twice for the new password.
OPTIONS	None.
ARGUMENTS	None.

Log in to the system. The system prompts you immediately for a password:

```
login: tony
$ New password:
```

Enter a word between six and eight characters long that will be easy for you to remember, then press ENTER. The password must contain at least one numeric or special character. Special characters include the space character and:

< > * ? | & $; \ " ' ` ^ ( ) [ ]

Note that when you type in the word you have chosen, the system does not display it on the screen. The system verifies the password by asking you to type it again:

```
New password:
Re-enter new password:
```

If you make a mistake entering your password the second time, the system will give you another chance.

Some system administrators assign all new users a default password when their login accounts are established. You should change this as soon as you log in so that others who may know the default password do not have access to your account. Use the `passwd` command to change your current password. Type the `passwd` command at the shell prompt, then press ENTER:

```
$ passwd
```

☛ Remember to type the command exactly as it is shown in the example. You may not type passw*o*rd or any other variation of the command.

The system asks you to type in your old password, prompts you for a new one, and then has you verify the new one by typing it a second time:

```
$ passwd
passwd: Changing password for tony
Old password:
New password:
Re-enter new password:
```

☛ Don't forget your password! You need it to log in to the system.

Logging Out

To end your access to the system, you must *log out*. Always log out when you have finished using the computer, to prevent unauthorized use of your account.

On most systems there are two ways you can log out. You can hold down the CTRL key and simultaneously press d, or you may type the `exit` command at the prompt:

```
$ exit
```

It is not necessary to press ENTER after logging out. The computer will display the `login:` prompt, indicating that it is ready to accept a login name for the next session.

Shutting Down the System

The INTERACTIVE UNIX Operating System is a *multi-tasking* system. A multi-tasking system means that the computer can be running many different processes (programs) at the same time. For example, you may be editing a file at the same time another file is being printed on your printer. When you are ready to turn off your computer, you must arrange for the computer to complete all of the tasks that are currently running. This is accomplished by logging in with a special administrative login account called `powerdown`, which runs a system maintenance procedure called `shutdown`. The `shutdown` program will gracefully terminate the tasks that are currently executing before halting the system. When `shutdown` has finished, you can safely turn off the computer.

☛ If you do not run the `shutdown` program, you may lose data that is stored on your system and cause damage to your *file system*.

When you are ready to turn your machine off, you may bring the system down with a special administrative login account called `powerdown`, which safely runs the `shutdown` procedure. You must be authorized to use this login account. Follow this procedure:

1. Log out of your ordinary user account.
2. Log in to the system with the `powerdown` user ID.

   ☛ Note that you must know the `powerdown` password if one has been set (see your system administrator).

3. Once you have completed the login procedure using the `powerdown` login, the system automatically executes the `shutdown` program. The system displays a screen similar to this:

```
login:  powerdown
Password:
UNIX System V Release 3.2 80386
Copyright (c) 1989 AT&T
All rights reserved
Once started, a powerdown CANNOT BE STOPPED.
Do you want to start an express powerdown [y, n, ?, q]
```

4. If you are ready to bring the system down, type `y`. The system responds:

```
Shutdown started  Wed Jun  3 17:31:44 PDT 1987

Broadcast message from root (console) on plato
Wed Jun  3 17:31:44 PDT 1987
THIS SYSTEM IS BEING SHUTDOWN !!!
Log off now or risk your files being damaged.

INIT: New run level 0
The system is coming down, please wait.
System services are now being stopped.
Stopping process accounting.

The system is down
Press any key to re-boot.
```

5. When the `Press any key to re-boot` message appears, the computer may be turned OFF.

USING UNIX SYSTEM COMMANDS

On INTERACTIVE UNIX Operating Systems the words *command* and *program* are nearly synonymous. In simple terms, the user types a command, followed by ENTER, and the INTERACTIVE UNIX System runs the program that performs the user's command.

An INTERACTIVE UNIX System *command line* can have three parts: the *command* name, its *options*, and its *arguments*. Each command line is entered in to the computer by pressing ENTER. When the ENTER is received, the shell sends the command to the operating system for execution.

To stop the execution of a command, press DEL.

Command Syntax

The INTERACTIVE UNIX System is *case sensitive*, which means that the system always distinguishes between uppercase and lowercase letters. Most INTERACTIVE UNIX System commands, options, and arguments are typed in lowercase letters. Options typically begin with a dash (–). Each command, option, or argument consists of one *word*, which is interpreted as a group, or *string*, of characters surrounded by spaces.

If you make an error when typing a command, use the BACKSPACE key to correct the error. You may *not* use the *cursor positioning* (arrow →) *keys*.

Always type the command name first, followed by a space; the desired option or options, each followed by a space; then any arguments, separated by spaces.

Here is the command format:

COMMAND NAME	`command name`
FORMAT	`command` [option(s)] *argument(s)*
DESCRIPTION	A brief description of what the command does.
OPTIONS	A list of the most useful options and a brief description of each.
ARGUMENTS	Mandatory or optional arguments.

If an argument is not required, it is shown in square brackets [ ]. Options are always "optional," so they are always shown in square brackets [ ]. Only the most common options and arguments are discussed in this primer. If there are additional options or arguments available for a particular command that are not presented in this primer, this is indicated by the phrase "Not presented in this document." For a complete listing of the available options and arguments for a command, refer to the reference manuals for your system.

Command Names

INTERACTIVE UNIX System command names are short or abbreviated words that describe the programs they invoke. They are deliberately kept short to save time and reduce typing errors. The `passwd` command is an example of an abbreviated word used as a command name. When you enter an INTERACTIVE UNIX System command name, you *cannot* abbreviate it or change its spelling in any way.

For the novice user, INTERACTIVE UNIX System command names can be cryptic and confusing. Because the UNIX System was originally developed for systems programmers, many of the commands were not created with the new user in mind. However, you need only learn a few commands to run application programs and other facilities on your system. If your system includes a user-friendly interface such as the TEN/PLUS* interface, you will not have to learn many UNIX System commands at all.

Using a Simple Command

You have already used the `passwd` command to set or change your system password. `passwd` is an example of a simple command that requires only the command name to execute properly.

The `ls` command can also be executed using only the command name. `ls` is used to "list" the names of the files and directories in the current directory. (You will learn more about files and directories in the section "INTERACTIVE UNIX SYSTEM DIRECTORY STRUCTURE.")

COMMAND NAME	`ls`
FORMAT	`ls` [*file name(s)*] `ls` [*directory name(s)*]
DESCRIPTION	For each directory named, alphabetically list its contents. For each file named, list the requested information.
OPTIONS	Given in a later section.
ARGUMENTS	A file name or a directory name. If no file or directory is named, list the current directory.

If you type `ls` at the shell prompt, the system will display an alphabetical listing of the file and directory names in the *current working directory*. For example:

```
$ ls
 bin
 bag.acct
 gen.memo
 letters
 memos
 midwest
 notes
 north
 south
```

If your current directory is empty, nothing will happen except that the system displays the shell prompt again:

```
$ ls
$
```

Command Arguments

An argument gives the system information that is required to process a specific command or to change the *default*, or standard, behavior of a command. Some commands require one or more arguments; for others, arguments are optional. An argument usually consists of a file, directory, or user name. Refer to the reference manuals for your system to determine the requirements of each command.

`ls` is an example of a command that accepts a file or directory name as an optional argument. If you type `ls` followed by a file name, the system redisplays the file name if the file exists in the current directory:

```
$ ls notes
notes
```

If a directory name is used as the argument, the system displays the names of all the files and directories in that directory:

```
$ ls letters
fox.ltr
lit.ltr
```

Command Options

An option (sometimes called a *flag*) is a special kind of argument that is specific to a particular command. As its name implies, an option is not required, but it provides additional versatility when used with a command. Most options are preceded by a dash (-). Some UNIX System commands can accept both arguments and options.

The `ls` command has several options. The `-l` or "long form" option is illustrated here. If you type `ls -l`, the system lists not only the names of the files and directories in the current directory, but also the owner, file access protections, size in bytes, and time of last modification for each file and directory (some of these file attributes are discussed in the section "ADDITIONAL INTERACTIVE UNIX SYSTEM COMMANDS"). A typical directory listing looks like this:

```
$ ls -l
drwxr-xr-x  3 tony staff   69979 Apr  2  14:41 bin
-r--r--r--  1 bobr staff   30873 Apr 16  09:02 bag.acct
-rwxrw-r--  1 tony staff       2 Apr  7  13:24 gen.memo
drwx------  2 tony staff    6107 Apr  2  10:48 letter
-rw-r--r--  1 tony staff   52709 Apr  8  16:41  memos
-rw-------  1 tony staff    1338 Apr 16  13:04 midwest
-rw-rw-rw-  1 tony staff   34395 Apr 20  11:43 notes
-rw-------  1 tony staff   44978 Apr 17  16:32 north
-rw-------  1 tony staff     922 Apr 17  11:21 south
```

You will learn more about the output of this command in later sections.

Who Is on the System (`who`)?

After you have logged in, you can use the `who` command to see a list of all users who are currently logged in.

COMMAND NAME	`who`
FORMAT	`who [ am i ]`
DESCRIPTION	List the users logged into the system.
OPTIONS	Not presented in this document.
ARGUMENTS	List the information for the user only.

The system displays the name, the terminal ID, the date, and time of login for each user logged into the system:

```
$ who
barbara tty4      Mar 15 08:44
michael console   Mar 15 07:23
terryr  tty10     Mar 15 09:02
```

Use `who` with the arguments `am i` to determine the login information for the terminal you are using:

```
$ who am i
tony   console  Mar 15 07:23
$
```

What Day Is It (`date`)?

You can use the `date` command to find out the date and time.

COMMAND NAME	`date`
FORMAT	`date`
DESCRIPTION	List the current date and time.
OPTIONS	Not presented in this document.
ARGUMENTS	Not presented in this document.

The `date` command displays output similar to this:

```
$ date
Wed Mar 15  11:53:14 EST 1987
```

What's in This File (`cat`)?

The `cat` command is used to display the contents of a file on the terminal screen.

COMMAND NAME	`cat`
FORMAT	`cat [ –s ] [` *file name(s)* `]`
DESCRIPTION	Print one or more files on your terminal screen. If more than one file name is supplied, it is "con*cat*enated" to (added onto the bottom of) the previous file and printed.
OPTIONS	`-s` Be silent – do not comment about files that do not exist.
ARGUMENTS	The file name(s) in the order that you want them to print.

For example, to view the contents of the file `/etc/password` (the system file that stores information about each user ID on the system), type:

```
$ cat /etc/passwd
```

The system displays the contents of the file on your screen. If the file is a long one, the system will scroll the contents on your screen. To initiate a pause in the scrolling, hold down the CTRL key and simultaneously press s. To resume scrolling, hold down the CTRL key and simultaneously press q.

To display the contents of two files, use the `cat` command followed by the two file names, which are separated by spaces:

```
$ cat /etc/passwd /etc/group
```

In the previous example, the file `/etc/passwd` is displayed, followed by the file `/etc/group`.

INTERACTIVE UNIX SYSTEM MANUAL ENTRIES

One type of documentation used for virtually all UNIX-based Systems is the reference manual, which consists of *manual entries*. Manual entries follow a specific format that may be confusing without some explanation.

INTERACTIVE UNIX System entries and the standard AT&T manual entries are grouped alphabetically within the following sections:

- Section 1 – system commands
- Section 1M – system maintenance commands
- Section 2 – system calls
- Section 3 – subroutines
- Section 4 – file formats
- Section 5 – miscellaneous facilities
- Section 7 – special files
- Section 8 – miscellaneous

Section 6 is no longer used. For the INTERACTIVE UNIX Operating System, manual entries for users and system administrators are located in the *INTERACTIVE UNIX System User's/System Administrator's Reference Manual*. (All of the commands discussed in this document can be found in this reference manual.) Manual entries for programmers are found in the *INTERACTIVE SDS Guide and Programmer's Reference Manual* that accompanies the optional INTERACTIVE Software Development System extension. Some of the optional software packages available to run on the INTERACTIVE UNIX Operating System also contain manual entries as part of their documentation.

Manual entries located in the reference manuals may also be found on-line if your system administrator has installed them on your system. If installed, type `man` and the entry name to see the text of that entry on your screen.

Reading a Manual Entry

Manual entries all follow a similar format. Only the Section 1 entries (commands) are discussed here, since as a new user you are most likely to need the information there. Entries in other sections follow the same general rules.

Every entry contains certain headings. An entry *always* has a name line, a synopsis, and a description. Other sections are added as needed.

Here is a typical manual entry for a simple user command, `cat`.

cat(1) cat(1)

NAME

cat – concatenate and print files

SYNOPSIS

cat [–u] [–s] [–v] [–t] [–e] file ...

DESCRIPTION

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

"cat file"

prints the file, and

cat file1 file2 >file3

concatenates **file1** and **file2** and writes the result in **file3 .**

If no input file is given, or if the argument – is encountered, *cat* reads from the standard input file. The following options apply to *cat :*

–u The output is not buffered. (The default is buffered output.)

–s *cat* is silent about nonexistent files.

–v Causes non-printing charcters (with the exception fo tabs, new-lines, and form-feeds) to be printed visibly. ASCII control characters (octal 000 - 037) are printed as ^*n*, where *n* is the corresponding ASCII character in the range octal 100 - 137 (@, A, B, C, . . ., X, Y, Z, [, \,], ^, and _); the DEL character (octal 0177) is printed ^?. Other non-printable characters are printed as M-*x*, where *x* is the ASCII character specified by the low-order seven bits.

The following options may be used with the **–v** option:

–t Causes tabs to be printed as ^I's and form-feeds to be printed as ^L's.

–e Causes a $ character to be printed at the end of each line (prior to the new-line).

The **–t** and **–e** options are ignored if the **–v** option is not specified.

WARNING

Redirecting the output of *cat* onto one of the files being read will cause the loss of the data originally in the file being read. For example, typing:

cat file1 file2 >file1

will cause the original data in **file1** to be lost.

SEE ALSO

cp(1), pg(1), pr(1).

Figure 1. The *cat*(1) Manual Entry.

The Name Line. The name line contains the name of the command and a brief description of its function.

The Synopsis Line. The synopsis is a line (or set of lines, if the command is more complicated) that summarizes how the command is stated at the system prompt with its options and arguments. The command is shown in bold face; options are bold and are always shown inside square brackets [ ]. *file* is the argument needed for the command (a file name) and the three dots "..." mean that you can give more than one file name at a time as arguments.

The Description Section. The description is just that – a description of how the command and its options function. For example, in *cat*(1) above, the description begins, "*cat* reads each *file* in sequence and writes it on the standard output." (The standard output device is your terminal screen. This is discussed in the section titled "USING THE SHELL.") It then shows you how to "cat" more than one file at a time and direct the results into a third file. The rest of the description gives details about using various options.

The See Also Section. This section contains cross references to other manual entries or documentation that contain information of related interest.

Miscellaneous Sections. Other sections found in various manual entries include the *Warnings* or *Caveats* section, which alerts you to possible problems when using the command in certain ways; the *Example* section, which contains an example using the more complicated commands; the *Files* section, which contains the system files in which the command is located; the *Diagnostics* section, which contains error messages of interest to programmers who may be using the command in a program; the *Bugs* section, which alerts you to known problems with the command; and the *Notes* section, which contains helpful hints or information that does not really fit anywhere else.

INTERPRETING INTERACTIVE UNIX SYSTEM ERROR MESSAGES

If something goes wrong when you use an INTERACTIVE UNIX System command, the system usually generates an *error message* to help you determine why the command did not work. Error messages are typically generated when you misspell a command or file name, when a command cannot be automatically accessed from your account, or when a file or directory name does not exist. Error messages on the INTERACTIVE UNIX System can be terse and cryptic. However, as you gain experience with the system, you will become more adept at deciphering them.

If you fail to provide the correct spelling for a command or file name, an error message is displayed on your screen. For example, if you type `$ datte` instead of `date`, the system displays this error message:

```
$ datte
datte: not found
```

The INTERACTIVE UNIX System also generates an error message if you fail to use space to separate the command name from its arguments. For example, if you type `$ ls-l` instead of `$ ls -l`, the system generates the same message:

```
$ ls-l
ls-l: not found
```

Some common error messages are:

command name: `not found`
: You may have misspelled the command or forgotten a space that was needed. Check your spelling and try typing the command again. If the problem persists, then either your search path does not include the directory where the command is stored or the command is not available on your system. Check with your system administrator.

directory name: `bad directory`
: You may have misspelled a directory name, but this message also appears if the directory does not exist. Check your spelling or `cd` to the parent of the directory you are trying to reach and verify that the directory exists and that you have the correct spelling of its name. If you get this message while trying to `cd` into a

directory, it may be that you do not have execute permission for that directory. Check the permissions for the directory by using the `ls -l` command while in the parent directory.

`Invalid argument`

You have used an argument that the command does not accept. Try typing the name of the command without an argument. The system will often display a message that describes how to use the command. You should also check your command reference manual for the correct arguments.

`No such file or directory`

You have specified the name of a file or directory that does not exist. Often this means that one of the names in your path name is misspelled. Check your spelling and try again. If the problem persists, `cd` to the parent of the directory you are trying to reach and verify that the directory exists and that you have the correct spelling of its name.

`Permission denied`

You do not have permission to write in the named directory or to perform some operation on the named file. Check the permissions of the target with `ls -l`. Change them or have the owner of the file or directory change them, as necessary.

```
Usage: {mv|cp|ln} f1 f2
       {mv|cp|ln} f1 ...fn d1
       mv d1 d2
```

You have failed to specify a mandatory argument or have specified arguments in the wrong order while using `cp`, `mv`, or `ln`. This type of message shows the correct usage.

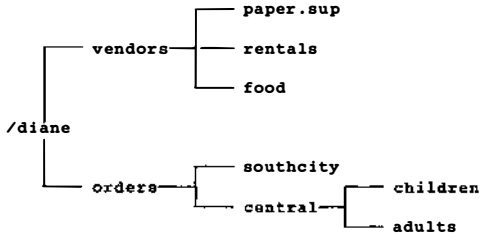
INTERACTIVE UNIX SYSTEM DIRECTORY STRUCTURE

On an INTERACTIVE UNIX Operating System, information is stored in an organized structure, called a *hierarchical file system*. It is called hierarchical because of its multi-level structure. A single master directory is at the top level, and additional files and directories are defined at various levels below it.

A file system is a collection of individual files and directories that are stored on a portion of a disk.

The master directory for the entire system is called the *root directory*. It is the first directory in the file system structure and is named slash (/). The `root` directory contains several directories, and each of these directories can contain other directories. Because the INTERACTIVE UNIX System file system is sometimes visualized as a "tree," each new directory created on the system can be viewed as a new branch added to the directory tree.

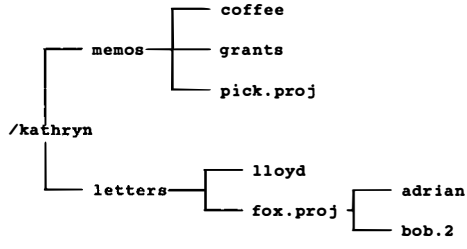
Here is a diagram of a simple directory structure for a computer user named Diane:



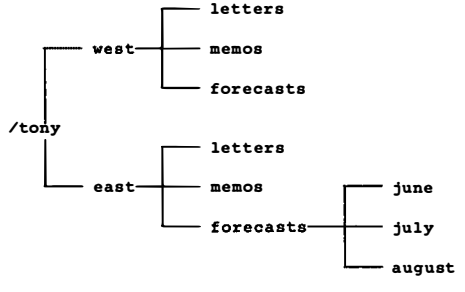
The `root` directory, (/), contains a directory named `diane`. `/diane` contains two directories: `vendors` and `orders`. These in turn, hold other files and directories. When Diane logs in, she is automatically placed in her home directory, `/diane`.

Because the INTERACTIVE UNIX System file system is very flexible, the organization of your files and directories can be changed at any time to fit your needs. For example, you might have a subdirectory in your home directory called `memos`, which contains all the memos you have written. Or, you could have a `letters`

directory, which could contain a number of other directories, one for each of your major projects:



If you deal with different regional offices, you might have a major subdirectory for each region, with each subdirectory containing parallel directories for sales figures, letters, memos, and forecasts:



The only limit on the number of files and directories you can have, or the number of directory levels you can have, is the amount of disk space available to you.

File and Directory Naming Conventions

All programs, text, and data on disks or diskettes reside in files. Each file or directory has a unique name. The INTERACTIVE UNIX System is case sensitive, which means that it distinguishes between upper- and lowercase letters. INTERACTIVE UNIX System file and directory names can be one to fourteen characters long and can include both uppercase and lowercase letters. Although

some symbols may be used in INTERACTIVE UNIX System file and directory names, it is safest to limit names to alphanumeric characters (`a - z` and `0 - 9`) and the period (`.`).

Some symbols have special meanings to the INTERACTIVE UNIX Operating System. Never use an asterisk (`*`), a question mark (`?`), a greater than symbol (`>`), a lesser than symbol (`<`), an ampersand (`&`), or a vertical bar (`|`) in a file or directory name. You will learn more about how these symbols are interpreted by the UNIX System shell in later sections.

Path Names

When you want to access a directory or file in the file system, you use a *path name*. A path name is a sequence of directory and file names, separated by slashes, that designates the location of a particular file or directory. Since the *full path name* is actually the complete name of the file or directory, it ensures that no two directories (or no two files) on the system have exactly the same full name. To use the filing cabinet analogy again, you could locate a letter addressed to John Smith using its "path name," by saying it is in the second cabinet drawer, behind the divider marked "Boni Project," in a folder marked "letters."

There are two kinds of path names, a full (or *absolute*) path name and a *relative path name*. A full path name *always* begins with a slash (`/`) and consists of the sequence of directories from the `root` directory to the file or directory you wish to access. A relative path name does *not* begin with a slash. It describes the location of the target file or directory relative to your current location.

A full path name consists of a slash followed by one or more directory names separated by slashes and ending with the name of the target file or directory. The slash at the beginning of the path name tells the system to begin its search for the target from the `root` directory. For example, the full path name for the file that contains user Tony's eastern region August forecast might be:

```
/tony/east/forecasts/august
```

A relative path name omits all directory names up to and including the current directory from the path name. It tells the system to begin the search for the target file or directory from the current directory. You can access a file in your current working directory by using its relative path name. In this case, the relative path name

consists of only the file name. Or, you can use the relative path name to access a file in a subdirectory of your current working directory by specifying the name of the subdirectory followed by a slash and the name of the file. For example, user Diane can display the contents of the `southcity` file in the `orders` subdirectory of her current working directory by using the relative path name with the `cat` command:

```
cat orders/southcity
```

It is not necessary to use the full path name to the file, `/diane/orders/southcity`.

Where Am I (`pwd`)?

If you work in a number of directories on many different projects, you may occasionally forget where you are working. The INTERACTIVE UNIX System command `pwd` (**p**rint **w**orking **d**irectory) will tell you where you are in the directory structure.

COMMAND NAME	`pwd`
FORMAT	`pwd`
DESCRIPTION	Print the current working directory. Prints the complete path name of the directory you are currently in.
OPTIONS	None.
ARGUMENTS	None.

Use the `pwd` command to see the full path name of your current directory. If you have just logged in, the system should respond with the name of your home directory:

```
$ pwd
/tony
```

Creating a Directory (`mkdir`)

It is a good idea to organize your files by filing them in subdirectories. Use the `mkdir` command to make new directories.

COMMAND NAME	`mkdir`
FORMAT	`mkdir` *name(s)*
DESCRIPTION	Make a directory with the name given.
OPTIONS	None.
ARGUMENTS	The name(s) of the directory(ies) you want to create.

Use the `mkdir` command to make a new directory called `notes`:

```
$ mkdir notes
```

A `notes` directory is created, but you are *not* automatically moved into it. You remain in the directory from which you issued the command. To verify that the new directory was created, use the `ls` command:

```
$ ls
 notes
```

Remember, follow all commands with ENTER unless otherwise instructed.

Changing Directories (`cd`)

Use the `cd` command to move from one directory to another.

COMMAND NAME	`cd`
FORMAT	`cd` [*pathname*]
DESCRIPTION	Change current directory. If no argument is given, `cd` moves you to your home directory.
OPTIONS	None.
ARGUMENTS	The path name (full or relative) of the directory to which you want to move.

For example, to move from your home directory to a subdirectory called `notes`, type:

```
$ cd notes
```

You could use the full path name, but using the relative path name is much easier. To return to your home directory, type `cd`. Using the `cd` command with no arguments always returns you to your home directory. To use the `cd` command with a full path name, type:

```
$ cd /tony/notes
```

What's in This Directory (`ls`)?

Use the `ls` command to check the contents of a directory. The `ls` command lists all the files and directories in a named directory.

COMMAND NAME	`ls`
FORMAT	`ls [ -ltau ] [` *name(s)* `]`
DESCRIPTION	For each directory named, alphabetically list its contents (no options requested) and any other information requested; for each file named, list the requested information.
OPTIONS	`-l` List in long format the mode, number of links, owner, group, size, and time of last modification. `-t` Sort contents by time of last modification (latest first) rather than by name. `-a` List all entries, including files that begin with a dot (.). `-u` Use time of last access instead of last modification for sorting (with `-t` option) or printing on the screen (with `-l` option).
ARGUMENTS	A file name or a directory name. If no file or directory is named, it gives the current directory.

If your current directory is empty, the `ls` command simply generates a new prompt:

```
$ ls
$
```

The `ls` command accepts a file name, directory name, or path name as an argument. If you use a file name as an argument, `ls` simply lists the name of the file if it exists:

```
$ cd /etc
$ ls profile
profile
```

If you use the `-l` option (as shown in the new command format), `ls` lists more detailed information about the file:

```
$ ls -l profile
-rw-r--r-- 1 bin staff 460 Jun 13 1984 profile
```

Copying a File (`cp`)

COMMAND NAME	`cp`
FORMAT	`cp` *filename pathname*
DESCRIPTION	Copy the named file to the named destination.
OPTIONS	None.
ARGUMENTS	The name of the file you want to copy and the its destination.

The `cp` command is used whenever you need to make a copy of a file. You may want to copy a file if you require several slightly different versions of a letter or if you plan to base a report or presentation on existing material.

To create a new file called `employees` that is an exact duplicate of the file `all.emp` in your current directory, use the `cp` command:

```
$ cp all.emp employees
```

The `cp` command may be used to copy a file in someone else's directory to one of your directories, or to copy a file in your directory to another name or location. You must have the read (`r`) permission for a file to copy it from someone else's area. Refer to the `chmod` command in the section "ADDITIONAL INTERACTIVE UNIX SYSTEM COMMANDS" in this primer for information on file permissions.

To copy a file that resides in another user's directory, you can use a full or relative path name to access the file:

```
$ cp  /susan/letters/sales.ltr  susan.ltr
$ ls
other.ltr
susan.ltr
```

When you copy a file from another directory, you may use the same file name or you may use a new file name. The only restriction is that you do not already have a file with the same name in your current directory. If you do, you will overwrite the contents of your existing file with the contents of the new file. This happens without a warning from the INTERACTIVE UNIX System.

When you use the copy command to copy the contents of one directory to another directory, the directory you are copying to must already exist.

Path Name Shortcuts: Dot and Dot Dot

The INTERACTIVE UNIX System provides many shortcuts to help make life a little easier and to minimize typing. A relative path name is one type of shortcut. Two other shortcuts are dot (.) and dot dot (..).

One "dot," when used in place of a directory name, indicates that the command should use the current directory. This shortcut is useful when you are copying or moving files from another directory to your current directory.

For example, you may use the dot (.) shortcut to copy a file into your current directory and give it the same name:

```
$ cp /susan/letters/sales.ltr .
```

Two "dots," when used in place of a directory name, indicate the directory directly *above* your current working directory. The directory above your current directory is also known as the *parent directory*.

You can use `..` to move quickly to a parent directory. For example, if you are currently located in the directory `/tony/east/letters` and want to return to the directory named `east` (the parent directory of your current directory), simply type:

```
$ cd ..
$ pwd
/tony/east
```

You can also use a series of "dot dots" to move more than one directory at a time. For example, to move to the `root` directory from the directory `/tony/east/letters`, type:

```
$ cd ../../..
$ pwd
/
```

Removing a File or Directory (`rm`, `rmdir`)

The `rm` command allows you to remove a file or directory you have created.

COMMAND NAME	`rm, rmdir`
FORMAT	`rm [ -fri ] [` *file name(s)* `]` `rmdir [` *directory name(s)* `]`
DESCRIPTION	Remove the named file or directory.
OPTIONS	`-f` Ask no questions about removal. `-i` Ask before removing each file or directory. `-r` Examine each directory before removing it.
ARGUMENTS	One or more file or directory names.

To remove a directory, you must first remove all of the files in that directory. Generally, the system will not allow you to remove a directory that contains files.

Use the `ls -a` command to display all files in the `letters` directory, including hidden files (those with file names that begin with a dot):

```
$ ls -a
.
..
.index
lit.ltr
confirm.ltr
```

Use `rm` to remove all the files in the directory except `.` and `..`. These are special file names used only by the system; they cannot be removed. To remove the files:

```
$ rm  .index  confirm.ltr  lit.ltr
```

To remove the `letters` directory, you must `cd` out of it, then use `rmdir` to remove it:

```
$ cd ..
$ rmdir  letters
```

INTRODUCTION TO UNIX SYSTEM EDITORS

An *editor* is a program that is used to enter and modify text and data. There are two basic types of editors: *line editors* and *full-screen editors*. When using a line editor, you must explicitly specify the line or lines you want to edit. You do not see the changes to your file as they occur. A line editor is useful if you are using a hardcopy terminal, i.e, one that does not have a screen. A full-screen editor allows you to see the changes made to a file as you make them. Screen editors are the most popular type of editor today.

Most UNIX Systems have a line editor and one or more full-screen editors. This section outlines three of the most commonly available editors on UNIX Systems; it does not attempt to explain how to use a UNIX System editor. Refer to the "Documentation Roadmap" in the *INTERACTIVE UNIX Operating System Guide* to determine where you can find tutorials that describe the editors available on your system.

The ed Line Editor

`ed` is the standard editor provided on every UNIX System. It is a line-oriented editor and was originally developed to run on terminals without screens. `ed` is not the editor of choice for most users since you cannot see changes and additions to a file in context. It uses two command "modes" (insert and command) and, in traditional UNIX System fashion, the editing commands are terse and cryptic. However, it is a very powerful editor for making global changes to a file, and it offers some advantages when you are editing text over slow telephone lines or on a terminal without a terminal definition for the available screen editor. Sometimes it is the only editor available, so it is useful to know a little about `ed`. Refer to the "Documentation Roadmap" to find references on how to use `ed`.

The vi Screen Editor

`vi` is one of the most widely used UNIX System screen editors. It was originally developed at the University of California at Berkeley, but it is now distributed as part of AT&T's UNIX System V.3. It is also distributed by many other UNIX System vendors. `vi` is an extremely flexible and powerful editor. It provides many tools and features that programmers find especially useful.

`vi` also uses two modes to perform editing tasks: command and insert. The user must toggle back and forth between the two modes to edit text. All commands, including those that move the cursor keys, must be issued while in command mode. Adding text to a file must be done in insert mode. This means that if you notice an error several lines above your current line, you must change to command mode to move the cursor back up to the error and delete the erroneous characters; return to insert mode to type in the correct text; then change once more to command mode to return to the current work. Some users find that switching modes while editing text can be difficult and time consuming, while other users have very little trouble using the editor efficiently.

Refer to the "Documentation Roadmap" to find references on how to use `vi`.

The TEN/PLUS User Interface

TEN/PLUS is a full-screen *user interface* to the UNIX System. It is based on a full-screen editor (the TEN/PLUS editor) developed by INTERACTIVE Systems Corporation to take the mystery out of text editing programs and the UNIX System. It is available on many systems today, including IBM's AIX*, PC/IX, and IX/370, as well as IN/ix* systems available from Wang, SCI, Hitachi, and others.

The TEN/PLUS editor does not rely on modes to perform editing operations; it uses ten basic function keys to perform most editing tasks and is very easy to learn. For a novice user, it is probably much easier to learn TEN/PLUS than the standard UNIX System editors `ed` and `vi`. The TEN/PLUS editor also includes a large number of advanced editing features to make modifying text files easier.

Refer to the "Documentation Roadmap" to find references on how to use the TEN/PLUS editor.

USING THE SHELL

Introduction

The INTERACTIVE UNIX Operating System consists of two major components, the *kernel* and the *shell*. The kernel is the core of the operating system. It is the part of the *system software* that is responsible for managing the computer's *hardware* and *software* resources and for processing commands. The work that the kernel performs is hidden from most users by the shell.

The shell is the part of the system that interacts with the user. It is also known as the command interpreter and is sometimes referred to as the user interface. It is responsible for passing user commands to the kernel for processing and for returning the results to the terminal screen, into a file, or to another device, such as a printer.

There are currently several versions of the shell available on UNIX Systems. The two most popular versions of the shell are known as the *Bourne shell* and the *C shell*. The Bourne shell was written at Bell Laboratories and released by AT&T. The C shell was written and released by the University of California at Berkeley. You should ask your system administrator or refer to your system reference guide to determine which version is available on your system. The Bourne shell is considered to be the standard UNIX System shell, but the C shell has gained widespread popularity, particularly with programmers. The information in this primer generally applies to both versions of the UNIX System shell.

When you have successfully logged into your computer, you are automatically in contact with the shell. As you learned earlier, the prompt you see on the screen is also known as the shell prompt. If you do not see a shell prompt and you are sure that you are logged into the system, your system may also include a friendly user interface. On some systems, the TEN/PLUS User Interface is included. In this case, the shell is still running, but it is hidden from you in order to provide an even easier method of executing commands and running applications.

Shell Features

Although the shell is an ordinary program that is run by the kernel, it is a very versatile program. Not only does it perform the work of the command interpreter, but it also functions as a programming language. The shell can take commands from special files called *shell scripts* or *shell programs*. A shell script can be a simple collection of shell commands such as those normally executed on the command line, or it can be a program that uses programming logic, variable assignment, and argument processing to perform complex tasks. Shell programming is beyond the scope of this primer, but it is covered in more detail in other documentation resources. Refer to the "Documentation Roadmap" for more information.

In addition to being a programming language, the shell has a number of distinctive features that permit even the novice user to perform sophisticated operations with a little practice. They include:

- File name shortcuts known as *wildcards*.
- The ability to *redirect* the input and output of a command to a file, a printer, or another program.
- The ability to use the output of one command as the input to another command, also known as a *pipeline*.
- The ability to run two or more programs simultaneously by using *background* processing.

Each of these features is described in the following sections.

Using Wildcards

Many INTERACTIVE UNIX System commands require one or more file or directory names as arguments to the command. A wildcard is used to match a character or a string of characters in a file or directory name. It is another type of shortcut. (You have already learned two shortcuts for relative directory names, . and ..) A wildcard is also referred to as a *pattern matching character*.

Wildcards are handy because they enable you to use a file name or group of file names without having to type the complete name. It is much easier to move all the files from one directory to another using a wildcard than to type out all the file names. Wildcards can also be used when you only remember part of a file name.

There are three wildcard characters:

?	Matches any single character
*	Matches one or more characters
[. . .]	Matches an array of characters

☛ When you create new files, be very careful not to use a wildcard character as part of the file name. It can be difficult to access or remove a file if there is a wildcard character in its name.

WILDCARD	?
FORMAT	*string*? ?*string* ?*string*? *string*??
DESCRIPTION	Substituting a question mark in a file or directory name argument matches one single character for each ? supplied.

A directory called `misc` might contain the following files:

```
letter1  letter01  letter10    repairs
letter2  letter02  letter.mmb  reporta
letter3  letter03  letter.pc   reportb
```

To remove the files named `letter1`, `letter2`, and `letter3`, type:

```
$ rm letter?
```

All of the other files will remain, either because they do not start with the string `letter`, or because they have more than one character following the string `letter`.

```
$ ls
letter01
letter02
letter03
letter10
letter.mmb
letter.pc
repairs
reporta
reportb
```

To remove the files named `letter01`, `letter02`, and `letter03`, you cannot use the command `rm letter??` because that would also remove the file `letter10`. Instead, type:

```
$ rm letter0?
```

because it removes only the files with one character following the string `letter0`.

```
$ ls
letter10
letter.mmb
letter.pc
repairs
reporta
reportb
```

WILDCARD	*
FORMAT	*string** **string* **string** **string*string*
DESCRIPTION	Substituting an asterisk in a file or directory name argument tells the computer to match a string of characters (zero or more characters).

To list the names of all files beginning with the letters `re` in the `misc` directory, use the wildcard * with the `ls` command. This matches any string of characters beginning with `re`:

```
$ ls re*
repairs
reporta
reportb
```

To list the names of only the files with the character "dot" (`.`) in them:

```
$ ls *.*
letter.mmb
letter.pc
```

Sometimes, it is very useful to use the `*` wildcard in conjunction with the `rm` command to remove all the files in a directory:

```
$ rm *
```

☛ This is a very dangerous command! Be *very* sure that you *really* want to remove all of the files in the directory before you use this command. Your files will be permanently lost!

WILDCARD	`[ . . . ]`
FORMAT	*string*`[`*n*–*n*`]` *string* `[`*n*–*n*`]`*string* `[`*n*,*n*,*n*–`]`*string*
DESCRIPTION	Substituting a list of characters between square brackets in a file or directory name used as an argument to a command tells the computer to match any characters present in the array.

To list all the letter files in the `misc` directory that end in a number, you could use the `[ . . . ]` wildcard:

```
$ ls letter[0-9]
letter1
letter2
letter3
```

Or, to list only those letter files that end in the numbers one and two, type:

```
$ ls letter[1,2]
letter1
letter2
```

Redirection of Input and Output

Every command to the computer requires input and produces output. Normally, INTERACTIVE UNIX System commands take their input from the *standard input* and direct it to the *standard output*. The standard input is usually taken from your terminal keyboard. For example, when you type a command and press ENTER, you are sending standard input to the kernel. The standard output is usually your terminal screen, although some programs write their standard output to a file or a printer. When you type the `ls` command, for example, the standard input is taken from your terminal keyboard and the standard output is written to the terminal screen.

A very important feature of the INTERACTIVE UNIX System is the ability to direct a command to take its input from a source other than its standard input, and to send its output to a destination other than its standard output. For example, you can direct a command to take its input from a file already stored on the computer and to direct its output to another file or to a device such as a printer. The ability to take input from one place and direct it to another at the time you start a process is called *redirection of input/output (I/O)*.

Two symbols are used to indicate I/O redirection. The "greater than" (>) symbol redirects output, and the "less than" symbol (<) redirects input. Since you will probably redirect the output of a command (>) more frequently than you redirect the input (<), input redirection is not discussed in this primer. Refer to the *User's Guide* for more information on input redirection.

I/O redirection is commonly used to store the output of a command in a file, perhaps to be used with another command or to be incorporated into a file or a message. For example, if you need a listing of all the files in your current directory to use in a report, you could use this command:

```
$ ls > files.list
```

The system automatically creates a new file called `files.list` and writes the output of the `ls` command in that file. You will not see any output on your screen. If you want to verify that the redirection worked, you can use the `cat` command to view the contents of `files.list`:

```
$ cat  files.list
letters
memos
```

To create one large file out of two smaller files, use the `cat` command and redirect the output to a new file:.

```
$ cat  /etc/passwd  /etc/group  >  users.groups
```

The two files will be written, one after the other, into the file called `users.groups`. The contents of the files `/etc/passwd` and `/etc/group` are not changed, moved, or harmed in any way by this process. The command simply causes a copy of the file's contents to be placed into the named file.

☛ When redirecting output, you must be careful not use an existing file name as your output destination. You can *overwrite* the contents of the existing file if the file's permissions allow it, thereby losing all the text or data in it. (You will learn more about file permissions in the discussion of the command `chmod` in the section "ADDITIONAL INTERACTIVE UNIX SYSTEM COMMANDS.")

For example, assume Tony owns a file called `widgets` that is located in his home directory. He wants to put a copy of Barb's eastern widgets file into his directory, but he has forgotten that he already has a file named `widgets` in his home directory. If Tony types the command:

```
$ cat  /barb/widgets/east.wid  >  /tony/widgets
```

the contents of his original `widgets` file is deleted and replaced with the contents of Barb's file. The file is still named `widgets`. It is a good idea to check the file names in your destination directory with `ls` before performing an operation like this.

To redirect the output of a command and store the results at the end of a file that already exists *without* overwriting the contents of the file, use two greater than (>>) symbols.

Tony could have *appended* (added to the bottom) the contents of Barb's `east.wid` file onto the end of his `widgets` file with this command:

```
$ cat barb/widgets/east.wid >> tony/widgets
```

Pipes and Filters

The UNIX System is extremely flexible. It was built on the philosophy that utilities on the system should build on the work of other utilities and be reusable. There are two special notions that are important to UNIX Systems: *pipes* and *filters*.

A filter is a program that accepts input from one source, performs the appointed task on the data, and then writes the results to another place without changing the input file in any way. For example, `sort` takes as its input the file you designate, sorts all of the lines in alphabetical order, and displays it on your terminal. The `sort` command is an example of a simple filter.

COMMAND NAME	`sort`
FORMAT	`sort [ -dfn ] [ -o`*output file* `] [` *file name(s)* `]`
DESCRIPTION	Sort the lines of the named file(s) together and write the results to standard output. If no input files are given, it sorts standard input. If no output file is given, it writes to standard output.
OPTIONS	`-d` Use "dictionary" order. Only letters, digits, and blanks are significant. `-f` Ignore case when sorting. `-n` Sort numeric strings arithmetically.
ARGUMENTS	The name of the file you wish to be the output file, preceded by `-o`, and the name(s) of the file(s) you wish to sort.

A pipe is a connection between two or more commands. To create a pipeline, you execute a command and pass its output ("pipe" its output) so that it becomes the input for the next command. A pipe is indicated by a vertical bar (|). Two filters are often used to create a pipeline. The output of a pipeline can be displayed on the screen or it may be redirected to a file.

The `sort` command is a filter that is frequently used in pipelines. For the following example, assume you have two files: `east.emp` and `west.emp`. The first file (`east.emp`) is a list of employees in the Eastern Regional Office, and the second file (`west.emp`) is a list of all the employees in the Western Region. To create a single file that contains a list of all the employees in the Eastern and Western regions sorted alphabetically, use the following command:

```
$ cat  east.emp  west.emp | sort -d > all.emp
```

You have taken the output from `cat`, piped it to `sort`, and redirected `sort`'s output into a file called `all.emp`.

To perform the same task but redirect the output of `sort` to a printer instead, use this command line:

```
$ cat  east.emp  west.emp | sort -d | lp
```

To learn more about the `sort` command, see your INTERACTIVE UNIX System documentation. The `lp` command is discussed in greater detail in "ADDITIONAL INTERACTIVE UNIX SYSTEM COMMANDS."

Background Processing

The INTERACTIVE UNIX System is an *interactive* operating system, which means that you can carry on a dialogue with the computer. Normally, you issue a command and wait for the shell to process your command and return the shell prompt. Sometimes commands can take a long time to complete and do not require any assistance from you. Waiting for the command to finish can be tedious and a waste of time.

To alleviate this situation, the INTERACTIVE UNIX System allows you to request that a command be run in the background. This means that the system starts the process, displays the process identification number (PID) associated with the process, and then returns the shell prompt. The process continues to run until it is completed, while you continue to use the system for other work. To request background processing, type an ampersand (&) at the end of the command line, immediately before pressing ENTER.

Many commands, such as `ls`, complete so quickly that it does not make sense to run them in the background. However, if you are formatting a document or compiling a program, you will find the ability to run a command in the background very useful. Here is an

example of a command line where the user requested that it run in the background:

```
$ cat east.emp west.emp | sort -d > all.emp &
 432
$
```

The shell first gives you an identification number for the process (in this case, `432`) and then returns the INTERACTIVE UNIX System prompt. You can now run another command. When your process in the background is finished, the output will be contained in the specified file.

Note that unlike processes being run in the foreground, a background process cannot be temporarily stopped by using CTRL d or terminated using DEL. It can be stopped only by logging off or using the `kill` command.

ADDITIONAL INTERACTIVE UNIX SYSTEM COMMANDS

There are literally hundreds of commands available on the INTERACTIVE UNIX System. There are commands to format, extract, and print text in a file; compile, link, archive, and store program code; and report available disk space or commands that are currently running on the system. However, a typical user only needs to use about two dozen commands on a regular basis. We have already described some of the most useful commands in the preceding sections. This section will familiarize you with some other useful commands.

Changing Permissions (`chmod`)

Every file and directory on an INTERACTIVE UNIX System is associated with a set of *file access permissions*, or *protection modes*, that specify who will have access to it. There are three different types of *permissions*: "read," "write," and "execute."

Read permission
: Allows a user to view the contents of the file or copy it to another area. Read permission does not allow the user to make changes to the file.

Write permission
: Allows a user to make changes to a file. Write permission on a directory allows a user to create and delete files in that directory.

Execute permission
: Allows a user to execute the file as an INTERACTIVE UNIX System command, if the file contains INTERACTIVE UNIX System command lines. Execute permission on a directory allows a user to `cd` into it. If you attempt to `cd` into a directory for which you do not have execute permission, the system will display the message *directoryname*`: bad directory`.

Each set of permissions applies to one of three categories of users: *owner*, *group*, and *other*. The "owner" of the file owns the file or directory. The "group" often consists of people working on the same project or in the same department and using the same files. Group membership is usually determined by the system administrator. "Other" refers to everyone on the system who is neither the owner nor in the owner's group.

The default permissions associated with each file and directory are:

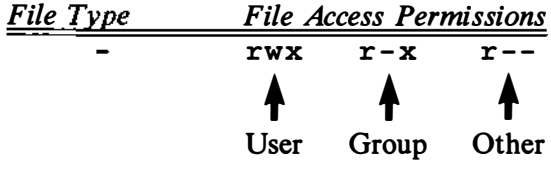
User Category	*Directories*	*Files*
User	read, write, execute	read, write
Group	read, execute	read
Other	read, execute	read

Use the `ls -l` command to look at file and directory permissions:

```
total 272
   -rw-rw-rw-  1 linda  staff     625 Apr  7 13:24 letter01
   -r--r--r--  1 linda  staff   69979 Apr  2 14:41 letter02
   drwxr-x---  1 linda  staff       2 Apr  7 13:24 notes
   -rw-rw-rw-  1 billr  staff   30873 Apr 16 09:02 proj.info
   -rwxr-xr--  1 linda  staff   69979 Apr  2 14:41 save.memo
```

The access permissions appear on the left of the display.

Now look more closely at the permissions for the file `save.memo`:



The first column specifies the file type: a dash (`-`) indicates a file and a `d` indicates a directory. The first column for the `save.memo` file indicates that it is a file. The next three columns specify the read (`r`), write (`w`), and execute (`x`) permissions for the owner. In this case, the owner can read, modify, and execute the file. The next three columns specify the permissions set for the owner's group. The permissions indicate that members of the group can read and execute the file. A dash indicates that a permission is denied; therefore, members of the group cannot modify the file. The last three columns specify permissions for others on the system. Others are allowed to read the file but are denied permission to modify or execute it.

COMMAND NAME	`chmod`
FORMAT	`chmod` *nnn filename(s)* `chmod` *nnn directory name(s)*
DESCRIPTION	Change mode (permissions) of the named file or directory.
OPTIONS	None.
ARGUMENTS	The octal number or a character representation of the the new file access permissions and the name(s) of the file(s) or directory(ies) whose file access permissions are going to be changed.

The `chmod` command takes an argument, which may be in the form of a character representation (`r`, `w`, `x`) or a numeric representation (using *octal* numbers) to indicate the permissions in effect.

Each permission is associated with a number: `read=4`, `write=2`, `execute=1`, and no permission=`0`. The octal equivalents are obtained by adding together the numbers associated with the basic permissions. This table shows the octal equivalents for the basic permissions:

Permission	*Octal representation*
-	0
x	1
w	2
wx	3
r	4
rx	5
rw	6
rwx	7

To change the permissions of the `save.memo` file so that members of the group only are allowed to read the file, you can use the `chmod` command with the octal representation:

```
$ chmod 744 save.memo
```

The numbers `744` are the octal equivalent of `rwx`, `r`, `r`. This gives the owner read, write, and execute permission in the file but allows only read permission to those in the group and others.

To make the same change using the character representation requires a slightly different approach. This table shows the character symbols and their definitions:

Permission	*Definition*
u	Permission applies to user (owner) only
g	Permission applies to group members only
o	Permission applies to others only
a	Permission applies to all (u, g, and o)
+	Add the indicated permissions
-	Remove the indicated permissions
r	Read permission
w	Write permission
x	Execute permission

To change the original permissions on the `save.memo` file so that group members are only allowed to read the file, use this command:

```
$ chmod g-x save.memo
```

The `g` indicates permissions are being changed for members of the group, the minus sign (`-`) indicates removal of a permission, and the `x` indicates `execute` permission.

To remove `read` permission for the group and others on the file, use:

```
$ chmod go-r save.memo
```

or:

```
$ chmod 700 save.memo
```

The listing for the `save.memo` file would now look like this:

```
-rwx------  1 linda staff   69979 Apr  2 14:41 save.memo
```

Moving a File or Directory (`mv`)

On many systems there is a "rename" command and a "move" command. On INTERACTIVE UNIX Systems these two tasks are combined into one command named `mv`. It is used when you want to change the name of a file or directory without making a copy of its contents. It is also used to move the contents of a file or directory from one place in the file system to another.

COMMAND NAME	`mv`
FORMAT	`mv` *filename pathname* `mv` *directoryname pathname*
DESCRIPTION	Move the named file or directory to the named position. If a directory name in the current directory is given as the destination for a file, the full path name is not needed.
OPTIONS	None.
ARGUMENTS	The name of the file or directory you wish to move and its destination.

When you move a file from one directory location to another, be sure that you do not already have a file with the same name in the new location. Like the `cp` command, if a file already exists with that name, it is overwritten. The `mv` command may be used to move a file from one directory to another or to rename a file. You must have the correct file access permissions to move a file from another user's area, i.e., you must have read, write, and execute (`rwx`) permission on the directory where the file resides.

To change the name of a file from `all.emp` to `employees`:

```
$ mv all.emp employees
```

The contents of the file remains the same, but it has been renamed `employees`. There is no longer a file named `all.emp`.

```
$ ls
employees
```

To move the `employees` file into an existing subdirectory called `payroll`:

```
$ mv employees payroll
```

Since you are moving a file into a directory, it is not necessary to supply the file name. The system simply moves the file into `payroll`, retaining the same file name. To rename the `employees` file to `dec.pay` and move it into the `payroll` subdirectory at the same time:

```
$ mv employees payroll/dec.pay
```

Changing Owner (chown)

If you create a directory, you are the owner of that directory. In some cases, however, you may want to change the ownership of a file or a directory to someone else. Do this by using the `chown` command.

COMMAND NAME	`chown`
FORMAT	`chown` *username filename(s)* `chown` *username directoryname(s)*
DESCRIPTION	Change ownership of the named file or directory.
OPTIONS	None.
ARGUMENTS	The name of the user whom you want to own the file(s) or directory(ies) and the name(s) of the file(s) or directory(ies) you wish to change the ownership of.

To change the ownership of your file, `local.wid`, to Tony, use the `chown` command:

```
$ chown tony local.wid
```

To change the ownership of all files in the directory to Tony, use the * wildcard:

```
$ chown tony *
```

Once you have changed the ownership of a file, your access to that file is limited to what the permissions on that file allow. If you are changing the ownership of a file and want to continue to be able to write in it, be sure the permissions are set accordingly before you use `chown`. Of course, the new owner may change the permissions again.

Checking Spelling (`spell`)

The INTERACTIVE UNIX System includes a utility to check the spelling in a file. The spelling checker is not an interactive command, and it does not provide suggestions for alternative spellings.

COMMAND NAME	`spell`
FORMAT	`spell` *filename(s)*
DESCRIPTION	This command checks the spelling of the input file and lists the words that do not appear in its dictionary on the standard output (the terminal).
OPTIONS	Not presented in this document.
ARGUMENTS	Not presented in this document.

It is a good idea to redirect the output of the `spell` command to another file so that you may refer to it. For example, to check the spelling of the file `notes`, redirect the output of `spell` into the `notes.er` file, then `cat` the `notes.er` file:

```
$ spell notes > notes.er
$ cat notes.er
errir
teh
```

Once you have a list of errors, you can use a text editor to correct the spelling errors. If your text editor allows you to alternate between two files or has a windowing capability, it can be very useful to store the errors in a file that you can refer to while you are making corrections. The system does not recognize the acronyms and other special words that you might use, so `spell` may list some words that you know are not really spelling errors.

Viewing a File (`pg`)

You have already learned about the `cat` command as a way to view a file on the screen. The `pg` command provides another way to view a file on the screen. It is more convenient to use because it displays the text one page at a time to allow you time to read the screen. `pg` displays the next page of text when you press ENTER.

COMMAND NAME	`pg`
FORMAT	`pg` *filename(s)*
DESCRIPTION	Display the file(s) on the standard output, one page at a time. Use **ENTER** to display the next page. If a file name is not specified, `pg` reads from the standard input.
OPTIONS	Not presented in this document.
ARGUMENTS	Not presented in this document.

For example, to display the file `letter02`, type:

```
$ pg letter02
```

Printing a File (`lp`)

After you have created a file, you may wish to obtain a printed copy of it. (This printed copy is often referred to as a hardcopy.) Use the `lp` command to print files on your company's printers.

COMMAND NAME	`lp`
FORMAT	`lp` *filename(s)*
DESCRIPTION	`lp` sends the named file to a printing device.
OPTIONS	Not presented in this document. Most offices have several different printers. See your system administrator for the options you should use to access your printers.
ARGUMENTS	The name of the file(s) you wish to print.

The `lp` program queues each request (puts it in a line for processing) and prints the file as soon as the printer is free. A title page with the system date and time is also printed. `lp` can be used in a pipeline with other commands if you want to send the output directly to a printer instead of to a file or the terminal screen. For example, to print the file `letter02`, type:

```
$ lp letter02
```

To get hard copies of two letter files, you can pipe the output of `cat` directly to the printer:

```
$ cat letter01 letter02 | lp
```

Checking on Processes (`ps`)

To see what processes you are currently running, use the `ps` (report process status) command. This can be very useful when you need to stop a process or to check to see if it has stopped prematurely.

COMMAND NAME	`ps`
FORMAT	`ps` [options]
DESCRIPTION	Report the status of all currently active processes on the terminal.
OPTIONS	`-f` Generate a full listing.
ARGUMENTS	Not presented in this document.

For example, suppose you want to check to see if the `spell` command you started is still running in the background. Type `ps`. Your screen will look similar to this:

```
PID TTY      TIME COMMAND
312 ttyi03   0:00 ksh
323 ttyi03   0:00 ksh
325 ttyi03   0:00 spell
326 ttyi03   0:01 deroff
317 ttyi03   0:16 sort
318 ttyi03   0:00 spellpro
320 ttyi03   0:00 comm
321 ttyi03   0:00 tee
327 ttyi03   0:01 sed
331 ttyi03   0:00 comm
332 ttyi03   0:00 tee
333 ttyi03   0:00 ps
```

The spell process is still running and its process ID number is `325`. If you do not see `spell` in the list, then the process has either completed running or died before it completed.

Killing a Process (`kill`)

At times you will want to stop a background process that is currently executing. To do this, use the `kill` command.

COMMAND NAME	`kill`
FORMAT	`kill` [*-signo*] *process_ID_number*
DESCRIPTION	Terminate the named process.
OPTIONS	`-9` Send a sure kill signal.
ARGUMENTS	The ID of the process you want to kill.

To stop a process while you are still logged in, type:

```
kill id_number
```

where *id_number* is the identification number of the process.

If you do not remember the process ID number, use the `ps` (report process status) command described above to find it. Occasionally `kill` alone is not sufficient to stop a process. If this occurs, type:

```
kill -9 id_number
```

USING VIRTUAL TERMINALS

The INTERACTIVE UNIX Operating System contains a very useful feature, the *virtual terminal*, that allows you to log in several times simultaneously on a console terminal. When you switch from one virtual terminal to another, your screen clears and the processes you started in another login session continue to run in the background even while out of view. In one session you can be editing a file, while in others you can be running different programs.

Virtual terminals can be enabled only on a console terminal. By default in the INTERACTIVE UNIX Operating System, you can have one session on a console terminal plus two other virtual terminals. This number can be changed by the system administrator; up to seven virtual terminals can be enabled, for a total of eight possible login sessions.

☛ Any privileges that require use of the console terminal, such as logging in as `root`, cannot be performed during the virtual terminal sessions. You can exercise such privileges only from your console login session.

Switching From One Virtual Terminal to Another

The virtual terminals that are enabled on your system are always ready for use. On most enhanced 386* keyboards, it is necessary to use ALT to perform a SYS-REQ function.

1. To switch to the first virtual terminal, press the ALT and SYS-REQ keys simultaneously, then release them and press F1.
2. When the new prompt appears, you may log in with your usual login ID and password:

```
VT1 login:
Password:
```

3. To switch to the second virtual terminal, again press ALT and SYS-REQ simultaneously, then release them and press F2. Similarly, pressing ALT and SYS-REQ simultaneously, followed by F3, F4, F5, F6, or F7 invokes virtual terminals 3 through 7, respectively. If the corresponding terminal is not enabled on your system, the terminal beeps.
4. To return to the system console, press ALT and SYS-REQ simultaneously, then release them and press F8.

5. You can also cycle forward through all the currently enabled virtual terminals by repeating the following sequence: press ALT and SYS-REQ simultaneously, then release them and press F10. You can cycle backward through all the currently enabled virtual terminals by repeating the following sequence: press ALT and SYS-REQ simultaneously, then release them and press F9.
6. Each virtual terminal behaves as if it were an entirely separate terminal sitting on your desk. Therefore, in order to be completely logged out of the system, you must log out of each virtual terminal that you have logged in to.

Appendix A: ACCESSING DOS FILES

The INTERACTIVE UNIX Operating System provides two mechanisms for accessing MS-DOS* (DOS) files from the INTERACTIVE UNIX Operating System: the DOS File System Support (DOS-FSS) and the Dossette File Exchange Utility (`dossette`).

DOS-FSS allows a DOS file system to function as an integral part of the INTERACTIVE UNIX System partition. This facility allows multiple users to simultaneously access the DOS file system. DOS files need not be copied into the INTERACTIVE UNIX System partition but can be manipulated directly, using path names that are similar to those used on the INTERACTIVE UNIX System. And INTERACTIVE UNIX System commands and utilities can be used with DOS files, just as they would be used with any INTERACTIVE UNIX System file. DOS-FSS is recommended for high-volume situations that require access to and manipulation of many DOS files, using INTERACTIVE UNIX System commands and utilities. DOS-FSS can be used with both fixed disk partitions and diskettes.

`dossette` provides a specific set of commands for manipulating DOS-format file systems and for moving files between DOS and INTERACTIVE UNIX System file systems. `dossette` allows DOS files to be copied into the INTERACTIVE UNIX System partition and edited, then copied back into the DOS file system. Format conversions are performed automatically. `dossette` also includes commands for creating DOS directories, deleting DOS files, and printing DOS ASCII files on the terminal screen. `dossette` is easily invoked and provides a convenient method for accessing DOS files from the UNIX System in low-volume situations that require a limited amount of file manipulation. `dossette` can be used with both fixed disk partitions and diskettes.

USING DOS-FSS

Before You Begin

Under DOS-FSS, a DOS file system must be *mounted* on an INTERACTIVE UNIX System directory before it can be accessed; and a DOS file system must be *unmounted* in order to ensure that all data is written to disk. Superuser privileges are required to mount and unmount a file system. These procedures are usually performed by your system administrator.

☛ Do not remove a DOS file system diskette from the disk drive until it has been unmounted, or you may lose data.

See your system administrator to ensure that your DOS file system is mounted and unmounted correctly. If you are the system administrator, refer to the "INTERACTIVE UNIX Operating System Maintenance Procedures" in the *INTERACTIVE UNIX Operating System Guide* for more information about mounting and unmounting a file system.

DOS File Names

The syntax used for DOS path names is different from that used for INTERACTIVE UNIX System path names. The INTERACTIVE UNIX System uses a series of directory names and (optionally) a file name, each separated by a slash (/). DOS uses a drive name (*device* specifier) followed by a series of directory names and (optionally) a file name, each separated by a backslash (\). (A device specifier usually consists of an alphabetic character followed by a colon. For example, the device specifier `a:` usually indicates the first diskette drive and the device specifier `b:` usually indicates the second diskette drive.)

Your system administrator will mount your DOS file system under an INTERACTIVE UNIX System directory name, such as `/dos`. See your system administrator for the appropriate path name to your DOS file system.

Once the DOS file system is mounted under DOS-FSS, DOS files can be accessed in the same way that INTERACTIVE UNIX System files are accessed. DOS files now function as part of the INTERACTIVE UNIX System file system, so path names need not specify a device name. Instead, they follow the conventions for INTERACTIVE UNIX System path names, i.e., they are made up of directory and

file names, each separated by a slash. The backslash character is not valid. For example, to look at the DOS file `autoexec.bat` in the `/dos` directory, you could use this command:

```
cat /dos/autoexec.bat
```

Note that the path name does not specify a device name and that the components of the path name are separated by slashes.

Under DOS, all file names are mapped to uppercase, regardless of whether they are typed in uppercase or lowercase. DOS-FSS provides this same mapping on input so that users accustomed to this feature can continue to specify DOS files using lowercase. Similarly, DOS file names are displayed in lowercase on output.

DOS-FSS also expands wildcard characters (`*` and `?`). For example, the DOS file `EXAMPLE.TXT` residing in the current directory could be found using any of the following:

```
ls e*
ls e*.*
ls exam???.*
cat e*.txt
```

Refer to the section "Using Wildcards" in this primer for more information about wildcards.

File Conversion Utilities

Text files are stored differently under the DOS and UNIX Systems. The INTERACTIVE UNIX System uses the `new-line` character to specify the end of a line; DOS uses a `carriage-return` followed by a `line-feed` to specify the end of a line. Because of this, DOS files viewed through an INTERACTIVE UNIX System editor display control characters at the end of each line. DOS-FSS provides three utilities to convert DOS files to UNIX System format and UNIX System files to DOS format:

`dtou` Converts DOS files to UNIX System format.

`utod` Converts UNIX System files to DOS format.

`lef` Determines the current format of the specified text file and converts it to the other format.

All three utilities take zero, one, or two arguments. If no argument is specified, the utility takes input from standard input (usually your keyboard) and sends output to standard output (usually your display

screen). If one argument is specified, it is assumed to be the input file, and the output is sent to standard output. If two arguments are specified, the first is taken to be the input file and the second is taken to be the output file. The following examples illustrate the command format of the conversion utilities.

This command converts the DOS text file `example.txt` to a UNIX System text file and displays it on the standard output:

```
dtou /dos/example.txt
```

This command converts the UNIX System text file `doc.txt` to a DOS text file in the `/dos` directory:

```
utod /tmp/doc.txt /dos/doc.txt
```

This command uses the `lef` utility to determine the format of the `doc.txt` file, convert it from the current format to the other format, and place the new file in the `/tmp` directory:

```
lef /dos/doc.txt /tmp/doc.txt
```

Permissions

The INTERACTIVE UNIX System is a *multi-user* environment that makes use of such concepts as owners of and permissions for files and directories. (Refer to the section "Changing Permissions (`chmod`)" for more information about file owners and file permissions.) DOS is a *single-user* system and has no equivalent concepts. In order to support multi-user access to a mounted DOS file system, DOS-FSS attempts to apply these UNIX System concepts to the DOS file system.

If you cannot access a file or directory on your DOS file system, the permissions may have been set incorrectly. See your system administrator. If you are the system administrator, refer to "INTERACTIVE UNIX Operating System Maintenance Procedures" in the *INTERACTIVE UNIX Operating System Guide* for more information about permissions.

USING DOSSETTE

Device Specifiers

Like DOS, `dossette` recognizes certain device specifiers. For example, the specifier for the first diskette drive is `a:`, the specifier for the second diskette drive is `b:`, and the specifier for the DOS partition on the first fixed disk is `c:`. If you attempt to access a diskette or fixed disk partition that is not in DOS format while using `dossette`, an error message will display. When `dossette` is initially invoked, the current device is always set to `a:`, but it can be changed in the usual DOS fashion by entering a different device specifier, followed by ENTER.

DOS File Names

The format for a DOS file name includes an optional device specifier, an optional path name, and the name of the file. If the device specifier is omitted, the current device is assumed. If no path name is specified, the current directory on the specified device is assumed. The components of a path name are each separated by a slash (`/`). (The backslash character is not valid in `dossette`.) For example, the file name `temp` indicates the file `temp` in the current directory on the current DOS device; the file name `b:/util/temp` indicates the file named `temp` in the directory `util` in a DOS file system on the second diskette drive.

Interactive Commands

When the `dossette` command is typed at the prompt, `dossette` is invoked in interactive mode. In this mode, commands that are similar to DOS commands are read from standard input, and each line is executed as it is read. `dossette` prompts the user for input, as necessary. The program mimics DOS in issuing a prompt at the beginning of each line to indicate the current device, for example, `A>`, `B>`, `C>`. To invoke `dossette` interactively, type `dossette` at the shell prompt:

```
$ dossette

A>
```

`dossette` is now ready to receive any of the following commands:

`dir`	Lists DOS directory information.
`type`	Prints a DOS ASCII file on the terminal screen.
`copy`	Copies a DOS file to a new DOS file name. You may also copy a DOS file to a DOS directory.
`rename`	Changes the name of a DOS file.
`erase, del`	Deletes a DOS file.
`mkdir, md`	Creates a DOS directory.
`rmdir, rd`	Removes a DOS directory.
`format`	Physically formats a DOS diskette. This command does not format a fixed disk partition.
`get`	Copies a DOS file or files to a UNIX System file system and changes the file or files to UNIX System format.
`put`	Copies a UNIX System file or files to a DOS file system and changes the file or files to DOS format.
`cd`	Changes or displays the DOS working directory.
`chloc`	Changes the INTERACTIVE UNIX System working directory.
`a:`	Changes the current DOS device to drive `A`, which corresponds to the first diskette drive.
`b:`	Changes the current DOS device to drive `B`, which corresponds to the second diskette drive.
`c: . . . q:`	Changes the current DOS device to the fixed disk partition corresponding to the device specifiers `c:` through `q:`.
`help`	Displays a list of available `dossette` commands on the screen.
`q`	Exits `dossette`.

The following examples illustrate how to use the `dossette` commands interactively.

The `get` command allows you to copy a DOS file into the INTERACTIVE UNIX System partition and converts the DOS control characters to UNIX System format:

```
get b:/util/tmp /src/unix.tmp
```

In this example, the DOS `tmp` file, located in the `util` directory on the second diskette drive, is copied to the UNIX System file, `unix.tmp`, in the UNIX System `src` directory. The file is converted to UNIX System format. Once the file is converted in this way, it can be edited using a UNIX System editor.

The `put` command allows you to copy a UNIX System file back to the DOS partition or diskette and converts the file to DOS format:

```
put /src/unix.tmp b:/util/dos.new
```

In this example, the `unix.tmp` file, located in the `src` directory, is copied to the DOS file, `dos.new`, in the DOS `util` directory on the second diskette drive. The file is converted to DOS format.

To exit `dossette`, type `q` at the prompt.

Batch Mode

The `dossette` command group is also available for use in *batch* (noninteractive) mode so that you can use the `dossette` commands in a shell script. In this mode, the `dossette` command line consists of the invoking command and its arguments, if any. The `dossette` batch commands are prefixed by `dos`; they operate in the same way as the interactive `dossette` commands described previously. The `dossette` commands available in batch mode are: `dosdir`, `dostype`, `doscopy`, `dosrename`, `dosmkdir`, `dosrmdir`, `dosformat`, `dosget`, and `dosput`.

This example illustrates how to use the `dosget` batch command. Note that the command and its arguments are typed at the shell prompt:

```
$ dosget b:/util/tmp /src/unix.tmp
```

In this example, the DOS `tmp` file, located in the `util` directory on the second diskette drive, is copied to the UNIX System file, `unix.tmp`, in the INTERACTIVE UNIX System `src` directory. The file is converted to UNIX System format.

GLOSSARY

absolute path name
See *full path name*.

append
To add text to the end of an existing file.

argument
A string of text that accompanies a command and gives the computer additional information to modify the result of the command.

background
Not in the foreground, i.e., not interfering with current work on the terminal. A program that is executing in the background allows the user to continue to issue commands and interact with the computer system. To execute a program in the background, type an ampersand (&) at the end of the command.

batch mode
A `dossette` interface that permits commands to be invoked noninteractively in shell scripts.

boot
To load an operating system or a standalone program into memory and execute it.

Bourne shell
The Bourne shell is considered the standard UNIX System shell and is available on UNIX System V.3. It was written at Bell Laboratories and released by AT&T.

case sensitive
Distinguishes between uppercase and lowercase.

command
An instruction the user gives to the computer. The command is interpreted by the user interface, which instructs the computer to run the program that will perform the task requested by the command. See also *argument* and *option*.

command interpreter
: A command interpreter passes the commands you enter to the operating system for processing and delivers the results to you. See also *shell*.

command line
: The complete instruction the user gives to the computer, including the command name, options, arguments, and pipes.

console
: The terminal directly attached to the computer unit.

C shell
: A UNIX System shell written and released by the University of California at Berkeley. It has gained widespread popularity, particularly with programmers.

current working directory
: The directory in which you are located. Its path name is returned when you use the `pwd` command.

cursor-positioning keys
: The arrow keys on your keyboard that are usually used to move the cursor when editing text.

default
: The alternative chosen by the system when no choice is specified by the user.

device-specifier
: An alphabetic character followed immediately by a colon, which is used under DOS and with `dossette` to indicate a particular diskette drive or fixed disk partition. The device specifier `b:` usually indicates the second diskette drive.

directory
: A special type of file that contains other files and/or directories. A typical directory contains related documents, such as memoranda or monthly sales reports.

directory files
: See *directory*.

editor
: A utility or program written to facilitate the modification and manipulation of text files.

error message
: A message from the operating system that appears on your screen in response to a problem the computer has encountered with a command.

execute permission
: The third permission in each set of file or directory permissions. Execute permission allows a user to execute a file or search a directory.

file
: A document or a collection of information stored on the computer. For example, a file could be a list of phone numbers, a memorandum, or a report.

file access permissions
: See *permissions*.

file system
: A collection of individual files and directories that are stored on a portion of a disk.

filters
: Programs that accept their input from one source, such as the standard input, perform their appointed task on the data, and then write their results to the standard output without changing the input file in any way.

flag
: An option to a command. See *option*.

full path name
: The full path name of a file completely describes the location of the file in the system. It is the sequence of directories from the `root` directory to the file or directory you wish to reference. It consists of a slash (`/`), followed by one or more directory and file names, separated by slashes.

full-screen editor
: A text editor that shows the changes you are making to a file on the screen as you make them.

group

A group is a collection of users. Permission for a group refers to the second set of three permissions for a file or directory. These apply to members of the owner's group only.

hardware

The physical components of a computer. Examples include the keyboard, the screen (sometimes called the display or the monitor), and the printer.

hierarchical file system

A directory structure that is arranged in a ranked series, with a single master directory at the top level and additional levels of directories or files defined beneath it.

home directory

The directory that contains the "master list" or index of the information you are storing on the computer. Most systems automatically place you in your home directory when you log in.

interactive

To carry on a dialogue with the computer.

I/O redirection

The capability of specifying the standard input and output at the time of executing a command.

kernel

That part of the UNIX Operating System that performs the system management aspects.

line editor

A text editor that requires that you explicitly specify the line or lines you want to edit and does not show the changes you make to your file as they occur. A line editor is useful if you are using a hard copy terminal, i.e, one where you do not have a screen.

log in

To type your login name and password onto your computer or terminal, indicating that you are ready to gain access to your information.

login account
: The information stored in a computer that provides authorization for a person to use that computer's resources. The computer's system administrator usually sets up login accounts.

log out
: To terminate your access to the system. On most systems, this is accomplished by holding down the CTRL key while simultaneously typing `d`.

manual entry
: A technical description and summary of use for UNIX System commands, system calls, subroutines, file formats, miscellaneous facilities, and special files found in the UNIX System reference manuals.

mount
: A system administration procedure used to attach a file system to other directories of the main system. A file system must be mounted in order to be accessible to users.

multi-tasking
: Capable of running many different processes (programs) at the same time.

multi-user
: Capable of supporting more than one user at a time. On a multi-user computer system, each user has his or her own terminal plugged into the computer, and all can share its data and resources at the same time.

octal
: The numbering system of the base 8 that uses the numbers 0, 1, 2, 3, 4, 5, 6, and 7.

operating system
: The operating system is part of the system software. It is a collection of instructions written in some machine or programming language that tells the computer how to manage its operations, process and execute the user's requests, and run application software.

option
: An optional argument that is available to modify the results of a specific command. Options are usually preceded by a dash (–).

other
: The third set of three permissions for a file or directory, which applies to everyone on the system who is neither the owner nor in the owner's group.

overwrite
: To perform an operation on a file that deletes its contents and replaces it with the output of the operation. For example, if you `cat` a file and use the name of an already existing file as its destination, the contents of the original file is replaced with the contents of the new file.

owner
: The user who creates a file and has control over the file's permissions.

parent directory
: The directory immediately above the one in which you are located.

path name
: The sequence of directory names and (optionally) a file name, separated by slashes, that describes the location of a file or directory on a UNIX System. There are two types of path names, the full path name and the relative path name. The full path name consists of the sequence of directories from the `root` directory to the file or directory you wish to reference. The relative path name omits the `root` directory and directory names up to the current directory.

pattern matching character
: A wildcard character. A wildcard character is used to match a character or a string of characters in a file or directory name. It is another type of shortcut.

permissions
: Permissions determine who can read, write, or execute a file or directory. They are indicated by the letters r, w, and x obtained when you list the contents of a directory using the long option.

pipeline
: To take the output of one program and direct it to be the input to another program.

pipes
: Connections between two or more UNIX System commands, indicated by a vertical bar symbol (|).

program
: A set of instructions for a specific set of tasks, written in a machine or programming language, that tells the computer how to perform the tasks.

prompt
: A symbol that displays on the screen to indicate that the system is ready to receive your commands. You may execute a command or run an application when the prompt is displayed.

protection mode
: The permissions on a file or directory, which determine how it is "protected" from the owner, group, and others. See also *permissions*.

read permission
: The first permission in each set of file or directory permissions. Read permission allows a user to view the contents of a file or directory.

redirect
: To change the destination of the output or the input of a command.

relative path name
: The path name of a file or directory, omitting the `root` directory and directory names up to the current directory.

`root` *directory*
: The top-level directory in a UNIX System, designated by a slash (/). All other directories in the system connect directly or indirectly to the `root` directory.

shell
: That part of the UNIX System that interacts with the user. It is also called a command interpreter or user interface. The two major ones in use are the Bourne

shell and the C shell. The shell prompts you for commands, "interprets" your commands for the computer, and sees that it carries out the task you have requested.

shell program
See *shell script*.

shell prompt
The computer symbol that appears on your screen to indicate that the shell is ready to receive your commands. You may execute a command or run an application as soon as you see it.

shell script
A simple collection of shell commands normally executed on the command line, or a program that uses programming logic, variable assignment, and argument processing to perform complex tasks. Shell scripts are usually created by the user to perform frequently needed complex tasks.

single-user
Capable of supporting only one user at a time. On a single-user computer operating system, no one else can simultaneously share the computer's data or resources. Most small personal computers are single-user systems.

software
The instructions that make a computer perform its functions. Software is divided into two main groups: application and system. Application software performs specific tasks, such as word processing, spreadsheets, educational programs, and games. System software includes operating systems that tell the computer how to run application software.

standard input
Information coming from the keyboard, unless otherwise specified by the user. The user can specify that information come from a file, a device, or a pipe as well.

standard output
: The destination of a program's data, considered to be the terminal screen, unless otherwise specified by the user. The user can specify that information be written to a file, a device, or a pipe as well.

string
: A series of characters surrounded by spaces.

system software
: System software is composed of programs and utilities that manage the computer's resources and run application software. See *operating system*.

unmount
: A system administration procedure used to detach a mounted file system from the main system and make it inaccessible to users.

user interface
: The command interpreter, or that part of the operating system with which the user interacts.

virtual terminal
: A feature that allows you to log in several times simultaneously on a console terminal and switch back and forth between the login sessions. Processes continue to run in the background even while out of view.

wildcard
: A wildcard character is used to match a character or a string of characters in a file or directory name. It is a type of shortcut.

write permission
: The second permission in each set of file or directory permissions. Write permission allows a user to modify (edit) the contents of a file or create and delete files in a directory.

INDEX

System Administration for New Users of the INTERACTIVE UNIX Operating System

CONTENTS

System Administration for New Users of the INTERACTIVE UNIX* Operating System

1. INTRODUCTION

This document discusses some of the most basic procedures required to keep your INTERACTIVE UNIX Operating System running smoothly and provides tutorial information for performing some of those fundamental system administration tasks. It is intended for people who are responsible for running an INTERACTIVE UNIX System who have little or no experience with any UNIX-based operating system. It does *not* provide complete instructions for all of the system management procedures required to maintain the operating system. It supplements the technical information found in the *INTERACTIVE UNIX System User's/System Administrator's Reference Manual* and the "INTERACTIVE UNIX Operating System Maintenance Procedures" in the *INTERACTIVE UNIX Operating System Guide*, which together provide more complete and detailed technical information on all aspects of installing, configuring, and maintaining an INTERACTIVE UNIX System.

1.1 Before You Begin

Before you attempt any of the procedures outlined in this document, you must read and understand the material in the "INTERACTIVE UNIX Operating System Primer." You must also be familiar with a text editor if your particular hardware and software configuration requires you to tailor some of the standard configuration files. Refer to the "Documentation Roadmap" in the *INTERACTIVE UNIX Operating System Guide* to determine where to obtain additional information about text editors available for your system or additional information about UNIX-based systems, including the INTERACTIVE UNIX System. You may also need to refer to the "INTERACTIVE UNIX Operating System Maintenance Procedures" in the *INTERACTIVE UNIX Operating System Guide* or to any additional documentation supplied by your vendor to completely tailor your system.

Manual entries are referenced in this document using the form *entryname*(n), where *entryname* is the name of the entry and n is the number of the section. All entries are in the *INTERACTIVE*

UNIX System User's/System Administrator's Reference Manual unless explicitly stated otherwise.

1.2 Overview of This Document

This document is divided into 14 major sections:

1. INTRODUCTION
   This section provides a general overview of this document.

2. WHAT IS A SYSTEM ADMINISTRATOR?
   This section explains the basic duties of the individual responsible for installing and maintaining a system.

3. SHUTTING DOWN AND BRINGING UP THE SYSTEM
   This section explains how to boot the system, how to shut it down gracefully, and how to reboot the system.

4. SPECIAL USERS ON THE INTERACTIVE UNIX OPERATING SYSTEM
   This section lists the privileged users and tells how to access those accounts.

5. THE SYSTEM ADMINISTRATION PROGRAM
   This section gives a very general description of the System Administration program available with this system and how to access its different levels using the `sysadm` command.

6. MANAGING USER ACCOUNTS
   This section describes the attributes of each user login and explains how to use the `sysadm` command to add, change, or delete user accounts.

7. SETTING UP YOUR PRINTER
   This section describes how to configure your software to add a printer to your system using the `sysadm` command.

8. TAILORING THE ENVIRONMENT
   This section describes the environment variables, the most commonly configured system files, and the `cron` program. It also explains how to set the date and time and change the message of the day.

9. UNDERSTANDING INTERACTIVE UNIX SYSTEM FILE SYSTEMS

   This section explains the structure of INTERACTIVE UNIX System file systems and describes the conventions used to name file systems and devices.

10. USING UTILITIES WITH THE DISKETTE DRIVE

    This section explains how to format diskettes and how to copy files to diskettes.

11. FILE SYSTEM MAINTENANCE

    This section explains how to create and maintain INTERACTIVE UNIX System file systems.

12. BACKING UP FILES

    This section describes how to back up files.

13. MANAGING DEVICES

    This section gives a brief explanation of devices, device drivers, the kernel, and the High Performance Disk Driver.

14. ADMINISTERING SYSTEM SECURITY

    This section discusses general requirements for system security and basic precautions.

A GLOSSARY is also provided.

2. WHAT IS A SYSTEM ADMINISTRATOR?

The *system administrator* is the individual responsible for installing, administering, and maintaining your INTERACTIVE UNIX Operating System. If the INTERACTIVE UNIX System is installed on a personal computer, you will probably act as the system administrator. If your personal computer is part of a larger network of computers, a system administrator from the computer operations department of your company may be assigned to help you keep your system running smoothly. This document assumes that *you* will be responsible for basic system administration tasks.

A system administrator can have many responsibilities, but there are several fundamental tasks that must be performed after your system is initially installed. You will be responsible for:

- Managing user accounts
- Tailoring the system
- Installing new software packages
- Installing new hardware devices and their associated drivers
- Maintaining file systems
- Performing routine maintenance procedures, such as backing up files

You should use the *console terminal* to perform all system administration tasks. The console terminal is comprised of the screen and keyboard directly attached to the Central Processing Unit (CPU). It is important for the system administrator to use the console terminal since most error messages generated by the system are displayed on the console terminal screen.

The following sections provide you with a conceptual overview of each of the topics listed above. Where appropriate, tutorial instructions are included. You may also want to refer to the "INTERACTIVE UNIX Operating System Maintenance Procedures" for more information when you are using your computer.

3. SHUTTING DOWN AND BRINGING UP THE SYSTEM

The procedures for shutting down and bringing up the system are already familiar to you if you have followed the installation instructions outlined in the "INTERACTIVE UNIX Operating System Installation Instructions."

3.1 Shutting Down the System

When you are ready to turn off your computer, you must arrange to have the computer complete all of the tasks that are currently running. This is accomplished with a system maintenance procedure called `shutdown`. The `shutdown` program gracefully terminates all tasks that are currently executing before it halts the system. When `shutdown` has finished, you may safely turn off the computer. If you do not run the `shutdown` program, you may lose data that is stored on your system and cause damage to your file system, even if no one is currently logged in.

The `shutdown` program can be run in two ways:

1. Use the `powerdown` administrative login.
2. Execute the `shutdown` command.

3.2 Using the `powerdown` Administrative Login

When you are ready to turn your computer off, you may use the `powerdown` administrative login to bring the system down.

1. Log out of your ordinary user account.
2. Log in to the system with the `powerdown` user ID.

   ☛ You must remember the `powerdown` password if you set one during the installation.

3. Once you have successfully logged in to the system using the `powerdown` login, the system automatically executes the `shutdown` program. The system displays a screen similar to this:

```
login:  powerdown
Password:
UNIX System V/386 Release 3.2
Copyright (C) 1984, 1986, 1987, 1988 AT&T
Copyright (C) 1987, 1988 Microsoft Corporation
All Rights Reserved

Login last used: Mon Sep 19 12:24:32 1988

/ : Disk Space: 4.86 MB of 26.96 MB available (18.06%)

Total Disk Space: 4.86 MB of 26.96 MB available (18.06%)
```

4. Press ENTER. The system displays:

```
Once started, a powerdown  CANNOT BE STOPPED.
Do you want to start an express powerdown [y, n, ?, q]
```

5. To bring the system down, type `y`. The system responds:

```
Shutdown started.  Wed Jun  3 17:31:44 PDT 1988

Broadcast message from root (console) on bud
Wed Jun 3 17:31:44
THE SYSTEM IS BEING SHUT DOWN!!!
Log off now or risk your files being damaged.

INIT: New run level 0
The system is coming down, please wait.
System services are now being stopped.
cron aborted: SIGTERM
! SIGTERM Wed Jun  3 17:31:44 1988
! ******* CRON ABORTED ******* Wed Jun  3 17:31:44 1988
Print services stopped.
Stopping Network Processes...


The system is down
Press any key to reboot.
```

6. When the `Press any key to reboot` message appears, the computer may be safely turned off.

3.3 Using the `shutdown` Command

To execute the `shutdown` command manually, you must log in to the system as the `root` user on the console terminal and you must be in the `root` directory.

☛ The `root` user is the most privileged user on the system; this login ID is commonly known as the *superuser*. There are no restrictions imposed upon `root`, which means that the person logged in to the system with the `root` ID can access, modify, and delete every file and process on the system. The `root` login is discussed in more detail in section 4.1.1. When you are

logged in as `root`, your login prompt is a pound sign (#). To execute the `shutdown` command manually:

1. Log in as `root`. Your screen will look similar to this:

```
login:  root
Password:
UNIX System V/386 Release 3.2
Copyright (C) 1984, 1986, 1987, 1988 AT&T
Copyright (c) 1987, 1988 Microsoft Corporation
All Rights Reserved

Login last used: Mon Sep 19 12:24:32 1988

#
```

   The # prompt indicates that the system is ready for you to type in a command.

2. Move to the `root` directory by typing:

```
# cd /
```

   Refer to the "INTERACTIVE UNIX Operating System Primer" for more information about directories.

3. Run the `shutdown` program by typing:

```
# shutdown
```

   The system automatically generates a message on every terminal currently in use to warn users that the system is being shut down. The message will look similar to this:

```
Shutdown started.  Wed Jun  3 17:31:44 PDT 1988

Broadcast message from root (console) on bud
Wed Jun 3 17:31:44
THE SYSTEM IS BEING SHUT DOWN!!!
Log off now or risk your files being damaged.
```

   The system gives users 1 minute to exit editors and save files. After this 1 minute warning period is over, the `shutdown` program prompts you for confirmation that you want to continue. Your screen will look similar to this:

```
Do you want to continue (y or n):
```

4. Type `y` if you want to continue shutting down the system, `n` if you want to stop. If you type `y`, the system is then shut down. You may press any key to reboot the system or turn off the power to the computer at this point. If you type `n`, the system issues the following message to all users currently logged on to the system:

```
Broadcast message from root (console) on bud
Wed Jun 3 17:31:44
False Alarm:  The system will not be brought down.
```

The system then aborts the shutdown and displays the following message on the console:

```
Shutdown aborted.
```

5. If you wish to give users a longer warning period before the system goes down, run the `shutdown` program using the `-g` option:

```
# shutdown -gtime -y
```

In this example, *time* represents the number of seconds the system will wait before it goes down. It is a good idea to allow at least 2 minutes (120 seconds) to elapse before the system is brought down. Using the `-y` option causes the shutdown procedure to continue directly without prompting you for permission to proceed.

After the warning period, the system automatically runs `shutdown`. Your screen will look similar to the one generated by the `powerdown` procedure:

```
Shutdown started.  Wed Jun  3 17:31:44 PDT 1988

Broadcast message from root (console) on bud
Wed Jun 3 17:31:44
THE SYSTEM IS BEING SHUT DOWN!!!
Log off now or risk your files being damaged.

INIT: New run level 0
The system is coming down, please wait.
System services are now being stopped.
cron aborted: SIGTERM
! SIGTERM Wed Jun  3 17:31:44 1988
! ******* CRON ABORTED ******* Wed Jun  3 17:31:44 1988
Print services stopped.
Stopping Network Processes...


The system is down
Press any key to reboot.
```

Be sure that you do not attempt to power down or reboot the system until the above completion message is displayed.

See *shutdown*(1) for more information.

3.4 Bringing Up the System

To bring up or *reboot* the system (as it is frequently called), use this procedure:

1. First check the diskette drives and remove any diskettes there. If the system finds a diskette when it is first turned on, it will attempt to boot from it. If it does not find a diskette, it will boot from the fixed disk when the power is turned on.
2. *If your computer is still turned on*, either turn it off, then turn on the power again, or press any key to reboot.
3. *If your system power is turned off*, turn on the power. The INTERACTIVE UNIX Operating System automatically boots.

   When it reboots, the system does the following:

   - Runs internal diagnostic programs to check your hardware
   - Displays memory and copyright information and the message:

     ```
     Booting the UNIX System...
     ```

   - Determines if the system was shut down properly
   - Checks the file systems for inconsistencies and attempts to make any necessary repairs
   - Starts various system processes, such as the line printer scheduler

If your system has crashed for some reason, the system may ask if you want to save a *system dump.* A system dump is used by experienced system administrators to obtain detailed information about the state of the system when it crashed. Under most circumstances, you will want to answer n to this prompt.

4. SPECIAL USERS ON THE INTERACTIVE UNIX OPERATING SYSTEM

There are three types of login accounts on an INTERACTIVE UNIX Operating System: an *ordinary* login account, a *system* login account, and an *administrative* login account. A login account gives access to the system.

Ordinary logins include a login ID, a password, and an area of the file system "assigned" to the user, called a *home directory*. Every user on the system (including the system administrator) uses an ordinary login account to perform most tasks. These accounts are used to store data and run application programs, such as an accounting or word processing program. Ordinary logins may have restricted access to other user login accounts and to system administration functions. The procedure for creating an ordinary user account is described in section 6.4.

System logins are used to perform system administration tasks that require privileged access to the restricted files and directories on the system. You typically use a system login when you need to perform system maintenance tasks for the operating system or hardware facilities on your system. These logins should be used with great care. System login accounts exist in the INTERACTIVE UNIX System; you should already have set passwords for these accounts when you installed the system.

Administrative logins are used to give ordinary users restricted access to system management functions that must be performed frequently. For example, there are administrative logins available for adding new users or shutting down the system. Administrative login accounts exist in the INTERACTIVE UNIX System; you should already have set passwords for these accounts when you installed the system.

4.1 System Login Accounts

When you initially installed your system, several system logins were automatically created for you. The two most important system logins are `root` and `bin`. They are used to perform most privileged system administration functions. Other system logins are available to monitor and manage continuously running processes, to run the program that updates the fixed disk, and to manage networking operations.

☛ All system logins should be used *only when it is absolutely necessary.*

System logins have few restrictions imposed upon them by the operating system, which means that they can cause serious damage to your system when used incorrectly. It is very important to provide a password for each of the system logins (see section 4.6.1 in the "INTERACTIVE UNIX Operating System Installation Instructions"). Passwords for system logins should only be given to the individual(s) designated as the system administrator(s) for your system. Each system login is described below.

4.1.1 The `root` *Login*

The most important INTERACTIVE UNIX System login belongs to the `root` user. This login account, also known as the superuser, is the most privileged and powerful account on the system. There are *no restrictions* imposed upon `root`, which means that individuals who have access to the `root` login also have access to every other account on the system. The `root` user can access, modify, and delete every file and process on the system.

☛ It is very important to:

- Assign a password to the `root` account.
- Change the `root` password frequently.
- Distribute it only to only a few individuals who are responsible for system administration tasks.
- Use it *only* for system administration tasks.
- Be very careful when using it.

You must be at the console terminal to log in to the system as `root`. This is an additional safeguard that helps limit the use of the `root` login. You will use the `root` login to configure the system kernel, add new peripheral devices (like printers), and mount or unmount file systems.

Once you have logged in to the system using the `root` login, the system displays the pound sign (#) as the prompt.

4.1.2 The `bin` Login

The `bin` system login owns most of the command files and directories on the system, including all of the special files used to access fixed disks and diskettes. The `bin` login account has the same privileges as an ordinary user, but also owns the directories `/bin`, `/usr/bin`, `/lib`, and `/etc`. You must log in as `bin` to install new programs and libraries in these directories. You can use the `bin` login to install new software or to modify the files and directories owned by `bin`. This login is also used to limit user access to files and directories that can affect the entire system.

4.1.3 Other System Logins

The system automatically provides several other system logins: `daemon`, `sync`, `uucp`, and `nuucp`. You should assign a password to these accounts, but you should never need to log in using `daemon`, `sync`, or `nuucp`. These system logins and their functions are described below.

`daemon` A *daemon* is a program that executes continuously on the system. For example, the program responsible for queuing print jobs is a daemon. This login is provided because certain programs on the system must be owned by `daemon`.

`sync` This login executes the `sync` program, which updates the fixed disk every 10 seconds with the work currently in progress.

`uucp` This login owns the files in the directory `/usr/lib/uucp`. It may occasionally be necessary to log in as `uucp` to change these files, which are used in the UNIX-to-UNIX-copy file transfer programs.

`nuucp` This login is used by remote machines to log in to your computer system and to initiate UNIX-to-UNIX file transfers.

4.2 Administrative Login Accounts

There are a number of tasks that require privileged access to the system. Because the `root` login is so powerful, it is not a good idea to use it to perform all of the administrative tasks that are

required to keep the system running properly. Instead, several administrative logins are provided with your system to facilitate system maintenance. As with all sensitive login accounts, you should set a password for each login and limit the password's distribution. The administrative login accounts are described below.

`sysadm` This login account gives you direct access to the `sysadm` facility. `sysadm` is a menu-driven facility that may be used to perform most system administration procedures. It is described in section 5.

`powerdown`
The `powerdown` login may be used to shut down the system. All the processes running on an INTERACTIVE UNIX System must be gracefully terminated before the computer is turned off, in order to prevent serious damage to the file system or loss of data. The `powerdown` login is provided to facilitate this process. Refer to the "INTERACTIVE UNIX Operating System Installation Instructions" or section 3.2 for more information.

`checkfsys`
`makefsys`
`mountfsys`
`umountfsys`
These four logins may be used to directly access the file system maintenance options that are available as part of the `sysadm` login and *utility*. (A utility is a program, usually from a set of programs, that represents a specific application available with your computer.) You can use the `sysadm` login to access these options as well. Refer to sections 5 and 11 for more information.

4.3 Accessing Other User Accounts (the `su` Command)

If you know another user's login name and password, you can use the `su` command to temporarily log in to that user's account and gain access to protected files, directories, and programs. This can be very useful, for example, if you need to temporarily access either a system or an administrative login without logging out. To access another user account without logging out, follow these instructions:

1. Use the `su` command to acquire `root`'s privileges without logging out. (Note that although you have to be at the console terminal to *log in* as `root`, you do not have to be at the console terminal to `su` to `root`.) Type:

   ```
   $ su - root
   ```

   and press ENTER.

2. If a password is assigned to that account, the system asks for a `Password:`. Enter the correct password and press ENTER.

3. The system responds with the # prompt. You now have all the privileges of the `root` user, and you are placed in `root`'s home directory, `/`.

4. Type `exit` or press CTRL d simultaneously to exit the `root` account and return to your normal user account.

   This same procedure can be used to access any user account for which you know the correct login name and password.

5. THE SYSTEM ADMINISTRATION PROGRAM

5.1 Introduction

The System Administration program is a standard feature of the INTERACTIVE UNIX Operating System. It is a menu-driven interface that makes it easy to perform commonly required system administration tasks. This section provides an overview of each System Administration submenu and its purpose. The use of specific System Administration procedures is discussed in the appropriate task-related sections of this document and in the "INTERACTIVE UNIX Operating System Maintenance Procedures."

☛ This document does not attempt to describe every possible maintenance procedure available through the System Administration menu. As you become familiar with this interface, you may wish to explore the usage of some of the menu items that you see that are not described here by simply trying them out. The system prompts you step-by-step through the procedures.

`sysadm` is both a login account and a utility program. This means that you may access `sysadm` either by logging in to the system with the `sysadm` login or by typing the `sysadm` command when you are already logged in to the system. If you log in as `sysadm`, the system displays a screen similar to this:

```
login: sysadm
Password:
```

You must know the `sysadm` password to use this login. If you are already logged in, type `sysadm` at the shell prompt:

```
$ sysadm
```

5.2 The Main Menu

When you type the `sysadm` command, the Main (or first level) System Administration menu is displayed. If you set a password for the `sysadm` login when you installed the system, you are prompted for a password as usual. The system displays:

```
                    SYSTEM ADMINISTRATION

  1 diskmgmt        disk management menu
  2 filemgmt        file management menu
  3 machinemgmt     machine management menu
  4 packagemgmt     package management menu
  5 softwaremgmt    software management menu
  6 syssetup        system setup menu
  7 ttymgmt         tty management menu
  8 usermgmt        user management menu

Enter a number, a name, the initial part of a name, or
? or <number>? for HELP, q to QUIT:
```

Each menu option represents a category of tasks that are usually performed by the system administrator. When you select an option, the system displays a new submenu that lists the tasks that are specific to that option. When you select a task from a submenu, the system either provides step-by-step procedures for performing the specified task or displays another submenu.

The following menus are available:

- **Disk Management Menu** (`diskmgmt`)
  The Disk Management menu is used for tasks related to your fixed disks and diskettes. Some of the tasks you can perform from this menu include copying diskettes, checking the integrity of diskette-based file systems, deleting data from diskettes, and making new diskettes accessible to the system. You may also use this submenu to perform similar tasks on a fixed disk, to inform the system about errors on your fixed disk, or to determine the current partition configuration of your fixed disk.

  Refer to section 11, "FILE SYSTEM MAINTENANCE," for information about using the Disk Management Menu and to the "INTERACTIVE UNIX Operating System Maintenance Procedures" for information about manipulating file systems on your fixed disk.

- **File Management Menu** (`filemgmt`)
  The File Management menu is used to back up, restore, and monitor file systems. Use this menu to back up data from your fixed disk onto a diskette, transfer data from a diskette to a fixed disk, and monitor files and available space on the file system.

  Refer to section 12, "BACKING UP FILES," for information about using the File Management Menu.

- **Machine Management Menu** (`machinemgmt`)
  The Machine Management menu is used to find out who is logged on to your system.

- **Package Management Menu** (`packagemgmt`)
  The Package Management menu is used to add a printer to your system and to configure the standard networking utility available for the INTERACTIVE UNIX System `uucp` (unix-to-unix copy). Refer to section 7, "SETTING UP YOUR PRINTER," for information about adding a printer to your system. Refer to section 10, "BASIC NETWORKING ADMINISTRATION," in the "INTERACTIVE UNIX Operating System Maintenance Procedures" for more information on the installation and configuration of `uucp`.

  This menu is also used to configure the extended networking utilities if these are available with your system. There may be additional entries on this menu, depending on the system you have purchased and the INTERACTIVE software extensions you have installed. Refer to the documentation that accompanied your extension for information about using `sysadm` to configure its software.

- **Software Management Menu** (`softwaremgmt`)
  The Software Management menu is used to list, install, and remove software packages and extensions. It can also be used to run a software package without installing it.

  The installation of software packages is discussed in section 6 of the "INTERACTIVE UNIX Operating System Installation Instructions."

- **System Setup Menu** (`syssetup`)
  The System Setup menu can be used to initialize your system the first time you install it or to change the current administrative and system passwords, date and time, and name of your machine at a later time. Most of these tasks are included in the current system installation procedure, so it is only necessary to use this menu if you want to change these parameters after your system is up and running.

- **tty Management Menu** (`ttymgmt`)
  The TTY (Terminal) Management menu is used to configure the lines that connect your terminal, modem, or other device to the main CPU. These lines are known as *tty* lines (terminal type).

They must operate at certain speeds, or *baud rates*, which must be set for each new device that is connected to a tty line. Refer to section 9.6.3, "Configuring Your Computer With Additional Terminals," in the "INTERACTIVE UNIX Operating System Maintenance Procedures" for more information.

This menu is also used to change the number of virtual terminals available at the console terminal. Refer to section 7.10.3.1, "Enabling Virtual Terminals," in the "INTERACTIVE UNIX Operating System Maintenance Procedures" for more information about configuring virtual terminals.

- **User Management Menu** (`usermgmt`)
  The User Management menu is used to establish and configure user login accounts. Among other things, it can be used to list, add, and delete users and groups. This menu is described in more detail in section 6.4, "Managing User Accounts With `sysadm`."

5.3 Using the `sysadm` Menus

The following commands and conventions apply when you are using the `sysadm` login, `sysadm` command, or any `sysadm` submenu:

- You must be at the console terminal to log in as `sysadm`, and you must know the `sysadm` password.
- To select a menu option from the `sysadm` Main menu or a submenu, type the number associated with that option, the name of the option, or the initial part of the option name.
- If you do not understand the screen displayed by the system, type `?` for help. The system provides a one- or two-sentence explanation of the screen display.
- Use `^` to return to the previous menu (the previous submenu or the Main menu) from a submenu.
- Type `q` (**q**uit) to safely leave the `sysadm` environment at any time and from any level within a menu.

5.3.1 `sysadm` Menu Bypass

You may access a `sysadm` submenu directly by typing the `sysadm` command followed by the submenu name. For example, type:

```
# sysadm diskmgmt
```

to access the `sysadm` Disk Management submenu.

Each `sysadm` submenu contains a list of options. You may bypass the submenu and access a known option directly by typing the option name as an argument to `sysadm`. For example, the Disk Management menu contains an option called `format`. Access that option directly by typing:

```
# sysadm format
```

If you use `sysadm` as a login, you may also supply the `sysadm` options described above, such as `format`, on the command line.

6. MANAGING USER ACCOUNTS

6.1 Introduction

The INTERACTIVE UNIX Operating System requires that users log in to the system with a login name and, optionally, a password. This enables the system administrator to keep track of users on the system, control access to the system, and control system resources. All individual users should use a private login. (Other computers can use generic logins such as `mail` or `nuucp` to log in to your computer and exchange information.)

The system administrator is responsible for establishing and maintaining a login account for each user who requires access to the system. This includes adding users, aging user passwords (if desired), changing user passwords (if necessary), and removing user logins that are no longer needed. Use the `sysadm` command or login to establish new user accounts. `sysadm` provides a safe and secure way to manage new and existing user accounts.

6.2 What Is a User Login Account?

A user login account defines the user's environment, which can be very restricted or highly privileged. It is defined by a number of attributes (characteristics) that are established when a login ID is created, and it determines the security levels that are imposed upon the user, including the programs and files that are accessible.

At the simplest level, a user login consists of the following information stored in the `/etc/passwd` file, which defines a user's individual environment:

- *login ID*
  The name used to log in to the system. The user types this name at the `login:` prompt. This one to eight character string must be unique for each user; a first name, last name, initials, or some combination are typically used. It may only consist of lowercase letters and/or numbers. Uppercase letters and special symbols are not permitted.

- *password mark*
  A mark indicating whether or not the user has a password. An `x` is placed in this field if there is an encrypted password entry in `/etc/shadow` for this user. If the user does not have an encrypted password entry in `/etc/shadow`, this field will be

blank. The actual password is a string the user must know and type in when logging in to the system. It should be known only to the user.

- *user ID*
  A **u**ser **id**entification number (frequently abbreviated UID) is a unique number assigned to each login account. The system starts with a default number (for example, 100), then increments the number by one each time a new user is added to the system. This number is associated with all files and processes created by that user name. The user ID is usually unique for each user login.

- *group ID*
  A **g**roup **id**entification number (frequently abbreviated GID) is a number that associates a user with a group. There may be one or more groups on the system. A default group and group ID number are automatically assigned to a user. An affiliation with a group means that a user can access the files and directories that are owned by members of that group, if the group permissions on those files or directories permit it. The GID is associated with an entry in the file `/etc/group`.

- *user name*
  This is the full name of the user (e.g., Joe Smith, Jane Grey, etc.). It is included because user login IDs are sometimes cryptic.

- *home directory*
  The full path name of the directory into which the user is automatically placed after logging in. This directory belongs solely to the user, who can restrict others' access to the files and directories created there. (Only the superuser can access this directory if the user chooses to restrict all others.) The home directory must be unique for each user login.

- *login program name*
  This is the full path name of the program the system runs each time the user logs in. This is usually the shell (`/bin/sh`), but it may be another program, such as a restricted shell (`/bin/rsh`), the C shell (`/bin/csh`), or an application program, such as the VP/ix* Environment.

6.3 System Files That Define the User's Environment

6.3.1 The `/etc/passwd` File

The `/etc/passwd` file identifies each user to the system. Each time you add a user to the system, an entry is added to this file. When a user attempts to log in to the system, the system checks the login ID against the list of authorized users that is stored in the file `/etc/passwd`. The user's password is checked against the entry for that user in `/etc/shadow`. If a login ID or password is missing or incorrect, the user cannot gain access to the system. If the user provides a correct login ID and password, the user is logged in to the system with access to all of the resources assigned to that login ID. A password provides an additional level of security on the system and should be set for each user account. By default, the system forces users to establish one the first time they log in, if the system administrator has not already assigned one. (To change the default, refer to *login*(1).)

Use the `sysadm` command to establish each of these fields for a new user or to change the information in these fields for an existing user account. `/etc/passwd` is automatically modified by `sysadm` each time you add, delete, or change a login account. You should never directly edit the password files themselves. Refer to section 6.4 for more information about using the `sysadm` utility.

6.3.2 The `/etc/group` File

Just as `/etc/passwd` defines the individual user's environment, the `/etc/group` file defines the environment for each system group. This file is used to establish another level of security for each login ID. Each user on the system is assigned to one or more groups. Members of a group have access to files and directories that are owned by other members of that group. The default group identity for users of the INTERACTIVE UNIX Operating System is `other`. Additional group identities supplied with your system are `root`, `bin`, `sys`, `adm`, `mail`, `rje`, and `daemon`. You may create new groups based on your use of the system. The file `/etc/group` contains a one-line entry with the following information for each group:

The group name
: A group name is a word, number, or abbreviation that is up to eight characters in length. You typically use a

name that refers to the group. For example, you may select `mktg` as the group name for all members of the Marketing Department or `fin` for the people in your Finance Department. A group name may only consist of lowercase letters and/or numbers. Uppercase letters and special symbols are not permitted.

A password
Not used.

A group ID number
This is a numeric identification that is associated with the group name. This number is the same as the GID number found in `/etc/passwd`.

A list of users in the group
You may include a list of user login IDs that are associated with the named group. To include a list of users in this group, separated by commas, you must edit the file directly rather than using the `sysadm` procedure. This entry is optional.

`/etc/group` is automatically modified by `sysadm` each time a new group is added to or deleted from the system.

6.3.3 The `/etc/shadow` File

Another file, `/etc/shadow`, is used primarily to provide tighter security. It contains the user's actual encrypted password and password aging information. Every time you log in the system checks your password against your entry in `/etc/shadow`. A password provides an additional level of security on the system and should be set for each user account. Only users with access to the password for a login ID (and the superuser) can access that account. The password is never displayed on the screen. It does not appear when it is first established, when you use it to log in, or when it is changed with `sysadm` or the `passwd` command. This procedure helps to maintain a high degree of system security.

`/etc/shadow` is automatically modified by `sysadm` each time a new user is added to or deleted from the system.

6.4 Managing User Accounts With `sysadm`

Use `sysadm` to manage the facilities that govern user logins. The User Management menu provides many of the options you need to add, delete, or modify entries in `/etc/passwd` and `/etc/group`. If you are not an experienced system administrator, it is best to use `sysadm` to make changes; do not attempt to modify these files directly.

You are now familiar with the fields that are used to establish login and group IDs. The `sysadm` menus provide step-by-step instructions that make it clear how to modify one or more of these fields. From the `sysadm` User Management menu, you can add a group to the system, add a user to the system, delete a group from the system, delete a user from the system, list the groups on the system, list the users on the system, modify the defaults used by the `adduser` command, obtain a menu of commands to modify group attributes, and obtain a menu of commands to modify a user's login.

After going through the two most commonly used options presented in this section, you should have no trouble following the others on the screen.

6.4.1 Adding a New User

1. Log in as the superuser or use the `su` command to become `root`:

```
$ su root
Password:
#
```

2. Access the User Management menu of `sysadm` by typing:

```
# sysadm usermgmt
```

   Your screen will look similar to this:

```
                    USER MANAGEMENT

   1 addgroup      add a group to the system
   2 adduser       add a user to the system
   3 delgroup      delete a group from the system
   4 deluser       delete a user from the system
   5 lsgroup       list groups in the system
   6 lsuser        list users in the system
   7 modadduser    modify defaults used by adduser
   8 modgroup      menu of commands to modify group attributes
   9 moduser       menu of commands to modify a user's login

 Enter a number, a name, the initial part of a name, or
? or <number>? for HELP, ^ to GO BACK, q to QUIT:
```

Select option 2, `adduser`.

Alternatively, if you know the name of the `usermgmt` menu option that you want to use, you may go directly to that option by typing the appropriate option name. For example:

```
# sysadm adduser
```

3. The screen will look similar to this:

```
Anytime you want to quit, type "q".
If you are not sure how to answer any prompt, type "?"
for help.

If a default appears in the question, press <RETURN>
for the default.

Enter user's full name [?, q]:
```

Type the user's full name. For example, type `Robin Hood`. You may not omit this field.

4. The system asks you to enter a login ID:

```
Enter user's login id [?, q]:
```

Type an appropriate login ID, for example, `robin`.

5. Next, the system prompts for a user identification number:

```
Enter user's id number (default 101) [?, q]:
```

Press ENTER to accept the default UID. The system automatically increments the last assigned number by one each time you add a new user to the system.

6. You are asked to supply a group identification number or group name:

```
Enter group id or group name
(default 1) [?, q]:
```

Press ENTER to accept the default GID. If this user is affiliated with a group other than the default, enter the correct group name (or number, if you know it). The group must already be established.

7. Next, indicate the path name of the user's login directory:

```
Enter user's login (home) directory name
(default '/usr/robin')[?, q]:
```

Press ENTER to accept the default, or type in another path name.

8. The system gives you an opportunity to install the entry permanently (`i` for install), correct any errors (`e` for edit), delete the entry (`s` for skip), or return to the menu (`q` for quit):

```
Do you want to install, edit, or skip this entry [i, e, s, q]?
```

Select the appropriate option. When you have installed the login ID, the system displays this message:

```
Login installed.

Do you want to give the user a password [y, n]
```

9. Type `y` to establish a password. The system displays:

```
New password:
```

10. Type in a password that is a unique word between 6 and 12 characters long. It may include upper- and lowercase characters, numbers, and symbols. It must contain at least two alphabetic characters and one number or symbol. (Note that if you do not assign passwords, users will be forced to set one the first time they log in.)

    Retype the password when the system asks for verification. The password you select is not displayed on the screen. When it is installed on the system, it is automatically encrypted and cannot be read.

11. The system then asks:

```
Do you want to add another login? [y, n, q]
```

If you want to add another user account, type `y`. The procedure for adding a new user is repeated. Type `n` if you do not want to add any other accounts.

6.4.2 Adding a New Group

1. Access the `addgroup` option of the `sysadm` User Management menu. The screen will look similar to this:

```
Anytime you want to quit, type "q".
If you are not sure how to answer any prompt, type "?" for
help.

If a default appears in the question, press <RETURN> for the
default.

Enter group name [?, q]:
```

   Type the name of the group. It should be a word, number, or abbreviation that is between three and eight characters. A group name may only consist of lowercase letters and/or numbers. Uppercase letters and special symbols are not permitted.

2. The system asks you to supply a group identification number:

```
Enter group ID number (default 101) [?, q]:
```

   Use ENTER to accept the default. Since the numbers from 0 to 100 are reserved, `sysadm` will not accept a number below 100.

3. The system displays the group name and number and gives you the opportunity to install, edit, or skip the entry:

```
This is the information for the new group:
      Group name:  staff
      group ID:    100
Do you want to install, edit, or skip this entry [i, e, s, q]:
```

   Type `i` to install the entry.

6.5 Managing Passwords

To log in to the system, both the login name and password must be used. Since the login names are known to other users on the system, the password is very important to system security. Observe the following guidelines when you use or assign passwords:

- A password should be hard to guess. Do not use names, birthdays, telephone numbers, or words that are of personal significance. For example, if you are known to be an enthusiatic sports fan, it is not a good idea to use the name of your favorite team.

- Passwords must be at least six characters long with at least two alphabetic characters and one non-alphabetic character.
- Never record your password or someone else's on paper.
- Make sure all system users understand the importance of keeping passwords secret.
- Encourage your users to change their passwords occasionally.

6.5.1 Aging User Passwords

Password aging allows the system administrator to set time requirements on passwords to make it harder for others to break into your system. After a specified period of time, passwords expire, and users are forced to enter new ones. Users are also prevented from changing a new password for a specified time.

When a login is first assigned, there is no aging. Password aging must be assigned by the system administrator (as the superuser) using the `passwd` command.

Password aging information consists of the following variables:

- `MINWEEKS` = *number*
  where *number* defines how soon a user can change a password after it was last changed. `MINWEEKS` is defined for all users on the system in `/etc/default/passwd`. The system administrator can also change this variable for each individual user account using the `passwd` command. The default is `0`.
- `MAXWEEKS` = *number*
  number defines the maximum amount of time a user can retain a password for a user account. `MAXWEEKS` is defined in `/etc/default/passwd`. The system administrator can also change this variable for each user account using the `passwd` command. The default is `1000`.
- `IDLEWEEKS` = *number*
  number defines the length of time a user's password account is allowed to remain idle without being changed. `IDLEWEEKS` may be defined in `/etc/default/login`; by default, it is not defined.

The general form of the `passwd` command used to set password aging is:

```
# passwd -l -xmax -nmin login_name
```

The `-x` option specifies the maximum number of days (*max*) that a password is valid for the user named *login_name*. If *max* is `0`, password aging is turned off for that login.

The `-n` option specifies the minimum number of days (*min*) that a password is valid for the *name* login. This means that *login_name* is forced to use the same password for at least *min* days.

For example, to force user `robert` to change his password every 30 days and to keep any assigned password for at least a week, type:

```
# passwd -l -x30 -n7 robert
```

See *passwd*(1) for a complete description of the `passwd` command.

6.5.2 Changing Other Users' Passwords

Ordinary users can use the `passwd` command to change their passwords any time (depending on how password aging is set). As superuser, you can change another user's password without knowing his or her password. This is useful when someone forgets a password for some reason.

To change a password, type:

```
# passwd login_name
```

7. SETTING UP YOUR PRINTER

A printer is added to the system with the `sysadm lpmgmt` option. Printers can be either serial or parallel. A serial printer must be plugged into a serial port, such as `COM1` or `COM2`; a parallel printer must be plugged into a parallel port, such as `LPT1` or `LPT2`.

The system supports a set of default printer types. However, the printers shown in this section are only examples. Your system may be configured to support additional or different printers. Before you attempt to install a new printer, refer to the documentation provided by your vendor or to your hardware manufacturer's instructions.

You can also refer to section 7.7, "Line Printer Driver," in the "INTERACTIVE UNIX Operating System Maintenance Procedures" for more information about the line printer driver. Refer to section 8 of that same document for a detailed description of the LP Print Services software, which allows you to tailor your printing environment, for example, to print out specific forms or to support specific features of a nonstandard printer.

7.1 Adding a New Printer (`lpsetup`)

To add a new printer to your system, follow this procedure:

1. Plug the printer into the appropriate serial or parallel port on your system. It may be located on your main board or on an adapter board. Refer to the instructions provided by your hardware manufacturer to determine the correct connection.
2. Access the `sysadm` Package Management menu. The system displays a screen similar to this:

```
                          PACKAGE MANAGEMENT

  1 lpmgmt          print spooler management menu
  2 uucpmgmt        basic networking utilities menu

Enter a number, a name, the initial part of a name, or
? or <number>? for HELP,  ^ to GO BACK,  q to QUIT:
```

3. There may be more options on this screen, depending on the software you have installed. Choose option `1, lpmgmt`. The

system will then display the Print Spooler Utilities Management menu:

```
                PRINT SPOOLER UTILITIES MANAGEMENT

  1 lpaccept      have a line printer accept print jobs
  2 lpreject      have a line printer reject print jobs
  3 lpremove      remove a line printer from the system
  4 lpsetup       add line printer
  5 lpstart       start line printer scheduler
  6 lpstop        stop line printer scheduler

Enter a number, a name, the initial part of a name, or
? or <number>? for HELP,  q to QUIT:
```

4. Select option 4, `lpsetup`. The system then displays a list of supported printers. The screen will look similar to this:

```
Available printer types are:

   5310      for AT&T 5310/5320 Dot Matrix Printer,
   lqp40     for LQP-40 Letter Quality Printer,
   dqp10     for DQP-10 Matrix Printer,
   hp        for Hewlett Packard 2631A line printer at 2400
             baud,
   prx       for Printronix P300 at 4800 baud using XON/XOFF
               protocol on a serial interface,
   1640      for Diablo 1640 at 1200 baud using XON/XOFF
               protocol,
   450       for DASI 450 using XON/XOFF protocol,
   mtx80     for a 80 column, parallel printer,
   hplaser   for a Hewlett Packard Laser printer,
   dumb_dos  for a line printer without special functions or
               protocol for DOS,
   dumb      for a line printer without special functions or
               protocol,
   serial    for a serial line printer without special
               functions or protocol,
   done      if no more printers are to be configured.

Enter type of printer to be added to lp system:
```

5. Enter the correct printer type. If you are adding a standard PC-type 80-column printer, type `dumb_dos`. If you are adding a standard 132-column printer attached to a parallel port, type `dumb`. If you are adding a standard serial printer, type `serial`. If you are adding a printer that has special characteristics but is not on the list, refer to section 8, "LP PRINT SERVICE ADMINISTRATION," in the "INTERACTIVE UNIX Operating System Maintenance Procedures" for information about how to add an unsupported printer to the system. If you use the `dumb` printer type for an

unsupported printer that has special characteristics, the special features will be unavailable to you.

```
The printer name is dumb_1.

If dumb_1 is OK, hit RETURN or
if you would prefer another name, enter the name:
```

In this example the system provides a default printer name, `dumb_1`, and asks you to verify it. You may want to enter a more descriptive name, for example, `laser`, if you are using a laser printer.

6. Press ENTER to accept the default printer name, or enter an appropriate printer name, then press ENTER.

7. The system lists the available device names for printers. The screen will look similar to this:

```
The choices for serial printer device name are:

/dev/tty00  /dev/tty01

The choices for parallel printer device name are:

/dev/lp0  /dev/lp1  /dev/lp2

Enter the device name:
```

8. The following tables give examples of the device name to use, depending on whether you are adding a serial or parallel printer and which port you plugged your printer into.

Serial Printer	
Port	*Device Name*
COM1	/dev/tty00
COM2	/dev/tty01

Parallel Printer	
Port	*Device Name*
LPT1	/dev/lp0
LPT2	/dev/lp1
LPT3	/dev/lp2

Consult your hardware manufacturer's information to determine which port you should use.

The default system is configured to support a parallel printer on port `LPT2` only (device `/dev/lp1`), so if you want to use a different or additional parallel printer port you must reconfigure your kernel. Refer to section 7.7, "Line Printer Driver," in the "INTERACTIVE UNIX Operating System Maintenance Procedures" for more information.

9. The system enables the printer and asks if you want to add another printer:

```
destination "dumb_1" now accepting requests
printer "dumb_1" now enabled

Available printer types are:

  5310      for AT&T 5310/5320 Dot Matrix Printer,
  lqp40     for LQP-40 Letter Quality Printer,
  dqp10     for DQP-10 Matrix Printer,
  hp        for Hewlett Packard 2631A line printer at 2400
            baud,
  prx       for Printronix P300 at 4800 baud using XON/XOFF
              protocol on a serial interface,
  1640      for Diablo 1640 at 1200 baud using XON/XOFF
              protocol,
  450       for DASI 450 using XON/XOFF protocol,
  mtx80     for a 80 column, parallel printer,
  hplaser   for a Hewlett Packard Laser printer,
  dumb_dos  for a line printer without special functions or
              protocol for DOS,
  dumb      for a line printer without special functions or
              protocol,
  serial    for a serial line printer without special
              functions or protocol,
  done      if no more printers are to be configured.

Enter type of printer to be added to lp system:
```

10. Type `done`, if you are adding only one printer.

    The system prompts you for the name of the default printer that is to be used by the print *spooler* (the program that is used to send jobs to the printer):

```
Select the name of the printer to be the default
destination from the following:
dumb_1
(or type <return> for none)
```

11. Enter the name of the designated printer. The system verifies the printer's status:

```
Current lp system status:
scheduler is running
system default destination: dumb_1
device for dumb_1: /dev/lp0
dumb_1 accepting requests since Aug  4 13:58
printer dumb_1 is idle.  enabled since Aug  4 13:58

Press the RETURN key to see the packagemgmt menu [?, ^, q]:
```

Return to the Package Management menu, then exit `sysadm`.

12. Test your new printer by sending a file to it using the `lp` command.

7.2 Instructing a Line Printer to Accept Print Jobs (`lpaccept`)

When you first use the `lpsetup` option to add a printer to your system, the printer will automatically accept print jobs. If you have previously stopped a printer on your system and now want it to accept print jobs again, use the `lpaccept` option.

1. Access the `sysadm` Print Spooler Utilities Management menu. The system displays a screen similar to this:

```
                    PRINT SPOOLER UTILITIES MANAGEMENT

  1 lpaccept        have a line printer accept print jobs
  2 lpreject        have a line printer reject print jobs
  3 lpremove        remove a line printer from the system
  4 lpsetup         add line printer
  5 lpstart         start line printer scheduler
  6 lpstop          stop line printer scheduler

Enter a number, a name, the initial part of a name, or
? or <number>? for HELP,  q to QUIT:
```

2. Select option `1`, `lpaccept`. Your screen will look similar to this:

```
Available printer(s):
mtx80_1
dumb_1
Enter the printer(s) you want to accept jobs[?, q]:
```

The system lists the names of the active printers on your system.

3. Type in the name of the printer (or printers, separated by a space) you want to accept jobs, for example `dumb_1`. Your screen will look similar to this:

```
destination "dumb_1" now accepting requests
```

The printer known as `dumb_1` will now accept print jobs. The system returns you to the system prompt.

7.3 Instructing a Line Printer to Reject Print Jobs (`lpreject`)

If you want a printer on your system to stop accepting print jobs, for example, while you perform routine maintenance, follow this procedure:

1. Access the `sysadm` Print Spooler Utilities Management menu. Your screen will look similar to this:

```
                PRINT SPOOLER UTILITIES MANAGEMENT

  1 lpaccept      have a line printer accept print jobs
  2 lpreject      have a line printer reject print jobs
  3 lpremove      remove a line printer from the system
  4 lpsetup       add line printer
  5 lpstart       start line printer scheduler
  6 lpstop        stop line printer scheduler

Enter a number, a name, the initial part of a name, or
? or <number>? for HELP,  q to QUIT:
```

2. Select option 2, `lpreject`. Your screen will look similar to this:

```
Available printer(s):
mtx80_1
dumb_1
Enter the printer(s) you want to reject jobs[?, q]:
```

3. Type in the names of the printers you want to reject jobs, separated by spaces. Your screen will look similar to this:

```
destination "dumb_1" will no longer accept requests

Press the RETURN key to see the lpmgmt menu [?, q]:
```

7.4 Removing a Line Printer From the System (`lpremove`)

If you want to remove a printer from your system, follow this procedure:

1. Access the `sysadm` Print Spooler Utilities Management menu. Your screen will look similar to this:

```
                    PRINT SPOOLER UTILITIES MANAGEMENT

  1 lpaccept        have a line printer accept print jobs
  2 lpreject        have a line printer reject print jobs
  3 lpremove        remove a line printer from the system
  4 lpsetup         add line printer
  5 lpstart         start line printer scheduler
  6 lpstop          stop line printer scheduler

Enter a number, a name, the initial part of a name, or
? or <number>? for HELP,  q to QUIT:
```

2. Select option 3, `lpremove`. Your screen will look similar to this:

```
Available printer(s):
mtx80_1
dumb_1

Enter the printer(s) you want to remove[?, q]:
```

3. Type in the name of the printer or printers you want to remove separated by spaces. The system will sequentially ask for verification for each printer.

```
Okay to remove printer dumb_1 [y, n, q]
```

4. Type `y` to remove the `dumb_1` printer from your system.

Note that if you remove the default printer, `dumb_1`, from your system, you then need to establish a new default printer. To do this, follow this procedure:

1. Return to the `sysadm` Print Spooler Utilities Management menu. Your screen will look similar to this:

```
                    PRINT SPOOLER UTILITIES MANAGEMENT

  1 lpaccept        have a line printer accept print jobs
  2 lpreject        have a line printer reject print jobs
  3 lpremove        remove a line printer from the system
  4 lpsetup         add line printer
  5 lpstart         start line printer scheduler
  6 lpstop          stop line printer scheduler

Enter a number, a name, the initial part of a name, or
? or <number>? for HELP,  q to QUIT:
```

2. Select option 4, `lpsetup`. Your screen will look similar to this:

```
Available printer types are:

  5310      for AT&T 5310/5320 Dot Matrix Printer,
  lqp40     for LQP-40 Letter Quality Printer,
  dqp10     for DQP-10 Matrix Printer,
  hp        for Hewlett Packard 2631A line printer at 2400
            baud,
  prx       for Printronix P300 at 4800 baud using XON/XOFF
              protocol on a serial interface,
  1640      for Diablo 1640 at 1200 baud using XON/XOFF
              protocol,
  450       for DASI 450 using XON/XOFF protocol,
  mtx80     for a 80 column, parallel printer,
  hplaser   for a Hewlett Packard Laser printer,
  dumb_dos  for a line printer without special functions or
              protocol for DOS,
  dumb      for a line printer without special functions or
              protocol,
  serial    for a serial line printer without special
              functions or protocol,
  done      if no more printers are to be configured.

Enter type of printer to be added to lp system:
```

3. Type `done`. The system then lists the printers still available on your system and prompts you for the name of the default printer that is to be used by the print spooler program.

```
Select the name of the printer to be the default
destination from the following:
mtx80_1
(or type <return> for none)
```

4. Type in the name of your choice. The system verifies the printer's status:

```
Current lp system status:
scheduler is running
system default destination: mtx80_1
device for mtx80_1: /dev/lp0
mtx80_1 accepting requests since Aug  4 13:58
printer mtx80_1 is idle.  enabled since Aug  4 13:58

Press the RETURN key to see the packagemgmt menu [?, ^, q]:
```

7.5 Starting the Line Printer Scheduler (`lpstart`)

The line printer scheduler is the daemon that controls the jobs that are queued to all your line printers. If you have stopped the line printer scheduler and want to restart it, follow this procedure:

1. Access the `sysadm` Print Spooler Utilities Management menu. Your screen will look similar to this:

```
                    PRINT SPOOLER UTILITIES MANAGEMENT

  1 lpaccept        have a line printer accept print jobs
  2 lpreject        have a line printer reject print jobs
  3 lpremove        remove a line printer from the system
  4 lpsetup         add line printer
  5 lpstart         start line printer scheduler
  6 lpstop          stop line printer scheduler

Enter a number, a name, the initial part of a name, or
? or <number>? for HELP,  q to QUIT:
```

2. Select option `5, lpstart`. Your screen will look similar to this:

```
Print services started.
```

   The line printer scheduler is now running.

7.6 Stopping the Line Printer Scheduler (`lpstop`)

If you want to stop the line printer scheduler, for example, to service a printer, follow this procedure:

1. Access the `sysadm` Print Spooler Utilities Management menu. Your screen will look similar to this:

```
                    PRINT SPOOLER UTILITIES MANAGEMENT

  1 lpaccept        have a line printer accept print jobs
  2 lpreject        have a line printer reject print jobs
  3 lpremove        remove a line printer from the system
  4 lpsetup         add line printer
  5 lpstart         start line printer scheduler
  6 lpstop          stop line printer scheduler

Enter a number, a name, the initial part of a name, or
? or <number>? for HELP,  q to QUIT:
```

2. Select option `5, lpstop`. Your screen will look similar to this:

```
Print services stopped.
```

   The line printer scheduler is no longer running. You will have to restart it when you are ready.

8. TAILORING THE ENVIRONMENT

When you first installed your system, you tailored several system parameters, such as the time zone and date. As you become more familiar with the INTERACTIVE UNIX Operating System, you may wish to customize your system to better suit your specific working environment.

An INTERACTIVE UNIX System is usually customized by modifying one or more *configuration files*. These are files that store information that affects the environment of an individual user or affects the system on a global basis. There are several configuration files on the system that may be modified to tailor the environment to meet system-wide or individual requirements.

Every INTERACTIVE UNIX System user operates in a unique environment that is controlled by the shell, the INTERACTIVE UNIX System command interpreter. A number of shell *variables* are defined at the time the user logs in to control many of the default actions of the user's login session. A variable is a word that is assigned a value and has a special meaning to the INTERACTIVE UNIX Operating System. For example, the default terminal type is a shell variable that is usually set for each user. Additional shell variables determine how commands issued by a user are run.

Many INTERACTIVE UNIX System configuration files are shell command scripts or contain shell variable assignments. To fully understand these concepts, refer to the *User's Guide* for more detailed information about the definition and use of shell command scripts.

Each file has a default configuration that can often be tailored through the use of comments. A comment is a part of a file that is not processed by the program that reads the file. In a shell script, a comment is preceded by a #. In a C program (C is the name of the programming language used to write UNIX programs), a comment is bracketed by /* and */. In an INTERACTIVE UNIX System text processing file, a comment is preceded by \" or '\", if it occurs at the beginning of a line. Comments are ignored by the system. Thus, you may activate or inactivate commands in system configuration files by adding or removing the comment character(s). Many configuration files on the INTERACTIVE UNIX System use comment lines to explain the function of a command sequence.

8.1 Environment Variables

A number of shell or *environment variables* are set automatically by the system for all users. An environment variable is assigned the value of a *string* (a word, file name, command, or other sequence of characters or letters). A system administrator may want to change some of these variables in the system's default profile (`/etc/profile`), or an individual user may want to change them in his or her own `.profile`.

The syntax for assigning a value to a variable is:

variablename=value

(Note that there are no spaces.) For example, the `HOME` variable is set equal to the path name of the user's home directory:

```
HOME=/usr/deb
```

The environment variables that are usually defined by default are listed alphabetically below.

`CDPATH`
: Defines the paths to be searched for an argument to the `cd` command. By default, the current directory is searched.

`HOME`
: Defines the path name of your login directory. The value of `HOME` is set when you log in and should not be changed. `HOME` is specified in `/etc/passwd`.

`HZ`
: Defines the number of ticks per second for the system clock.

`IFS`
: Defines the internal field separator characters. The shell initially sets these characters to include the space (blank), tab, and newline characters.

`LOGNAME`
: Defines your login name and is set when you log in to the system. This variable is often referred to by shell programs. `LOGNAME` is specified in `/etc/passwd`.

`MAIL`
: Defines the full path name of the directory where you will receive mail from other users. `MAIL` is usually kept in `/usr/mail` (i.e., `/usr/mail/`*LOGNAME*).

`MAILCHECK`
Defines the interval at which the system checks for new mail (in seconds).

`PATH`
Defines the directory search path for commands. By default, this variable includes the current directory, the `/bin` directory, and the `/usr/bin` directory.

`PS1`
Defines the primary shell prompt. By default, the shell prompt is set to `$` for regular INTERACTIVE UNIX System users and `#` for users who log in as `root`.

`PS2`
Defines the secondary shell prompt. By default, the secondary shell prompt is set to `>`. When this prompt appears, it means that additional information (input) is needed for the command to run.

`TERM`
Defines your terminal type for certain programs (such as screen editors). By default, `TERM` is set to `AT386`.

`TZ`
Defines the time zone in which the computer is located.

8.2 The System Default Profile

When you first log in to the INTERACTIVE UNIX System, your working environment is defined by the default system `profile` located in `/etc/profile`. This file is a shell script that is executed each time the Bourne shell is started. It contains the commands needed to initialize your environment and the commands common to all users.

`/etc/profile` executes a sequence of commands and sets a number of shell (environment) variables. `/etc/profile` also runs a number of commands. It displays the message of the day and runs `umask`, the command that determines the default protections for new files and directories created by users. (Refer to section 14.4 for more information about the `umask` command.) In addition, `/etc/profile` may be tailored to run other programs that should be run when a user first logs in, for example, the `news` program. Here is a sample `/etc/profile` file:

```
#       Copyright (c) 1984 AT&T
#         All Rights Reserved

#       THIS IS UNPUBLISHED PROPRIETARY SOURCE CODE OF AT&T
#       The copyright notice above does not evidence any
#       actual or intended publication of such source code.

#ident  "@(#)profile    1.3"
#       The profile that all logins get before using their own
#       .profile.

trap "" 1 2 3
umask 022       # set default file creation mask
. /etc/TIMEZONE

case "$0" in
-sh | -rsh)
# calculate available disk space in root file system.
#       . /etc/dfspace

# issue message of the day
        trap : 1 2 3
        if [ -s /etc/motd ] ; then cat /etc/motd; fi

        trap "" 1 2 3
        stty erase '^h' echo
        export TERM;    TERM=AT386      # default terminal type

# check mailbox and news bulletins
        if mail -e
        then echo "you have mail"
        fi
        if [ $LOGNAME != root ]
        then news -n
        fi
        ;;
-su)
        :
        ;;
esac
export PATH;
trap 1 2 3
```

You must be logged in as `root` to modify the default system profile. Use an editor supplied with your system to modify this file.

8.3 The Individual's `.profile`

Individual users can further customize their environment by creating a personal version of `/etc/profile`, named `.profile`, that is stored in the user's login directory. `.profile` may be used to reset any of the environment variables listed above to suit the individual's requirements better than the system defaults. The system reads `.profile` *after* it reads `/etc/profile`, so the variables set in `.profile` "override" the system defaults. Here is an example of a typical `.profile`:

```
PATH=:/bin:/usr/bin:$HOME/bin
SHELL=/bin/sh
MAIL=/usr/mail/deb
TERM=wyse60
export PATH SHELL MAIL TERM
```

The export statement is used to make the variables apply to every environment the user enters.

After creating or making changes to the `.profile`, you can initiate the changes by logging off and logging back in again or you can type:

```
. .profile
```

and press ENTER. The shell will reinitialize your environment. The dot (.) is a special shell command used to execute commands in command scripts such as `.profile`.

To look at the variables set in your environment, you can use the `set` or `env` commands. Type:

```
env
```

or

```
set
```

and press ENTER.

You can set an environment variable temporarily, so that it applies only to the current login session, by typing it at the system prompt. For example, if your `TERM` variable is set to `AT386` because you usually use the system console, but you want to temporarily log in on a DEC* VT100* terminal, type:

```
$ TERM=vt100 export TERM
```

and press ENTER. For that login session, your terminal type is `vt100`. The next time you log in, your `TERM` variable will be `AT386`, as usual.

8.4 Setting System Date and Time

System time under the INTERACTIVE UNIX Operating System is controlled and affected by several different factors. Each of the following items plays a role in the date and time that users and programs observe:

- System clock
- CMOS RAM
- User environment
- Daylight Saving Time
- `setlocal` program

Your computer has an electronic quartz clock powered by a small battery. This clock runs all the time, even when the computer is off. The computer operator can set the date and time that the clock maintains in one of three ways:

- The hardware manufacturer's *setup* program
- The ROM BIOS setup program, accessed by a "hot-key" before booting the system
- An operating system utility

The actual time displayed by the clock depends on the settings stored in CMOS RAM (CoMplementary Oxide Silicon). This type of memory requires very little energy to maintain. The CMOS RAM memory is powered by the clock battery, even when the system is off. In addition to the date and time, system configuration information is also stored here. Usually, the manufacturer's *setup* utility or *setup* program is used to change the system date and time.

If you use the INTERACTIVE UNIX System `sysadm` utility (using the `datetime` option on the System Setup Menu) or the `date` command to change the date or time, the change will be made permanently, updating both the time stored in the CMOS RAM and the UNIX System time.

When the UNIX System kernel is booted, it reads the system date and time from the CMOS RAM. The UNIX System determines which time zone it is running in from the `TZ` environment variable (set by `/bin/sh` as part of the system startup process). The `TZ` environment variable contains the main local time zone, the difference between the main local time zone and Greenwich Mean Time (GMT), and a possible alternate time zone, e.g., for areas that use Daylight Saving Time. The information in `TZ` is used in the calculation to convert the local time from the CMOS RAM to GMT, which is the time used internally by the UNIX System kernel. When the system displays dates and times, it converts GMT to the local time, again using the `TZ` environment variable to adjust for

the local time zone. The default time zone used initially by the system is Eastern Standard Time.

The user's environment affects programs that use or display the date and time, for example the `date` program. When the user logs in, the `login` program sets the `TZ` environment variable from the `TIMEZONE` value in the file `/etc/default/login`. For Bourne shell users (this is the default user environment), the `TZ` environment variable will be overridden by the value in `/etc/TIMEZONE`, which is executed via `/etc/profile` when `/bin/sh` is invoked by `login`. For C shell users, `/etc/TIMEZONE` will not be executed, so it is important that both `/etc/default/login` and `/etc/TIMEZONE` are changed from the default Eastern Standard Time to contain the correct time zone for the system.

In areas that use Daylight Saving Time, local time is moved forward or backward at different times of the year. If the system is running during a change to or from Daylight Saving Time, the programs that use or display dates or times will correctly reflect the new date and time as long as the `TZ` environment variable appropriately specified the alternate time zone. However, the CMOS RAM time does not automatically change, so the next time the system is booted, the kernel will reinitialize the date and time based on the CMOS RAM, and it will thus revert to the old time. The date and time in the CMOS RAM must be updated manually using one of the methods specified above.

8.4.1 Using the `date` *Command*

Use the `date` command to change the system clock. Although several options of this command can be used by ordinary users, only the superuser can use the command to change the system date and time. To set the system date and time:

1. Log in as `root`.
2. Type in the correct month, day, hour, minute, and year using the following format:

   `date` *MMddhhmmyy*

   `MM` = month (01 for January, 12 for December)

   `dd` = day of month (01 for the first, 10 for the tenth)

`hh` = hour of day (24-hour clock: 00 for midnight, 23 for 11:00 P.M.)

`mm` = minute of hour

`yy` = year (90 for 1990)

For example, to set the date to February 15, 1989 and the time to 1:30 P.M., enter:

```
# date 0215133089
```

See *date*(1) for a complete description of the `date` command.

8.5 Setting Up a Message of the Day (`/etc/motd`)

Your system may be set up so that a message-of-the-day is displayed whenever a user logs in to the system. The text of the message is stored in the file `/etc/motd`. You can modify this file, using an editor delivered with your system, to add important messages that should be seen by all system users. It is a good place to put messages regarding scheduled system downtime or other daily reminders. Here is a sample `/etc/motd` file:

```
*****   The system will be down tonight    *****
*****   from 5:00 p.m. to 7:00 p.m. for    *****
*****    scheduled system maintenance.     *****

*****    The new Networking Package is     *****
***** installed.  Problems to operations.  *****
```

The message-of-the-day is displayed each time a user logs in to the system and whenever you use the `sysadm` command or login.

8.6 Changing the News

The message of the day should be used for important, but short announcements. If you have longer messages you want to put in a location accessible to all users, use the `/usr/news` files. Use one `news` file per news item. To add or change `news`, create a file in the `news` directory and type in the news you want system users to read. To read the news after logging in, each user types `news` and presses ENTER. See *news*(1) for more information about `news`.

8.7 Automatic Program Execution

8.7.1 The `cron` *Program*

The `cron` program runs other programs automatically at specified times. This program and, more specifically, the `crontab` command, allow you to run programs during off hours such as:

- File system administration
- Long running, user-written shell procedures
- Cleanup procedures

Any task that needs to be done repeatedly at a specific time can be put into the `cron` file, which is located in the `/usr/spool/cron/crontabs` directory. If authorized by the system administrator, users can use the `crontab` command to establish their entries.

The `crontab` command has several options, shown in the following table:

Command	*Result*
`crontab`	Copies the standard input from the screen into a directory that holds all users' `crontabs`.
`crontab` *file name*	Copies the named file into a directory that holds all users' `crontabs`.
`crontab -r`	Removes a user's `crontab` from the `crontab` directory.
`crontab -l`	Lists the `crontab` file for the invoking user.

See *crontab*(1) for additional information.

Each line in the `crontab` file defines one procedure. For example:

```
minute hour day month day-of-week command
```

Each field is defined as follows:

Field	*Definition*
minute	0-59
hour	0-23
day	1-31
month	1-12
day-of-week	0-6 (0=Sunday)
command	the command to be executed at the time specified

The following rules apply to the first five fields:

- Two numbers separated by a hyphen indicate a range of numbers between the two specified numbers.
- A list of numbers separated by commas indicates that only the numbers listed will be used.
- An asterisk specifies all legal values.

For example, `0 0 1,14 * 2` indicates a command will be run on the first and fourteenth of each month, as well as on every Tuesday. If a percent sign (`%`) is placed in the command field (sixth field), the INTERACTIVE UNIX System translates it as a new-line character. Only the first line of a command field (character string up to the percent sign) is executed by the shell. Any other lines are made available to the command as standard input.

For example, suppose a file called `anyfile` contains the following `cron` entry:

```
0 0 1 * * mailx $LOGNAME % Subject: Call Mom! % now
```

When the command line `crontab anyfile` is executed, the user whose login is `$LOGNAME` will get a reminder mail message with `Call Mom!` as the subject the first of every month.

8.7.2 Automatic System Cleanup

The INTERACTIVE UNIX System should be cleaned up occasionally. Fortunately, the `crontab` command and `crontab` file make this task easier. You can specify cleanup jobs (e.g., remove aged files) and the time you want them to execute in the `crontab` file.

Your system comes with some default cleanup procedures already defined. These cleanup procedures are done by the `root` login under the control of `crontab` each Sunday morning at 5:17 A.M. The file `/etc/cleanup` defines what cleanup procedures are done.

Some of the files cleaned up automatically each Sunday morning are:

`/etc/wtmp`
: This file contains a history of system logins. Every time a user logs in, a record is made in this file. If not cleaned, the size of this file will grow forever. Instead of deleting the contents of this file yourself, `cron` can do it for you.

- `/usr/adm/sulog`
  This file contains a history of users that use the `su` command to switch logins. As a security measure, this file should not be readable by other users. See *su*(1) for additional information.

Log in as `root` and execute `crontab -l` to see the `crontab` entry that executes `/etc/cleanup` as well as other cleanup routines for UUCP (Basic Networking). Examine `/etc/cleanup` to see the default routines run each Sunday morning. To change how cleanup jobs are performed, edit `/etc/cleanup` and modify the `root crontab` in `/usr/spool/cron/crontabs`.

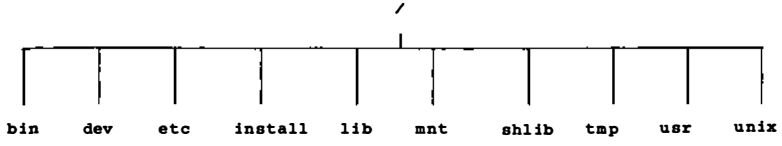
9. UNDERSTANDING INTERACTIVE UNIX SYSTEM FILE SYSTEMS

Many of the system administrator's responsibilities involve maintaining and monitoring file systems. This section explains how a file system is structured and the naming conventions that are used to access file systems and peripheral devices (usually hardware). You will use the `sysadm` File Management menu to perform most file system maintenance. However, before you can begin using the *utilities* available with the `sysadm` command, you must understand the definition and the structure of an INTERACTIVE UNIX Operating System file system.

9.1 What Is an INTERACTIVE UNIX System File System?

An INTERACTIVE UNIX System *file system* is a complete directory structure that is contained on a diskette or a fixed disk. It contains a collection of files arranged in a hierarchical order. On a diskette, a file system usually takes up the entire volume. On a fixed disk, a file system is associated with a single subpartition of the INTERACTIVE UNIX System partition on the fixed disk.

There is at least one permanent file system established on the first subpartition of the INTERACTIVE UNIX System partition on the first fixed disk. This permanent file system is known as the *root file system*. The main (top level) directory in the `root` file system is called the *root directory*. It contains all of the important files and subdirectories that are required to run the INTERACTIVE UNIX Operating System. The `root` directory is automatically created and given the unique name of / (slash) when your system is installed and initialized. A typical `root` file system looks like this:

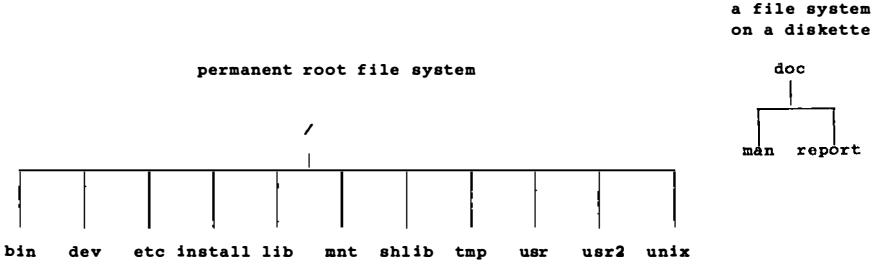


In this example, the file `unix` in the `root` directory contains the kernel. The remaining entries are files and subdirectories that

contain text files, command files, shell scripts, and other directories that are used by the operating system.

In addition to the permanent `root` file system, you may establish additional file systems on fixed disks or diskettes. Additional file systems can be *mounted* on the original `root` file system. Mounting a file system attaches it to the `root` directory and makes it accessible to users. If the file system is located on a portion of the fixed disk, it can be automatically mounted when the system is brought up. It is often useful to establish multiple file systems on the fixed disk to improve system performance or to prevent unauthorized access to sensitive data. Fixed disk file systems are created when you install your INTERACTIVE UNIX System and when you install additional fixed disks on your computer. If the file system is established on a diskette, it must be mounted before it can be accessed.

The figure below illustrates a typical INTERACTIVE UNIX System file system structure. The `root` file system contains one additional mounted file system, `usr2`. In addition, a user owns a diskette that contains a file system labeled `doc`. It has not been mounted. To be made accessible to users, the diskette must be physically present in the diskette drive and the file system on it must be mounted.

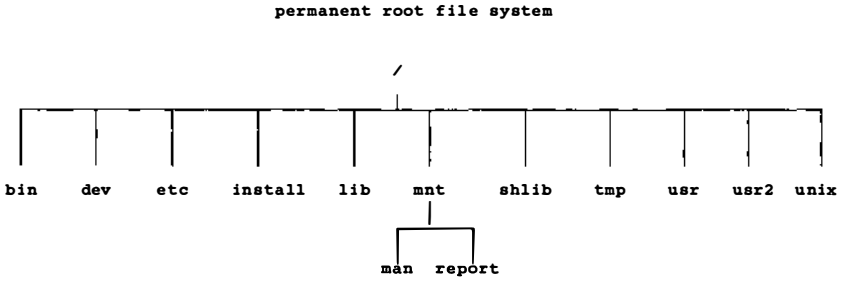


All files and directories in a file system are accessed using a path name. Path names have the following form:

/name1/name2/name3

where *name1* and *name2* are directory names and *name3* (the last name in the path name) is either a directory name or a file name.

The first slash (`/`) in the path name is the name of the `root` directory. The subsequent slashes in the path name are used to separate the directory names and file names. Once you have mounted a file system, it appears to be an ordinary directory that is part of the `root` file system and is accessible by standard INTERACTIVE UNIX System naming conventions. For example, when the `doc` file system described above is mounted under the `/mnt` directory, the file system will look like this:



To access the file called `report`, use the path name:

```
/mnt/report
```

The file system is still the one labeled `doc` on the diskette, but the label is not used while the file system is mounted on the `root` directory.

9.2 File System Naming Conventions

Before you can begin any routine maintenance of the file system, you must understand the basic conventions that are used to name file systems and devices. When you mount a file system, back up a file system, or check a file system for errors, you may need to identify the device on which the file system is located. This section outlines the naming conventions for file systems on the INTERACTIVE UNIX System. The next section outlines the naming conventions for the most common devices found on your system. For more detailed information, refer to section 7 of the "INTERACTIVE UNIX Operating System Maintenance Procedures."

A file system name or "label" may be up to six characters in length and may contain upper- or lowercase letters and/or numbers. You should not use any characters in a file system name that have a special meaning to the shell. For example, `*`, `?`, and `&` are used by the shell and should not be used in file system labels.

It is a good idea to give the file systems meaningful names. For example, if you are creating a new file system that will be used exclusively by the Documentation department, you might label the file system `doc`. Depending on the size of your fixed disk and the INTERACTIVE UNIX System partition, the INTERACTIVE UNIX System creates one or more file systems when you install it. If only one is created, it is named `root`; if two are created, they are named `root` and `usr`. If created, the third file system is named `usr2` and so on, through `usr5`.

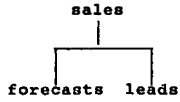
9.3 Mount Point Conventions

A file system is attached to the `root` directory through a process called mounting. Mounting a file system "tells" the system where the file system is located. The file system is mounted onto a directory in the `root` directory with a system administration procedure called `mount` (this is discussed in more detail in section 11).

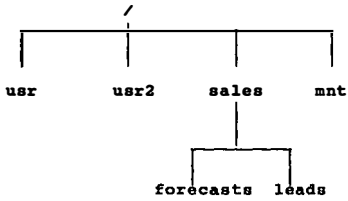
You must create a directory (using `mkdir`) to be used as the *mount point* for the file system. A mount point is a directory where the file system will be installed. Only one file system at a time can be installed on a mount point directory. The mount point directory may already contain other files and directories, but they will be inaccessible if a new file system is mounted on that directory. These existing files and directories are not deleted, but they remain "hidden" until the newly mounted file system is deactivated or "unmounted." You should warn the users on your system not to put any files into the mount point directory because the files will be unavailable to them while a file system is mounted there.

Both fixed disk and diskette file systems must be mounted to be used. By convention, diskettes are often mounted under a directory named `/mnt`. This is a convenient way to remember the file system name of all diskettes. It also eliminates the need to create extra directories under the `root` directory. By convention, directories created under the `root` directory for mounting fixed disk file systems are usually given the same name as the file system itself.

The diagram below illustrates how a directory serves as a place to mount a file system. The `root` file system is shown below on the left, and the unmounted `sales` file system on the fixed disk is shown on the right:



After you have created a directory under the `root` directory called `sales` and have mounted the `sales` file system there, it would look like this:



The complete path name to the directory where the `sales` file system is mounted is `/sales`.

You may create several file systems on a fixed disk during system installation or when adding a new fixed disk to your system. Refer to section 4 of the "INTERACTIVE UNIX Operating System Installation Instructions" for more information about creating file systems. Create a new directory under the `root` directory for each new file system. The directory name should be the same as the file system label. This makes it easier to remember the location of each new file system. For example, if you create a new file system labeled `doc`, create a new directory in the `root` directory called `doc`. The complete path name to the directory where the `doc` file system is mounted is `/doc`. You can also use this convention for diskette file systems, but it is more common to use only one directory, `/mnt`, for mounting file systems on diskettes. (The

illustrations in section 9.1 show the process of mounting a diskette file system on the `root` file system.)

9.4 Device Naming Conventions

The hardware devices attached to the INTERACTIVE UNIX System are named and accessed using ordinary path names. Device files are called *special files* because they do not contain data; they are references to the actual programs that run the peripheral devices attached to your computer. These programs are called *device drivers* and are used to control terminals, fixed disks, and diskette drives (among other things).

☛ Note that device names are arbitrary in most cases. The INTERACTIVE UNIX System is supplied with certain preestablished device names. These are adequate for most needs. System administrators may create additional special files or links between existing special files at their discretion. (Refer to *ln*(1) for more information about linking files.)

By convention, device files reside in a special directory called `/dev`. There are two basic types of device files: *character files* and *block files*. Character devices, such as terminals, generally process data one character at a time. Block devices, such as fixed disks, access hardware one block (sector or record) at a time. The INTERACTIVE UNIX System accesses a file system via a block device.

The device files for both fixed disks and diskettes reside in two parallel subdirectories, `rdsk` and `dsk`, under the directory `/dev`. `rdsk` stores character device files, and `dsk` stores block device files. Each device file has a unique name, which may appear in the `dsk` directory or the `rdsk` directory, or both. The device file name describes the physical attributes of the device it refers to. The naming conventions for diskettes and fixed disks are different; they are described in the following sections. As a system administrator, you need to know the names of the devices on your system because certain errors are reported in connection with specific devices. Device names for mountable file systems are always found in `/dev/dsk`, unless they are XENIX* file system names, which are located in `/dev`. (For a discussion of XENIX diskette device file names, see section 9.4.6.)

Note that some special files are also located in `/dev/SA` and `/dev/rSA`. These directories are used by the `sysadm` program to determine some device names.

9.4.1 `fdisk` Partitions and INTERACTIVE UNIX System File Systems

When the INTERACTIVE UNIX System is installed, its `fdisk` program is used to divide the fixed disk into *partitions*. You may have only one partition that takes up the entire fixed disk, or you may have several partitions, each containing its own operating system, such as UNIX or DOS. The partitions that divide up the entire fixed disk and are used to hold different operating systems are referred to in this document as `fdisk` partitions.

Even if you have multiple `fdisk` partitions, the INTERACTIVE UNIX System is installed on only one. This partition is itself subdivided into a number of sections that are sometimes referred to in technical documents as partitions. To avoid confusion, we refer to these subdivisions of the INTERACTIVE UNIX System partitions as subpartitions, since they are divisions of the `fdisk` partition on which the INTERACTIVE UNIX Operating System is installed. Both types of partitions are described in the following section.

9.4.2 Device Naming Conventions for Partitions on Fixed Disks

A fixed disk device name associated with an INTERACTIVE UNIX System partition has the following format:

`c`*controller_number*`d`*disk_number*`s`*partition_number*

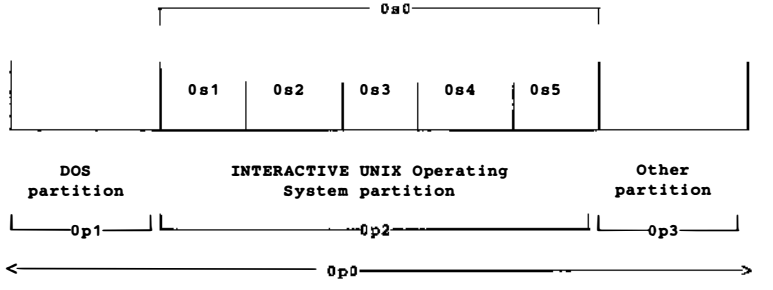
A fixed disk device name associated with an `fdisk` partition has the same format, except that the partition number is preceded by an `p` rather than a `s`.

`c`*controller_number*`d`*disk_number*`p`*partition_number*

The *controller*, *disk*, and *partition* numbers are 0-based (counting begins with 0 instead of 1). The *controller_number* refers to the single controller that most systems have. The controller is named `c0`. If you add a second controller, it is named `c1`.

The *disk_number* refers to your fixed disk. Your first fixed disk is named `d0`; if you have a second fixed disk, it is named `d1`. Each additional fixed disk adds one to the previous disk's number.

The *partition_number* refers to the `fdisk` partitions on your fixed disk, if preceded by a `p`, or to the "subpartitions" of your INTERACTIVE UNIX System partition if preceded by an `s`. You must always specify the partition number when naming a device. A typical disk might be partitioned as in the following illustration:



`s0` refers to the entire INTERACTIVE UNIX System partition and is generally used for maintenance purposes (for example, in repairing a damaged disk). The first actual subpartition is called `s1`. Each subsequent partition number (up to `s9`) is incremented by one until the last, which is called `sa`.

Certain INTERACTIVE UNIX System subpartitions are reserved for use by the system and cannot be accessed by user programs. These are: `s2` (the *swap area*, which is used to store portions of programs or data that have been temporarily swapped out of main memory), `s8` (a small partition that stores the bootstrap program), and `s9` (used to store alternate sectors that will be substituted for bad sectors in other partitions).

The remaining subpartitions (`s1`, `s3` through `s7`, and `sa`) are used to store program and data files. Of these subpartitions, only the first, `s1`, is required. This subpartition contains the `root` file system, where system files and programs are stored. This subpartition must be located on the first (primary) fixed disk.

If present, subpartition `s3` contains the `usr` file system, which is used to store some system files, and possibly users' files. Subpartitions `s4` through `s7`, if present, contain user-specified file systems, typically named `usr2` through `usr5`. These file systems are used to store users' files. Finally, partition `sa`, if present, contains the

`tmp` file system, which is used to store temporary files and data. (If separate `usr` and `tmp` file systems are not created, their function is performed by the `usr` and `tmp` directories in the `root` file system.)

If you have a small fixed disk, it may contain only one partition, with the `root` file system located on it. If you have a larger disk, or have two fixed disks, then these will contain additional partitions. Subject to size constraints, you may specify during installation how your fixed disk is to be partitioned. Refer to the "INTERACTIVE UNIX Operating System Installation Instructions" for more information about partitioning your fixed disk. For more information on the optimal division of file systems between different fixed disks, refer to section 4.8, "Adding a Second Fixed Disk," in the installation instructions.

The default organization of INTERACTIVE UNIX System subpartitions is described in the following table. The last two columns indicate, for both the first (primary) and additional disks, whether the specified partition is required or optional. Note that file system names must be unique, so that it is not possible, for example, to create a file system named `usr` on both the primary disk and the second disk.

Partition Name	*Contents*	*Primary Disk*	*Additional Disks*
p0	used for maintenance purposes only (refers to entire disk)	required	required
s0	used for maintenance purposes only (refers to entire INTERACTIVE UNIX System subpartition)	required	required
s1	contains the `root` file system	required	no
s2	contains the swap area	required	optional
s3	contains the `usr` file system	optional	optional
s4	contains user-specified file system	optional	optional
s5	contains user-specified file system	optional	optional
s6	contains user-specified file system	optional	optional
s7	contains user-specified file system	optional	optional
s8	contains the bootstrap area	required	required
s9	contains alternate sector space	required	required
sa	contains the `tmp` file system	optional	optional

These partitions are automatically created by the system when you install your disks.

9.4.3 Examples of Device File Names for Fixed Disks

The complete device file name for the second (first usable) partition of the first fixed disk on a system with one controller is `/dev/dsk/c0d0s1`. If you have only one controller, the name is abbreviated to `/dev/dsk/0s1` by shortening `c0d0` to `0`. In the INTERACTIVE UNIX Operating System, `c0d0` is always shortened to `0` for convenience.

The device file name for the third subpartition of the second fixed disk on a system with one controller is abbreviated to `/dev/dsk/1s3`.

If you have only one partition on a single fixed disk, the device file name is `/dev/dsk/0s1`.

The table below shows some common device file names for fixed disks that reside in the directory `/dev/dsk`. Your system may not require all of these special files.

Common Fixed Disk Device File Names

Controller Number	*Fixed Disk Number*	*Partition Number*	*Device Name*
1	1	1	`0s1`
1	1	2	`0s2`
1	2	1	`1s1`
1	2	4	`1s4`
2	1	1	`c1d0s1`
2	1	2	`c1d0s2`
2	2	1	`c1d1s1`

9.4.4 Device Naming Conventions for Diskettes

The naming conventions for diskette devices are as follows:

`f`*diskette_number*`[ d, q ]`*number_sectors*`[ d, dt ]`

The term *diskette_number* refers to your diskette drive number. `0` designates the first (or only) drive, sometimes referred to as the `A` drive. If you have two drives, the second drive is designated `1`. Do not type the brackets shown in the statement above. They indicate the options that may be available to you. Use `d` if you have a

double-sided, double density drive, and `q` if you have a high density (quad) diskette drive.

The term *number_sectors* refers to the number of sectors per track on your diskette. Use `8` for single density diskettes, `9` for most double density diskettes, `15` for high density 5¼ inch diskettes, and `18` for high density 3½ inch diskettes. Select `dt` if you plan to use the entire diskette, including the first cylinder (the boot block). Select `d` if you plan to begin reading and writing on the diskette with the second cylinder.

9.4.5 Examples of Device File Names for Diskettes

If you have a high density 5¼ inch diskette in the first drive and you plan to use the entire diskette, you will use the device file name `f0q15dt`. The complete path name to this device is `/dev/dsk/f0q15dt`.

If you have a high density 3½ inch diskette in the second drive and you plan to use the entire diskette, use the device file name `f1q18dt` to access the second drive. The complete path name to this device is `/dev/dsk/f1q18dt`.

The table below shows some standard diskette device file names.

Common Diskette Device File Names	
File Name	*Description*
`f0d8dt`	first drive, 5 ¼", single density diskettes, 320K
`f0d9dt`	first drive, 5 ¼", double density diskettes, 360K
`f0q15dt`	first drive, 5 ¼", high density diskettes, 1.2MB
`f0q9dt`	first drive, 3 ½", double density diskettes, 720K
`f0q18dt`	first drive, 3 ½", high density diskettes, 1.44MB
`f1d8dt`	second drive, 5 ¼", single density diskettes, 320K
`f1d9dt`	second drive, 5 ¼", double density diskettes, 360K
`f1q15dt`	second drive, 5 ¼", high density diskettes, 1.2MB
`f1q9dt`	second drive, 3 ½", double density diskettes, 720K
`f1q18dt`	second drive, 3 ½", high density diskettes, 1.44MB

There are also four generic diskette device files that automatically determine the density of the diskette. These support 360K and 1.2 MB 5 ¼ inch diskettes and 720K and 1.44 MB 3 ½ inch diskettes (the entire diskette). These files are:

`/dev/fd0`	block device, first drive
`/dev/fd1`	block device, second drive
`/dev/rfd0`	character device, first drive
`/dev/rfd1`	character device, second drive

Note that an unformatted 360K 5 ¼ inch diskette cannot be formatted in a 1.2 MB drive and an unformatted a 720K 3 ½ inch diskette cannot be formatted in a 1.44 MB drive using the generic device files.

9.4.6 XENIX Diskette Device Files

The INTERACTIVE UNIX Operating System also supports XENIX diskette device file names. The following table shows INTERACTIVE UNIX System device names and their XENIX equivalents.

INTERACTIVE UNIX Operating System and XENIX Diskette Device File Names	
INTERACTIVE UNIX System Names	*XENIX Names*
`/dev/dsk/f0d8dt`	`/dev/fd048ds8`
`/dev/dsk/f0d9dt`	`/dev/fd048ds9`
`/dev/dsk/f0q15dt`	`/dev/fd096ds15`
`/dev/rdsk/f0d8dt`	`/dev/rfd048ds8`
`/dev/rdsk/f0d9dt`	`/dev/rfd048ds9`
`/dev/rdsk/f0q15dt`	`/dev/rfd096ds15`

10. USING UTILITIES WITH THE DISKETTE DRIVE

Utilities are provided with the INTERACTIVE UNIX System to read and write data to and from diskettes. These utilities allow you to perform the following basic diskette operations:

- Format diskettes
- Copy files from the fixed disk to diskettes
- Copy files from diskettes to the fixed disk
- Duplicate diskettes

The INTERACTIVE UNIX System provides access to several types of diskettes. A diskette is accessed through either block or character (raw) device names. For information about INTERACTIVE UNIX System device names, refer to section 9.4.

10.1 Formatting Diskettes

Before a diskette can be used in any diskette operation, it must be formatted. You can use the `sysadm` program to format diskettes, or you can use the `format` command directly.

10.1.1 Formatting a Diskette Using `sysadm`

1. If you are using a new diskette, access the `sysadm` Disk Management Menu (`diskmgmt`) and select option `4`, `format`. The system asks:

   ```
   Do you want each format verified? (default: yes) [y,n,?,q]
   ```

   Press ENTER to accept the default.

2. The system displays:

   ```
   Select which drive to use:

    1 disk0_1.2M     3 disk0_360K    5 disk1_1.2M     7 disk1_360K
    2 disk0_1.44M    4 disk0_720K    6 disk1_1.44M    8 disk1_720K
    9 tape

   Enter a number, a name, the initial part of a name, or
   ? or <number>? for HELP,  q to QUIT:
   ```

   Select the option that corresponds to the type of diskette in the drive you want to use (remember that `disk0` is drive `A` and `disk1` is drive `B`). Check your manufacturer's documentation to determine this if you are unsure. Note that if you have a high density diskette drive, you can format both

low and high density diskettes. Option 1 selects a high density diskette in the first (5 ¼ inch) diskette drive; option 2 selects a high density diskette in the first (3 ½ inch) diskette drive. Option 3 selects a low density diskette in the first (5 ¼ inch) diskette drive; option 4 selects a low density diskette in the second (3 ½ inch) diskette drive. Option 5 selects a high density diskette in the second (5 ¼ inch) diskette drive; option 6 selects a high density diskette in the second (3 ½ inch) diskette drive. Option 7 selects a low density diskette in the second (5 ¼ inch) diskette drive; option 8 selects a low density diskette in the second (3 ½ inch) diskette drive. (Note that option 9 will only appear on your screen if you have a cartridge tape drive installed on your system.)

3. The system instructs you to insert the diskette in the drive, then formats your diskette:

```
Insert the medium in the diskette drive.
Press <RETURN> when ready [q].

Formatting in progress
formatting .........
Formatted 160 tracks; 0 through 159 interleave 1
The medium in the diskette drive is now formatted and
verified; it may be removed.
Insert another medium in the diskette drive.
Press <RETURN> when ready.  Type q to quit.
```

Your screen may show different values, depending on the type of diskette you are formatting. If you do not have any other diskettes to format, type `q` and press ENTER to return to the Disk Management menu.

10.1.2 Formatting a Diskette From the Command Line

1. Determine the diskette type.
2. Insert the diskette into the drive.
3. Enter the `format` command followed by the appropriate character device name for that type of diskette.

The following command will format a 1.2 MB double-sided, high density (DSHD) diskette:

```
$ format /dev/rdsk/f0q15dt
```

10.2 Copying Files to a Diskette

Several methods can be used to copy files to a diskette. One method is to format a diskette, create a file system on the disk, mount the file system, and access the diskette as part of the INTERACTIVE UNIX System directory structure. This might seem a bit complicated, but it is a very convenient way to create a portable file system.

Simpler methods involve using the `cpio` command with other commands, such as `find` or `ls`, or using the `tar` command.

10.2.1 Copying Files Using `cpio`

The `cpio` command copies files to and from diskettes by taking a list of files from standard input and creating a file archive that can be redirected to a diskette. Several commands can be used to supply the file names to be archived by using the INTERACTIVE UNIX System shell pipe facilities. For example, to copy all the files in the current directory to a 1.2 MB diskette:

1. Insert a formatted 1.2 MB diskette into the drive.
2. Change directory to the directory containing the files to be copied.
3. Type:

```
$ ls | cpio -oc > /dev/dsk/f0q15dt
```

In this example, `ls` lists the files in the current directory and pipes the list to `cpio`. Then `cpio` archives them and redirects the output to the diskette drive.

Similarly, any files copied onto a diskette using the `cpio` command can be copied back to the fixed disk using different `cpio` options. The following example copies the files that were copied to a diskette in the previous `cpio` example back to the fixed disk into the current directory.

1. Insert the diskette containing the `cpio` archive into the drive.

2. Type:

```
$ cpio -ic < /dev/dsk/f0q15dt
```

Refer to *cpio*(1) for a complete description of the `cpio` command.

10.2.2 Copying Files Using the `tar` Command

As an alternative to `cpio`, you can use the `tar` command to transfer files to and from diskettes. The following example copies files from the fixed disk onto a 1.2 MB diskette:

1. Insert a formatted 1.2 MB diskette into the drive.
2. Change directories to the directory containing the files to be copied.
3. Type:

```
# tar  -cf  /dev/dsk/f0q15dt
```

These files can then be copied back to the fixed disk using the `x` option to `tar`, as follows:

1. Insert the diskette containing the `tar` archive into the drive.
2. Change directories to the destination directory where you want to copy the files.
3. Type:

```
$ tar  -xf  /dev/dsk/f0q15dt
```

Refer to *tar*(1) for a complete description of the `tar` command.

11. FILE SYSTEM MAINTENANCE

When the INTERACTIVE UNIX Operating System is first installed and initialized, a `root (/)` file system is automatically created. Depending on the size of your fixed disk, you may also have chosen to create one or more user file systems, typically named `/usr` `/usr2`, `/usr3`, and so on. Refer to section 3, "CREATING AND USING INTERACTIVE UNIX SYSTEM FILE SYSTEMS," in the "INTERACTIVE UNIX Operating System Maintenance Procedures" for more information on manipulating file systems on fixed disks.

If you plan to store data on diskettes or to add a second fixed disk to your system, you will need to create and initialize additional file systems. You will use the `sysadm` Disk Management menu to facilitate file system management on your system. In this section you will learn how to:

- Create a file system
- Mount a file system
- Unmount a file system
- Check the amount of disk space available for a file system
- Check and repair a file system

11.1 Creating a File System on a Diskette

Before you can create a new file system on a diskette, you must format the diskette. To create a file system on a formatted diskette (a removable file system), use the `sysadm` command to access the Disk Management menu. Your file system will use the entire diskette.

☛ If you plan to use the diskette solely to store data using the `sysadm backup` or `store` utilities, you do not need to create a file system on it. Any files or file systems found on the diskette are automatically destroyed by `backup` or `store`.

To create a file system on a formatted diskette, follow these steps:

1. Access the Disk Management menu by typing `sysadm diskmgmt`. The system displays a screen similar to this:

```
                    DISK MANAGEMENT

  1 checkfsys     check a removable medium file system for errors
  2 cpdisk        make exact copies of a removable medium
  3 erase         erase data from removable medium
  4 format        format new removable diskettes
  5 harddisk      hard disk management menu
  6 makefsys      create a new file system on a removable medium
  7 mountfsys     mount a removable medium file system
  8 umountfsys    unmount a removable medium file system

Enter a number, a name, the initial part of a name, or
? or <number>? for HELP, q to QUIT:
```

2. Select option 6. The system instructs you to put the diskette in the drive and then displays a screen similar to this:

```
Select which drive to use:

 1 disk0_1.2M     3 disk0_360K    5 disk1_1.2M     7 disk1_360K
 2 disk0_1.44M    4 disk0_720K    6 disk1_1.44M    8 disk1_720K
 9 tape

Enter a number, a name, the initial part of a name, or
? or <number>? for HELP,  q to QUIT:
```

3. Select the option that corresponds to the type of diskette in the drive you want to use (remember that `disk0` is drive `A` and `disk1` is drive `B`). Check your manufacturer's documentation to determine this if you are unsure. Option 1 selects a high density diskette in the first (5 ¼ inch) diskette drive; option 2 selects a high density diskette in the first (3 ½ inch) diskette drive. Option 3 selects a low density diskette in the first (5 ¼ inch) diskette drive; option 4 selects a low density diskette in the second (3 ½ inch) diskette drive. Option 5 selects a high density diskette in the second (5 ¼ inch) diskette drive; option 6 selects a high density diskette in the second (3 ½ inch) diskette drive. Option 7 selects a low density diskette in the second (5 ¼ inch) diskette drive; option 8 selects a low density diskette in the second (3 ½ inch) diskette drive. The system then displays:

```
Insert medium in the disk0_1.2M drive.
Press <RETURN> when ready [q].
```

Note that the drive name that appears on your screen depends on the diskette type you just chose. Press ENTER after you have inserted the diskette and are ready to continue. The system then asks you to label the diskette:

```
Enter the label to be put on the medium [?,q]:
```

4. Type in a label (name) for the file system. This is a name up to six characters in length, which may contain upper- or lowercase letters and/or numbers. For example, you might use the name `doc`.

5. Next, the system asks you to enter the file system name:

   ```
   Enter the file system name [?, q]:
   ```

   This will be the name of the mount point directory created on the fixed disk. The system creates a directory with the name you specify as the mount point for the new file system. The conventions for the file system name are the same as those for the file system label. As discussed previously, by convention, the directory name used for mounting diskette-based file systems is usually `/mnt`.

6. Next, the system displays a screen similar to this:

   ```
   Enter the maximum number of files and directories on this
   medium (default 340) [q]:
   ```

   Press ENTER to accept the default.

   ☛ Always use the suggested default unless you are very familiar with the INTERACTIVE UNIX System or other UNIX-based operating systems.

7. As the system creates, initializes, and mounts your new file system, these messages are displayed on your screen:

   ```
   Building 'mnt' file system on 'mnt'.
   Initializing 'mnt' file system.
   Do you want to leave 'mnt' mounted? [y,n,?,q]
   ```

   Type `y` to keep the file system mounted. It is now possible to access the new file system with an ordinary path name as though it were part of the permanent `root` file system.

8. Exit the `sysadm` menu. To access the files in the file system you have just mounted (`/mnt`), log in to the system with your ordinary user login. Use `cd` to access the `/mnt` file system.

☛ Remember that you should always *unmount* the file system before removing the diskette from the drive. Refer to section 11.3.1 for information on how to do this.

11.2 Mounting a File System

There can be many file systems on an INTERACTIVE UNIX System. The file systems may be located on a number of devices, such as diskettes and fixed disk partitions. To use a file system, the INTERACTIVE UNIX System kernel must have information about the device where the file system is located and must know how to access the device. In addition, users must be able to access the file system with an ordinary INTERACTIVE UNIX System path name.

A file system cannot be used unless it is mounted on another file system, usually the `root` file system. File systems are mounted under the `root (/)` directory with the `mount` command or the `sysadm` Disk Management menu. Use the Disk Management menu to mount a diskette-based file system. Fixed disk file systems are usually mounted automatically each time you boot the system and are unmounted whenever you run the `shutdown` command. You may also use the `sysadm` option, `mounthdfsys`, located on the Hard Disk Management submenu, to mount a fixed disk file system.

It is strongly recommended that you use `sysadm checkfsys` and `sysadm checkhdfsys` to perform a file system check before attempting to mount file systems.

11.2.1 Mounting a Diskette-Based File System

`sysadm` automatically mounts a diskette-based file system when it is first created. Any time you use the diskette after that, you will need to explicitly mount the file system that is on it. Use the `sysadm` Disk Management menu to mount a diskette-based file system.

1. Access the Disk Management menu and select option 7, `mountfsys`. Alternatively, you may use the `mountfsys` administrative login account. Your screen will look similar to this:

```
Select which drive to use:

 1 disk0_1.2M     3 disk0_360K    5 disk1_1.2M     7 disk1_360K
 2 disk0_1.44M    4 disk0_720K    6 disk1_1.44M    8 disk1_720K
 9 tape

Enter a number, a name, the initial part of a name, or
? or <number>? for HELP,  q to QUIT:
```

2. Select the option that corresponds to the type of diskette in the drive you want to use. Option 1 selects a high density diskette in the first (5 ¼ inch) diskette drive; option 2 selects a high density diskette in the first (3 ½ inch) diskette drive. Option 3 selects a low density diskette in the first (5 ¼ inch) diskette drive; option 4 selects a low density diskette in the second (3 ½ inch) diskette drive. Option 5 selects a high density diskette in the second (5 ¼ inch) diskette drive; option 6 selects a high density diskette in the second (3 ½ inch) diskette drive. Option 7 selects a low density diskette in the second (5 ¼ inch) diskette drive; option 8 selects a low density diskette in the second (3 ½ inch) diskette drive. Your screen will then look similar to this:

```
Insert medium in the disk0_1.2M drive.
Press <RETURN> when ready [q].
```

3. Press ENTER after you have inserted the diskette and are ready to continue. Note that the drive name that appears on your screen depends on the drive you chose. The system then asks:

```
Disk 'sales.1', file system '/reports'.  Mount it? [y, n, q]
```

4. Type y to mount the diskette's file system. The system then displays:

```
/reports mounted. DO NOT REMOVE MEDIUM UNTIL IT IS UNMOUNTED!

Press the RETURN key to see the diskmgmt menu [?,q]
```

   You may now access the file system on the diskette using an ordinary INTERACTIVE UNIX System path name.

11.2.2 Mounting a File System on the Second Fixed Disk

Although fixed disk file systems are automatically mounted when the system is booted, at times you may need to mount one explicitly, for example, if you unmounted a file system to check or repair it. (Refer to sections 11.3.2 and 11.5.4 for information about unmounting and checking file systems on fixed disks.) A file system on a fixed disk can be mounted using the `sysadm` command. Before you can mount a file system, you must know two things:

- The name of the device that contains the file system
- The directory name of the file system "mount point"

Refer to section 9.4 for device file naming conventions. You will use the same device file name the system used when it created the file system at the time of system installation or when adding a second fixed disk as described in section 4.8 of the "INTERACTIVE UNIX Operating System Installation Instructions."

1. To mount a new fixed disk file system, access the Hard Disk Management submenu of the Disk Management menu, and select option `5, mounthdfsys`.
2. The system asks which file system to mount and lists the options:

   ```
   Which file system do you want to mount? (/usr, /usr2, q)
   ```

   Type the name of the file system you want to mount, for example, `/usr2`.
3. The system asks for the name of the device the file system resides on:

   ```
   Which device is the file system on? (0s1, 1s0, 1s1, ..., q)
   ```

   Type the name of the device that contains the file system, for example, `0s4`.
4. The system executes the `mount` command and displays a message similar to this:

   ```
   /etc/mount  /dev/dsk/0s4 /usr2
   ```

   The file system is now mounted and accessible to users.

11.3 Unmounting a File System

Before you can remove a diskette from the drive, you must *unmount* or deactivate it. You must also unmount a fixed disk partition that you do not want to be a part of the active file system.

11.3.1 Unmounting a Diskette-Based File System

Use the `sysadm` Disk Management menu to unmount a diskette-based file system. If any system users or programs are using any directory under the mount point of this file system, the unmount will fail.

1. Change to any directory not on the diskette, for example, the `root` directory:

```
$ cd /
```

2. Access the Disk Management menu from the `sysadm` menu. Select option 8, `umountfsys`. The system displays a screen similar to this:

```
Do you wish to unmount 'mnt'? [y, n, q]:
```

3. Type `y` to unmount the file system. The system responds:

```
/mnt unmounted.   You may now remove the medium from
the diskette drive.
Press the RETURN key to see the diskmgmt menu [?,q].
```

The file system is now unmounted and is no longer accessible.

11.3.2 Unmounting a File System on a Fixed Disk

You can unmount a file system on a fixed disk using the `sysadm umounthdfsys` command. If any system users or programs are using any directory under the mount point of this file system, the unmount will fail.

1. Access the Hard Disk Management submenu of the Disk Management menu, and select option 6, `umounthdfsys`.
2. The system lists the file systems that are currently mounted and asks which one(s) you wish to unmount:

```
Currently:
File System                Device
/usr is mounted on      /dev/dsk/0s3
/usr2 is mounted on     /dev/dsk/0s4

Select the file system(s) (/usr /usr2) ALL
you want to unmount [?, q]
```

Type the name(s) of the file system(s) you want to unmount, for example, `/usr2`.

3. The system asks for confirmation that this is what you intend:

```
Going to unmount /usr2.
Continue [y, n, ?, q]
```

Type `y`.

4. The system displays a message similar to this:

```
/etc/umount /usr2

Press the RETURN key to see the harddisk menu [?, q]
```

The file system `/usr2` is now unmounted.

11.4 Checking the Space Available on a File System

You can find the disk space available for each file system on your computer with the `df` command. To use the `df` command, type the following from the system prompt:

```
# df
```

The file system mount point, special device, available space, and available i-nodes (which determines the number of new files that can be created) will be displayed for each mounted file system. You can also use the `df` command with a file system argument and display the used and available disk space for that file system. Refer to *df*(1M) for additional information.

11.5 Checking and Repairing a File System

The INTERACTIVE UNIX Operating System has several reliability features built in. Every time a file is modified, the system performs a series of file system updates. These updates, when written to the disk, produce a consistent file system.

Every time the INTERACTIVE UNIX Operating System is booted, your computer runs a consistency check on the status of the `root` file system. If a potential problem exists, the `fsck` program automatically attempts to repair it. The `fsck` program is a file system check-and-repair program that uses information that is stored in the file system to check for inconsistencies. If the information it checks does not match, it detects the inconsistency and attempts to repair it. Because `fsck` runs automatically on the `root` file system when the system is booted, you should not have to run `fsck` manually for the `root` file system.

The `fsck` program can also be run manually to check diskettes that have INTERACTIVE UNIX System file systems on them; or if you suspect something is wrong with your file system, you may want to check it. This should only be attempted by expert users. For more information about `fsck`, refer to *fsck*(1M) and to section 4

of the "INTERACTIVE UNIX Operating System Maintenance Procedures."

11.5.1 File System Corruption and Increasing Reliability

A file system can become corrupt in a variety of ways. Three of the most common ways are:

- Improper system shutdown and startup
- Removing media before unmounting its file system
- Hardware failure

☛ To help keep your file systems reliable:

- Never remove a diskette while the disk drive is on (the red light on the disk drive is lit).
- Never remove a mounted diskette; always unmount it *before* you remove the diskette.
- Always use the `shutdown` command to bring the system down. This will unmount all file systems and ensure file system consistency.

11.5.2 Checking a File System

Use the `checkfsys` option on the Disk Management menu to run `fsck` on damaged diskettes. Use the `checkhdfsys` option on the Hard Disk Management submenu to run `fsck` on a damaged file system on a fixed disk.

It is possible to run the `fsck` command manually on a file system. Only an expert user should attempt to use the `fsck` command to check a diskette-based file system. Refer to section 4 in the "INTERACTIVE UNIX Operating System Maintenance Procedures" and *fsck*(1M) for more information if you need to run `fsck` manually.

If an inconsistency in the file system is found, `fsck` displays a message describing the problem. If the system experiences a power failure, it is usually possible to recover from it without suffering extensive file system damage. However, if the system experiences a hardware failure, the file system may be damaged beyond repair.

If file system damage is relatively minor, `fsck` will automatically repair many of the problems it encounters without assistance from you. In some cases the system asks you to confirm the action it plans to take. Let `fsck` repair as much of the file system as possible. Type `y` when `fsck` asks if it should proceed with a repair, or press ENTER to accept the defaults. If `fsck` cannot repair the problem, you may have to reinstall a backup version of the file system.

You may find that badly damaged file systems that cannot be repaired can sometimes be mounted read-only, using the `-r` flag from the command line.

11.5.3 Checking Diskette File Systems

To check the integrity of a diskette file system, use `sysadm` to access the Disk Management menu.

1. Select option 1, `checkfsys`. The system displays a screen similar to this:

```
Select which drive to use:

 1 disk0_1.2M     3 disk0_360K    5 disk1_1.2M     7 disk1_360K
 2 disk0_1.44M    4 disk0_720K    6 disk1_1.44M    8 disk1_720K
 9 tape

Enter a number, a name, the initial part of a name, or
? or <number>? for HELP,  q to QUIT:
```

2. Select the option that corresponds to the type of diskette in the drive you want to use. Option 1 selects a high density diskette in the first (5 ¼ inch) diskette drive; option 2 selects a high density diskette in the first (3 ½ inch) diskette drive. Option 3 selects a low density diskette in the first (5 ¼ inch) diskette drive; option 4 selects a low density diskette in the second (3 ½ inch) diskette drive. Option 5 selects a high density diskette in the second (5 ¼ inch) diskette drive; option 6 selects a high density diskette in the second (3 ½ inch) diskette drive. Option 7 selects a low density diskette in the second (5 ¼ inch) diskette drive; option 8 selects a low density diskette in the second (3 ½ inch) diskette drive. Your screen will then look similar to this:

```
Insert medium in the disk0_1.2M drive.
Press <RETURN> when ready [q].
```

3. Press ENTER after you have inserted the diskette and are ready to continue. Note that the drive name that appears on your screen will depend on the drive you chose. Your screen will look similar to this:

```
Disk 'mnt' file system 'mnt'
Select:
        check
        interactive
        automatic
    [c, i, a, q, ?]
```

4. Type `a` and press ENTER. `fsck` automatically checks and repairs any errors that it detects. Note that if you choose `check`, the system will not attempt any repairs. Choosing `interactive` causes `fsck` to ask for confirmation before attempting to repair any problems found while checking the file system.

5. At the end of the file system check, your screen will look similar to this:

```
/dev/SA/diskette
File system: /mnt    Volume: mnt

** Phase 1 - Check Blocks and Sizes
** Phase 2 - Check Pathnames
** Phase 3 - Check Connectivity
** Phase 4 - Check Reference Counts
** Phase 5 - Check Free List
  3 files    9 blocks  2345 free
```

Your file system is checked, repaired, and now ready for use.

11.5.4 Checking Fixed Disk File Systems

1. To check the integrity of a fixed disk file system, use `sysadm harddisk` to access the Hard Disk Management submenu.

   ☛ Before checking the integrity of a fixed disk file system, it is important to make sure all other users are off the system. The checking process starts immediately after you select the appropriate option, so you must warn users *before* you start this `sysadm` process.

2. Select option `3`, `checkhdfsys`. `fsck` automatically checks and repairs any errors that it detects. The system displays a screen similar to this:

```
All other users MUST be logged off the system.
Going to unmount and check the following file systems:

/usr

Continue [y, n, ?, q]
```

3. When the file system has been checked, your screen will look similar to this:

```
Press the RETURN key to see the Hard Disk Management
menu [?, q]
```

Your file system is checked, repaired, and now ready for use.

12. BACKING UP FILES

Once you have started to use your system, you will need to *back up* the files on your fixed disk (or disks) on a regular basis. The copying of files to removable media (such as a diskette or cartridge tape) is called a *backup*. This minimizes the possibility of lost data caused by hardware failures or operator error. It is important to perform backups on a regular basis because these copies are your only protection in the event of a file system crash.

There are two kinds of backups. A *complete* backup is used to back up all the files and directories on a particular file system. An *incremental* backup is used to back up only those files and directories that have changed since the last complete backup.

You should plan a backup strategy for your system that includes a regular complete backup, supplemented by more frequent incremental backups. You may also transfer a file, a set of files, or the contents of a directory onto a diskette for storage.

After performing a backup, keep the removable media in a secure place so that there are always up-to-date copies of the files on your fixed disk. If the system should become corrupted or a file is mistakenly removed from the fixed disk, the corrupted or missing files can be copied from the removable media to the fixed disk. The procedure of copying the backup files from the removable media to the fixed disk is called a *restore*.

Use the `sysadm` File Management menu to transfer data from your fixed disk to another storage medium, usually a diskette. This chapter explains how to back up file systems and individual files or directories on diskettes.

12.1 Before You Begin to Back Up Your System

When you back up a file system, you must either have a supply of formatted diskettes available or cartridge tapes, if your system is capable of backing up on tape. (Note that you do not need to format tapes for backup.) Depending upon the size of your file systems, a single backup can require many diskettes or several tapes. Be sure you have enough formatted diskettes or tapes available to complete the backup procedure before you run the `backup` procedure.

To format a diskette, use the `sysadm` Disk Management menu `format` option. You will be guided through the procedure step-by-step.

12.2 Backing Up File Systems

1. Access the `sysadm` File Management menu. Select option `1, backup`. Your screen will look similar to this:

```
Available file systems:
/   ALL

Enter file system(s) you want to back up [?, q]:
```

2. This example shows only one file system, `/`. You may have more than one file system if you have a large fixed disk or if you have established new file systems. Enter the name(s) of the file system(s) you want to back up. The system displays a screen similar to this:

```
Select complete or incremental backup [c, i, ?, q]:
```

   The first time you use `backup`, you should perform a complete backup. Remember, when you do a complete backup of a file system, *all* the files on a specified file system are transferred to diskettes. This may take some time to complete.

3. The system asks if the name of each file should be printed on your terminal screen as it is transferred to the diskette:

```
Print each file name as it is copied? [y, n, ?, q]:
```

   Type either `y` or `n`, depending upon your requirements.

4. The system asks which drive to use:

```
Select which drive to use:

 1 disk0_1.2M     3 disk0_360K    5 disk1_1.2M     7 disk1_360K
 2 disk0_1.44M    4 disk0_720K    6 disk1_1.44M    8 disk1_720K
 9 tape

Enter a number, a name, the initial part of a name, or
? or <number>? for HELP,  q to QUIT:
```

   Select the option that corresponds to the type of diskette in the drive you want to use. Check your manufacturer's documentation to determine this if you are unsure. Note that if you have a high density diskette drive, you can back up onto either a low or high density diskette. Option `1` selects a high density diskette in the first (5 ¼ inch) diskette drive; option `2`

selects a high density diskette in the first (3 ½ inch) diskette drive. Option 3 selects a low density diskette in the first (5 ¼ inch) diskette drive; option 4 selects a low density diskette in the second (3 ½ inch) diskette drive. Option 5 selects a high density diskette in the second (5 ¼ inch) diskette drive; option 6 selects a high density diskette in the second (3 ½ inch) diskette drive. Option 7 selects a low density diskette in the second (5 ¼ inch) diskette drive; option 8 selects a low density diskette in the second (3 ½ inch) diskette drive.

5. Next the system asks you to insert a diskette in the drive:

```
Complete backup of /, Thu Jun 25 10:20:32 PDT 1987
      part 1
Insert medium in the disk0_1.2M drive.
Press <RETURN> when ready
```

6. Note that the drive name that appears on your screen depends on the drive you selected previously. Press ENTER. The system copies all of the files on the named file system onto the diskette. If more than one diskette is required, you are asked to insert additional diskettes. When the backup procedure is complete, the system displays this screen:

```
Complete backup of / finished.

You may now remove the medium.

Press the RETURN key to see the filemgmt menu.
```

7. When you have finished, label the backup diskettes with the date of the backup and the name of the file system (and disk, if you are backing up multiple fixed disks).

12.3 Backing Up Individual Files and Directories

There may be occasions when you do not want to transfer a complete file system to a diskette. The `store` facility of the File Management menu may be used to transfer a file, a group of files, or the contents of a directory to a diskette.

1. Access option 7, `store`, on the File Management menu. The system asks which drive to use:

```
Select which drive to use:

1 disk0_1.2M     3 disk0_360K    5 disk1_1.2M     7 disk1_360K
2 disk0_1.44M    4 disk0_720K    6 disk1_1.44M    8 disk1_720K
9 tape

Enter a number, a name, the initial part of a name, or
? or <number>? for HELP,  q to QUIT:
```

2. Select the option that corresponds to the type of diskette in the drive you are using. Check your manufacturer's documentation to determine this if you are unsure. Note that if you have a high density diskette drive, you can back up to either a low or high density diskette. Option 1 selects a high density diskette in the first (5 ¼ inch) diskette drive; option 2 selects a high density diskette in the first (3 ½ inch) diskette drive. Option 3 selects a low density diskette in the first (5 ¼ inch) diskette drive; option 4 selects a low density diskette in the second (3 ½ inch) diskette drive. Option 5 selects a high density diskette in the second (5 ¼ inch) diskette drive; option 6 selects a high density diskette in the second (3 ½ inch) diskette drive. Option 7 selects a low density diskette in the second (5 ¼ inch) diskette drive; option 8 selects a low density diskette in the second (3 ½ inch) diskette drive.

3. The system then displays a screen similar to this:

```
1.  Select a single file for storing.
2.  Select all the files under a directory for storing.

Enter a number [?, q]:
```

4. Type 1 or 2, depending upon your requirements. If you select option 1 (store a single file), the system displays a screen similar to this:

```
Enter a full path name of file to be stored [q]:
```

   A similar response is displayed for option 2.

5. Type the path name of the file or directory you want to transfer to a diskette. The system displays another set of options:

```
1.  Select a single file for storing.
2.  Select all the files under a directory for storing.
3.  List files selected so far
4.  Store selected files
Enter a number [?, q]:
```

6. Select option 3 if you want to review the files that you have designated for storing. The system displays the path names

for the selected files. You may choose to add additional files or directories to your list of files to be stored. When you are ready to write the files to the diskette, select option 4, `Store selected files`. The system asks if you want the name of each file displayed on the screen as it is transferred to the diskette:

```
Print each file name as it is being stored? [y, n, ?, q]:
```

Select the response that is appropriate for your requirements.

7. The system indicates how you should label the diskette:

```
Before inserting the first part into the diskette driver,
mark it

     File store on:
Thu Jun 25 12:01:45 PDT 1987
        part 1

Insert medium in the disk0_1.2M drive.
Press <RETURN> when ready [q]:
```

Note that the drive name that appears on your screen will depend on the drive you selected previously. Press ENTER. The system transfers the files to the diskette.

8. The system asks if you want to verify that the transfer was completed properly. It is a good idea to verify the validity of the transfer. Your screen will look like this:

```
/etc/profile
/usr/robin/letter.1
5 blocks
Store complete
Do you want to verify that your file(s) were
stored properly? [y,n,?,q]

PLEASE NOTE:
To verify that the store worked properly, you must
re-insert all parts that were just written to,
starting with "part 1"

Insert the medium in the disk0_1.2M drive.
Press <RETURN> when ready [q]:

5 blocks
Verification complete.
```

9. If you no longer require the information on the fixed disk, answer y to the following query:

```
Should the files stored be deleted from the built-in
disk? [y,n,?.q]:
```

10. You may now remove the diskette from the diskette drive.

If you need to transfer this data from the diskette back to your fixed disk, use the `restore` option on the File Management menu.

12.4 Restoring Files

You will need to restore files in the event of hardware or user errors. Your users should be aware that some data may be lost, because any modifications made to the files since the time of the last backup will not be reflected in the restored files.

1. Access the `sysadm` File Management menu. Select option `6, restore`. This option can be used to restore files that were copied using the `sysadm backup` and `store` commands. Your screen will look similar to this:

```
Option 1 restores only the individual files selected.
Option 2 restores all files AND directories below the
directory that you specify.
Option 3 restores every file from the media.
Option 4 lists the file names from the media without
restoring them.
REMEMBER - If you do not rename a file, "restore" will
write over any copies of files that are presently on
the file systems.  Also, you MUST use a full path name
when renaming files.

Select:
1. restore single files
2. restore directories of files
3. restore all files
4. list all the files
Enter a number [?, q]:
```

2. Type the number of the option that corresponds to the type of backup you want to do, for example, `3`. The system then asks:

```
Do you want to rename the files as they are copied in?
```

3. In most cases, you will probably answer `n`. If you answer `y`, the following warning will appear:

```
Be very careful when you rename a file.  Files incorrectly
named by typing errors are difficult to find and repair.
Remember that only the first 14 characters of each part of
the file name (i.e., the characters between the slashes (/)
are significant.  You will be asked to rename each file in
turn.  An empty response skips that file.  An answer of ``.''
(period) restores the file with its original name.
```

4. The files are then copied and your screen looks similar to this:

```
You may now remove the medium from the n drive.

Restoration complete.
```

12.5 Restoring XENIX Files

The INTERACTIVE UNIX Operating System includes the `xrestore` command, which restores backups that were made under the XENIX Operating System. See *xrestore*(1) for information on restoring XENIX file system backups.

13. MANAGING DEVICES

13.1 What Are Devices and Device Drivers?

In the context of your computer system, a *device* is generally defined as a piece of hardware. For example, a diskette drive, a printer, or an on-board modem are all considered devices. For each device that you use with your computer system, you must have an associated software program called a *device driver*. The device driver is responsible for communicating between the hardware and the application program or operating system. It ensures that the device receives and carries out the instructions given by the user. Drivers for your devices can be supplied with the operating system, with the application program, or by the manufacturer of the device.

The INTERACTIVE UNIX Operating System is delivered with device drivers that support a wide variety of devices, such as commonly used terminals and printers. These drivers support your hardware in its standard configuration and allow you to easily add new devices. Some of these device drivers, such as the line printer driver, are configured into the kernel by default. If you plan to add a device that is already *supported* in this way by the INTERACTIVE UNIX System, it is not necessary to install a new device driver. Section 7 of the "INTERACTIVE UNIX Operating System Maintenance Procedures" details which drivers are already configured into the kernel. To add a printer to your system, refer to section 7 of this document; to add a terminal, refer to section 9 of the "INTERACTIVE UNIX Operating System Maintenance Procedures."

Several device drivers that are available with your operating system are not configured into the kernel but are in the *Additional Drivers* optional subset, which is delivered on two diskettes. You only need to insert the appropriate diskette and load the driver you want. Drivers available for installation in this subset are:

- *Additional Drivers I Diskette: Terminal Support*
  - DigiCHANNEL MC/Xi Asynchronous Driver
  - Bell Technologies ICC* Multiport Card Driver
  - HUB Multiport Serial Card Driver
  - SunRiver* Fiber Optic Station Drivers
  - UnTerminal* Drivers

- *Additional Drivers II Diskette: Peripheral Devices Support*
  - INTERACTIVE MultiView DeskTop HSM
  - Wangtek Tape Controller Driver
  - Archive* Cartridge Tape Driver
  - LOGITECH* Bus Mouse Driver
  - Microsoft* Mouse Driver
  - PS/2* Mouse driver
  - Floppy Tape Drivers

Each time you add one of these devices to your system, you must physically install the device, add its device driver to the system, and reconfigure the system kernel.

13.2 The Kernel

The kernel is the core of the INTERACTIVE UNIX Operating System and all UNIX-based operating systems. It is the program that is responsible for managing the hardware and software resources of the computer. It manages user logins, runs programs, and keeps track of input to and output from devices. The kernel is stored in an ordinary file (`/unix`) and is tailored to a specific hardware and software configuration. You loaded the standard INTERACTIVE UNIX System kernel when you installed your INTERACTIVE UNIX Operating System.

If you plan to add new pieces of hardware to your system, in most cases you will have to reconfigure your system to support the new devices. If the type of device you plan to add is not already supported in your standard hardware configuration, then you must also reconfigure the system kernel. This process builds a new kernel that includes the driver for the new device.

INTERACTIVE has attempted to make this process as easy for you as possible by providing two system enhancements. First, a friendly, menu-driven interface called `kconfig` is provided to help you configure your kernel. `kconfig` is used to configure, rebuild, and install new kernels. (For additional information about `kconfig`, see the *kconfig*(1) manual entry.) Like `sysadm`, `kconfig` provides step-by-step procedures that are designed to make it easy to perform a common set of system administration tasks. The

`kconfig` program is discussed in later sections, as appropriate. Second, the *High Performance Disk Driver* is provided to increase the range of supported devices. It is discussed in the following section and in sections 6 and 7 of the "INTERACTIVE UNIX Operating System Maintenance Procedures."

13.3 The High Performance Disk Driver

The INTERACTIVE UNIX System kernel contains a set of integrated software packages called the High Performance Disk Driver (HPDD, for short.) The HPDD allows several controllers to share a single driver, making it easier for you to add and use different types of controllers. It also significantly improves the speed of your system during most operations.

The HPDD supports most AT*-compatible fixed disk controllers, two popular SCSI host adapters (including cartridge tape support), and a RAM disk. (A RAM disk is created by reserving a portion of the computer's available memory, which is then treated as if it were a disk storage device. Refer to section 7.1 of the "INTERACTIVE UNIX Operating System Maintenance Procedures" for more information about RAM disks. Refer to the release notes that accompanied your INTERACTIVE UNIX System for a list of controllers supported by the HPDD.) Each time you add, remove, or change any of *these* devices, you must change your system configuration by:

- configuring the HPDD to support the new configuration,
- rebuilding and installing the kernel, and
- physically installing the hardware (except in the case of the RAM disk).

Note that there are two exceptions to this. If you are replacing a standard AT controller of one type with a different standard AT controller, you do not need to reconfigure the HPDD. Also, if you are simply adding a fixed disk drive to a controller that you have already configured, you may not need to reconfigure the HPDD.

You will use the `kconfig` program to reconfigure the HPDD.

These procedures are discussed in section 6 of the "INTERACTIVE UNIX Operating System Maintenance Procedures."

13.4 Tailoring Your System Kernel

Your system kernel is initially configured to support a basic hardware and software configuration. The default kernel includes drivers for the following configuration:

Hardware:

- A keyboard
- A fixed disk and diskette controller
- A monochrome, color, EGA, or VGA display adapter
- Two serial communications ports (COM1, COM2)
- Up to three parallel ports
- A real time clock
- CMOS RAM

UNIX System V Release 3 Software, including:

- MS-DOS* file system service
- UNIX System V file system service
- Common interprocess communication routines, including:
  - –Interprocess communication message facility
  - –Interprocess communication semaphore facility
  - –Shared memory
- Intel* 80387 floating point co-processor (80287 supported on COMPAQ 386*)
- 287 or 387* floating point co-processor emulator

Your vendor may deliver a different default configuration. Check the documentation supplied with your system to determine its default configuration.

You will need to reconfigure and rebuild your system kernel to:

- Add a new device and its driver
- Reconfigure the High Performance Disk Driver
- Remove a device and its driver
- Add or remove certain software packages

- Change the system memory size
- Add or change tunable parameters

You will use the `kconfig` utility to configure, build, and install a new kernel after you make any of the changes listed above. Refer to section 6 in the "INTERACTIVE UNIX Operating System Maintenance Procedures" to learn how to reconfigure and rebuild your system kernel.

14. ADMINISTERING SYSTEM SECURITY

Security deserves special consideration on a multi-user system. The information you have stored on your computer belongs to you and no one else. It is a valuable resource that requires protection. Three security features provided for your computer are:

- System backups
- Access permissions
- Passwords

14.1 Sources of Potential Damage

To provide adequate security for your system, you first need to consider the kinds of problems that might arise. It is important to consider every possible contingency: theft of the hardware, breakdowns in the hardware or software, tampering with data in the computer by unauthorized people, theft of data, and simple human error. To determine the types of problems to which your system may be vulnerable, ask yourself the following questions:

- *Are you using your computer in a large or a small office? Is it possible for someone to steal your computer?*

  Prevent theft by setting up your computer in a secure place.

- *How many people have ready access to your system?*

  Restrict the number of people with access to your computer by using passwords.

- *How sensitive or valuable is the information you are storing?*

  Restrict the number of people who have access to sensitive data using the access permissions.

- *Do you transmit data over telephone lines?*

  Limit access to your computer telephone numbers.

How you answer these questions will help you decide the measures you want to take to safeguard your data.

14.2 Basic Precautions

The following safeguards are recommended to everyone for ensuring the security of their computer and data:

- Set up your computer in an area that can be secured when you are not using it.
- Never leave your computer terminal logged in and unattended.
- If you are using your computer in an office environment, restrict the number of people who have access to the computer.

14.3 System Backups

If a breakdown occurs in the system, you may lose information that you had stored in the computer. You can avoid this type of loss if you have copies of your files stored on removable storage devices. Copy your data periodically onto diskettes or cartridge tapes and store them in a safe place, away from the computer site. Refer to section 12 for more information about the backup procedure.

14.4 Access Permissions

If you are sharing your computer with other users, you may want to share some information while keeping other information private. The INTERACTIVE UNIX Operating System allows you to satisfy both of these needs by letting you define:

- The users that have permission to access data
- The types of permission they have (that is, how they are allowed to use the data)

The INTERACTIVE UNIX System recognizes three categories of users of stored data: the owner of the data, the group to which the owner belongs, and all other people who use the system. Users can be given permission to use data in any or all of three ways: to read, write, and/or execute it. Refer to the "INTERACTIVE UNIX Operating System Primer" for more information about access permissions.

Whenever you create a file or directory, the system automatically identifies the users who will be allowed to read, write, and/or execute the file and the users who will be allowed to access the directory. As the owner of the file, only you and the superuser (usually the system administrator) can change the access permission after it has been created.

One way to restrict access to your files is to change the permission modes. Refer to the "INTERACTIVE UNIX Operating System

Primer," *chmod*(1), and *ls*(1) for more information about changing access permissions.

To keep people from looking at the contents of a directory, you can deny execute permission to that directory. Others will be prevented from listing or looking at the contents of your directory.

Whenever a file or directory is created, the permission modes are automatically set for everyone to have access, unless the `umask` command is set in your system profile or personal `.profile`. You can use the `chmod` command to change the permissions each time you create a file, or you can use the `umask` command to change the default permission modes of a file that you will be creating. Individual users can place the `umask` command in their `.profile`, or you can alter the system default profile, `/etc/profile`. Type:

```
umask nnn
```

nnn represents three octal digits that refer to read, write, and execute permission for owner, group, and other, respectively. The value of each permission digit is subtracted from the corresponding permissions digit. For example, if you add `umask 022` to `/etc/profile`, all files normally created with 777 permission will now be created with 755 permissions. (`umask 022` is set by default in `/etc/profile` in the INTERACTIVE UNIX Operating System.) A file created with 666 permissions will be created with 644 permissions. See *umask*(1) for additional information.

14.5 Password Administration

If you are at all worried about security, it is a good idea to force users to maintain passwords. When a login is first established, a password can be assigned, or one can be assigned later. The superuser can assign or change a password for any login. For more information about the password files and their management, refer to section 6, "MANAGING USER ACCOUNTS."

GLOSSARY

This glossary defines terms and acronyms used in this document that may not be familiar to you.

administrative login account
: A login account used to give ordinary users restricted access to system management functions that must be performed frequently.

back up
: To copy files to removable media (such as a diskette or cartridge tape). A backup is the spare copy of data or software that you keep in case the original is damaged or lost. A backup may be partial or complete.

baud rate
: The speed at which a device that is connected to a tty line operates.

block files
: Files that reside in `/dev/dsk`, which reference block devices, such as fixed disks, that access hardware one block (sector or record) at a time.

character files
: Files that reside in `/dev/rdsk`, which reference character devices, such as terminals, which generally process data one character at a time.

configuration files
: Files that store information that affects the environment of an individual user or affects the system on a global basis. Many INTERACTIVE UNIX System configuration files are shell command scripts or contain shell variable assignments.

console terminal
: The console terminal is comprised of the screen and keyboard directly attached to the Central Processing Unit (CPU). It is important for the system administrator to use the console terminal since most error messages generated by the system are displayed on the console terminal screen.

daemon A program that runs as a background process to handle UNIX System activities, for example, file transfers, command executions, and cleanup routines.

device In the context of a computer system, a device is generally defined as a piece of hardware. For example, a diskette drive, a printer, or an on-board modem are all considered devices.

device driver
A software program associated with each specific device that is responsible for communicating between the hardware and the application program or operating system. It ensures that the device receives and carries out the instructions given by the user. Drivers for devices can be supplied with the operating system, with the application program, or by the manufacturer of the device.

environment variable
A word relating to the user's environment that is assigned a value and has a special meaning to the INTERACTIVE UNIX Operating System. For example, the default terminal type is a variable that is usually assigned a value for each user.

High Performance Disk Driver
A set of integrated software packages that are part of the INTERACTIVE UNIX System kernel (also called the HPDD, for short). The HPDD allows several controllers to share a single driver, making it easier to add and use different types of controllers and significantly improves the speed of the system during most operations.

mount point
The directory where a file system will be installed.

ordinary login account
Every user on the system uses an ordinary login account to perform most tasks. Ordinary logins include a login ID, a password, and an area of the file system assigned to the user, called a home directory.

partitions
: The fixed disk is divided into sections called partitions. You may have only one partition that takes up the entire fixed disk, or you may have several partitions, each containing its own operating system, such as the UNIX System or DOS.

restore
: The procedure of copying the backup files from the removable media to the fixed disk.

root directory
: The main (top level) directory in the `root` file system, which is automatically created and given the unique name of / (slash) when your system is installed and initialized.

root file system
: The permanent file system established on the first subpartition of the INTERACTIVE UNIX System partition on the first fixed disk. It contains all of the important files and subdirectories that are required to run the INTERACTIVE UNIX Operating System.

root user
: See *superuser.*

special files
: Files that do not contain data. For example, device files are called special files because they are references to the actual programs that run the peripheral devices attached to your computer.

spooler
: The term "spool" is an acronym for **s**imultaneous **p**eripheral **o**perations **o**n-**l**ine. The line printer spooling system allows you to send a file to be printed while you continue with other work.

superuser
: The most privileged user on the system (also called the `root` user). There are no restrictions imposed upon `root`, which means that the person logged in to the system with the `root` ID can access, modify, and delete every file and process on the system.

swap area
: An area of the computer used to store portions of programs or data that have been temporarily swapped out of main memory.

system administrator
: The individual responsible for installing, administering, and maintaining your INTERACTIVE UNIX Operating System.

system dump
: A procedure used by experienced system administrators to obtain detailed information about the state of the system when it crashed.

system login account
: Accounts used to perform system administration tasks that require privileged access to the restricted files and directories on the system. The two most important system logins are `root` and `bin`.

unmount
: To deactivate (make inaccessible) the file system on a diskette or fixed disk.

utility
: A program. Related utilities can be combined into a package that represents a specific application available with your computer.

variable
: A variable is a word that is assigned a value and has a special meaning to the INTERACTIVE UNIX Operating System. For example, the default terminal type is a variable that is usually set for each user.

INDEX

INTERACTIVE

A Kodak Company

DOC0111-2Z

INTERACTIVE UNIX* System V/386 Release 3.2 Guide for New Users

INTERACTIVE