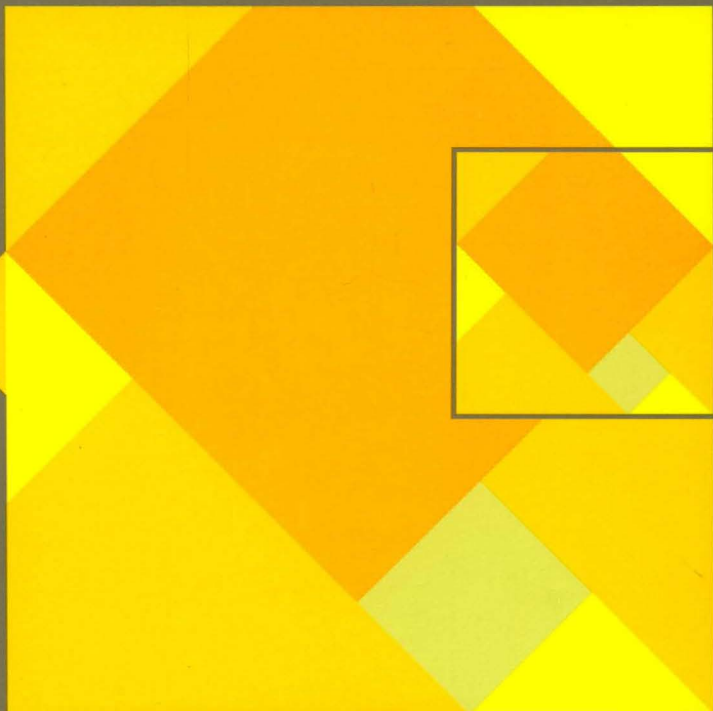




XENIX[®] Tutorial



XENIX[®] System V

Operating System

XENIX Tutorial

Information in this document is subject to change without notice and does not represent a commitment on the part of The Santa Cruz Operation, Inc. nor Microsoft Corporation. The software described in this document is furnished under a license agreement or nondisclosure agreement. The software may be used or copied only in accordance with the terms of the agreement. It is against the law to copy this software on magnetic tape, disk, or any other medium for any purpose other than the purchaser's personal use.

Portions © 1980, 1981, 1982, 1983, 1984, 1985, 1986, 1987, 1988 Microsoft Corporation.

All rights reserved.

Portions © 1983, 1984, 1985, 1986, 1987, 1988 The Santa Cruz Operation, Inc.

All rights reserved.

ALL USE, DUPLICATION, OR DISCLOSURE WHATSOEVER BY THE GOVERNMENT SHALL BE EXPRESSLY SUBJECT TO RESTRICTIONS AS SET FORTH IN SUBDIVISION (b) (3) (ii) FOR RESTRICTED RIGHTS IN COMPUTER SOFTWARE AND SUBDIVISION (b) (2) FOR LIMITED RIGHTS IN TECHNICAL DATA, BOTH AS SET FORTH IN FAR 52.227-7013.

Microsoft, MS-DOS, and XENIX are trademarks of Microsoft Corporation.

UNIX is a trademark of AT&T Bell Laboratories.

Contents

1 Introduction

- 1.1 Introduction 1-1
- 1.2 About This Tutorial 1-1
- 1.3 Notational Conventions 1-2

2 Basic Concepts

- 2.1 Introduction 2-1
- 2.2 Accounts 2-1
- 2.3 Files 2-2
- 2.4 Naming Conventions 2-6
- 2.5 Commands 2-10
- 2.6 Input and Output 2-13

3 Logging In

- 3.1 Introduction 3-1
- 3.2 Gaining Access to the System 3-1
- 3.3 Keeping Your Account Secure 3-5
- 3.4 Changing Your Terminal Type 3-6
- 3.5 Entering Commands 3-7

4 Working with Files and Directories

- 4.1 Introduction 4-1
- 4.2 Working with Directories 4-1
- 4.3 Editing Files with vi 4-7
- 4.4 Printing Files 4-10
- 4.5 Working with Files 4-13
- 4.6 Processing Text Files 4-20
- 4.7 Using File and Directory Permissions 4-23

5 Housekeeping

- 5.1 Introduction 5-1
- 5.2 Making Backups 5-1
- 5.3 Copying Diskettes 5-9
- 5.4 Getting Status Information 5-10
- 5.5 Controlling Processes 5-12
- 5.6 Shell Programming 5-17

6 XENIX Desktop Utilities

- 6.1 Introduction 6-1
- 6.2 Using the System Clock and Calendar 6-1
- 6.3 Using the Mail Service 6-2
- 6.4 Using the Automatic Reminder Service 6-7
- 6.5 Using the Calculator 6-8

Chapter 1

Introduction

- 1.1 Introduction 1-1
- 1.2 About This Tutorial 1-1
- 1.3 Notational Conventions 1-2

1.1 Introduction

This tutorial is an introduction to the XENIX™ Operating System. It is intended for users who have little or no familiarity with XENIX or UNIX™.



The XENIX operating system is distributed with over two hundred commands and programs. This tutorial does not explain how to use all of these. Instead, the commands and programs described in this tutorial are those that you will use most often, and those that you will find most useful.

The aim of this tutorial is to teach you how to do useful work on a XENIX system as quickly as possible. To this end, it is not necessary to provide you with complete information about each command described in this tutorial. For complete information, refer to the appropriate sections of the *XENIX User's Reference* and the *XENIX User's Guide*.

1.2 About This Tutorial

This tutorial is organized as follows:

- Chapter 1, “Introduction,” presents an overview of the contents of the entire tutorial, and explains how to use it.
- Chapter 2, “Basic Concepts,” explains the concepts that you need to understand to work effectively in the XENIX environment. The chapters that follow presuppose an understanding of the material presented in this chapter.
- Chapter 3, “Logging In,” explains how to log in to the system, how to keep your account secure, how to edit the login prompt and how to enter XENIX commands.
- Chapter 4, “Working with Files and Directories,” explains how to perform some of the basic tasks involving files and directories. This chapter explains how to create files and directories, how to move, copy, delete, and rename files and directories. The chapter also explains how to use various XENIX text processing utilities, and how to use access permissions with files and directories.

- Chapter 5, “Housekeeping,” explains how to use the XENIX “housekeeping” utilities. This chapter explains how to create backups, how to copy diskettes, how to get information about the status of the system, and how to place commands in the background. The chapter also contains a brief discussion of XENIX shell programming.
- Chapter 6, “XENIX Desktop Utilities,” explains how to use the XENIX desktop utilities. This chapter explains how to use the automatic reminder service, how to communicate with other users on the system and how to use the system calculator.

The best way to use this tutorial is to begin by reading Chapter 2. This will provide you with the background information that you need in order to understand the material presented in subsequent chapters. You should then read Chapters 3 through 6 at your terminal, entering commands as instructed in the examples.

Each section of each chapter is a self-contained unit. You do not have to read previous sections in order to understand the material presented in any particular section. If you only need to know how to perform a specific task, for example, if you already know how to create text files but you do not know how to print them, you can turn to the section of the chapter that explains how to perform that task. In this case, you can turn to section 4.4 of Chapter 4. You do not have to read sections 4.1 through 4.3 in order to understand the material presented in section 4.4.

1.3 Notational Conventions

This tutorial uses the following notational conventions:

- Examples in the text are indented.
- **Commands** that you enter are printed in **boldface** type.
- *Directories* and *filenames* are printed in *italics*.
- **Keys** to be pressed are printed in **boldface** type. For example, the Return key is represented by:

RETURN

- **Key combinations** are printed in **boldface** and are hyphenated. An example is:

Ctrl-d



When you see a key combination, you are supposed to hold down the first key and press the second key. In this example, you should hold down the Control key and press the d key.

- An uppercase letter in parentheses is often appended to command names, as in:

touch(C)

The letter in parentheses refers to the section of the *XENIX User's Reference* that contains complete information on the command.

Chapter 2

Basic Concepts

- 2.1 Introduction 2-1
- 2.2 Accounts 2-1
 - 2.2.1 User Accounts 2-1
 - 2.2.2 Super User Account 2-2
- 2.3 Files 2-2
 - 2.3.1 Ordinary Files 2-2
 - 2.3.2 Special Device Files 2-3
 - 2.3.3 Directory Files 2-3
 - 2.3.4 Directory Structure 2-4
- 2.4 Naming Conventions 2-6
 - 2.4.1 Filenames 2-6
 - 2.4.2 Pathnames 2-6
 - 2.4.3 Sample Names 2-7
 - 2.4.4 Special Characters 2-8
- 2.5 Commands 2-10
 - 2.5.1 Command Line 2-11
 - 2.5.2 Syntax 2-12
- 2.6 Input and Output 2-13
 - 2.6.1 Redirection 2-13
 - 2.6.2 Pipes 2-14

2.1 Introduction

This chapter explains the basic concepts that you need to understand to work effectively in the XENIX environment. After reading this chapter, you should understand the fundamentals of user accounts, as well as how the system's files and directories are organized and named, how commands are entered, and how a command's input and output can be redirected. It is important to read this chapter before proceeding to the tutorial chapters that follow.

2.2 Accounts

2.2.1 User Accounts

An account must be created for you before you can log on to a XENIX system. Your account contains the following information:

- Your login name. This is the name by which you are known on the system. It is the name you enter at the login prompt.
- Your password. To increase system security, each user may be given a password. This password is entered when you log on to the system.
- Your group identification. Each user is known to the system as an individual and as a member of a group. Group membership is important for system security reasons. As a member of a group, you may be permitted to access files and directories that you cannot access as an individual.
- Your "home directory." This is the place in the filesystem where you can keep personal files. When you first log in to the system, you are placed in your home directory.
- Your "login shell." This is the program that reads and executes the XENIX commands you input. In most cases, your login shell will be the "Bourne Shell." The Bourne Shell uses the dollar sign (\$) as a prompt. However, you may be configured to use the "C-Shell," which uses the percent sign (%) as a prompt, or the "Visual Shell," which is a menu-driven interface. Throughout this tutorial, the expression "XENIX prompt" is used to refer to your shell prompt, whether it is the percent sign or the dollar sign.

Once an account has been established for you, you can manipulate the files, directories, and commands that make up the XENIX system.

2.2.2 Super User Account

2 In addition to each user's individual account, every XENIX system has a "super user" account. (The super user is also referred to as "root.") In order to perform certain system administration tasks, the system administrator must log in as the super user. The super user has free rein over the system. The super user can read and edit any file on the system, as well as execute any program.

2.3 Files

The file is the fundamental unit of the XENIX filesystem. There are three different types of XENIX files: ordinary files (what we usually mean when we say "file"), special device files, and directories. Each of these is described in the sections that follow.

2.3.1 Ordinary Files

An ordinary file is simply a collection of 8-bit bytes. Ordinary files are usually documents, program source code, or program data. Executable binary files, or computer programs, are also considered ordinary files. The bytes of an ordinary file are interpreted as text characters, binary instructions, or program statements, by the programs that examine them.

Every ordinary file has the following attributes:

- a filename (not necessarily unique),
- a unique filesystem number called an inode number,
- a size in bytes,
- a time of last change,
- a set of access permissions,
- an owner and a group.

File Protection

On a multi-user system, it is often necessary to “protect” certain files, denying some users access to the files while allowing access to others. Files are protected by assigning appropriate “access permissions” to them. XENIX provides three levels of access permissions:

- | | |
|----------------|---|
| read | Having read permission on a file allows a user to view the contents of the file with such commands as cat and more . A user with read-only permission cannot edit a file. |
| write | Having write permission on a file allows a user to edit the file. |
| execute | If the file is a program, having execute permission on the file allows a user to run the program. You cannot run a program for which you do not have execute permission. |

Access permissions are assigned by a file’s owner. (By default, the owner of a file is its creator.) Any combination of the three levels is permitted. This allows the file’s owner to determine which users can read, write, and/or execute the file. Note that the super user has read, write, and execute permissions on all files on the system.

The file security mechanism implemented under XENIX is very flexible. It allows separate access permissions to be set for a file’s owner, a file’s group, and for all other users. In a typical case, the owner of a file might have read and write permissions, the group read-only permission, and all other users no access permissions at all.

2.3.2 Special Device Files

Each physical device on the system, such as hard and floppy disks, line-printers, terminals, and system memory, is assigned to a “special file.” These files are also called “special device files.” Special device files are not discussed in this tutorial. (For more information on special device files, see the *XENIX System Administrator’s Guide*.)

2.3.3 Directory Files

Directory files are more like file drawers than files. They are places where files are stored (conceptually, not physically). Directory files are usually referred to as “directories.”

A directory contains the names and inode numbers of the files “within it.” These inode numbers point to “inodes.” XENIX uses inodes internally to organize filesystems. There is one inode per file. Inodes contain information about files. This information includes the file type, the number of links to the file, the location of the file on the disk, the size of the file, the identity of the file’s owner and group, the file’s access permissions, and the time the file was last modified.

2

Like ordinary files, directories can be protected by assigning appropriate access permissions. These are read, write and execute. In order to do anything useful in a directory, a user must have execute permission on that directory. Execute and write permissions determine whether files can be added to or removed from a directory. Execute and read permissions determine whether the contents of a directory can be listed. Access permissions are assigned to a directory by its owner. By default, the owner of a directory is its creator.

2.3.4 Directory Structure

With multiple users working on multiple projects, the number of files in a filesystem can proliferate rapidly, creating an organizational nightmare. The inverted “tree-structured” directory hierarchy that is a feature of the XENIX system allows users to organize large numbers of files efficiently. Related files can be grouped together in a single directory. In addition to ordinary files, a directory can contain other directories, sometimes called “subdirectories.” Subdirectories themselves can contain ordinary files and more subdirectories, and so on. The `cd` command is used to move from one directory to another.

In this typical tree of files, the root of the tree is at the top and the branches of the tree grow downward. Directories correspond to “nodes” in the tree, while ordinary files correspond to “leaves.” Figure 2-1 represents this inverted tree-structured directory hierarchy.

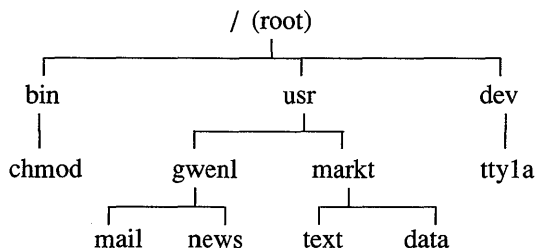


Figure 2-1 A Typical Filesystem

In Figure 2-1, the names *bin*, *usr*, *dev*, *gwenl*, and *markt* all represent directories, and are all nodes in the tree. At the top of the tree is the root directory, which is given the name slash (/). The names *mail*, *news*, *text*, and *data* all represent ordinary files, and they are all “leaves” of the tree. The file *chmod* is the name of a command that can be executed. The name *ttyla* is a special device file. It represents a terminal and is also represented in the tree.

If a directory contains a downward branch to other files or directories, those files and directories are “contained” in the given directory. All directories and files on the system are contained in the root directory. In Figure 2-1, the files *mail* and *news* are contained in the directory *gwenl*, which itself is contained in the directory *usr*. The directory *usr*, in turn, is contained in the root directory.

It is possible to name any file in the system by starting at the root and traveling down any of the branches to the desired file. Files can also be named relative to any directory. XENIX naming conventions are discussed later in this chapter.

The User Directory

Each XENIX user is given a personal or “home” directory. This is a place where you can keep files that no other user is likely to need. Within the home directory, you may have other subdirectories that you own and control. All of the home directories on a XENIX system are often placed in the *usr* directory, as illustrated by Figure 2-2.

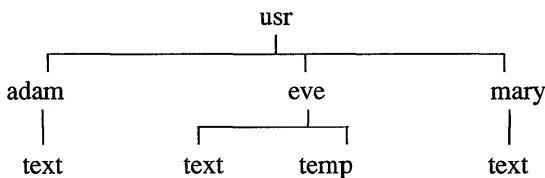


Figure 2-2 A Typical User Directory

In Figure 2-2, the *usr* directory contains each user’s home directory. There are three users on this system, *adam*, *eve*, and *mary*.

2.4 Naming Conventions

Every single file, directory, and device in XENIX has both a filename and a pathname. Filenames and pathnames are discussed in the following two sections.

2

2.4.1 Filenames

A filename is a sequence of 1 to 14 characters consisting of letters, digits and other special characters such as the underbar (`_`). Every single file, directory, and device in the system has a filename. Although you can use almost any character in a filename, it is best to confine filenames to the alphanumeric characters and the period. Other characters, especially control characters, are discouraged for use in filenames.

Filenames should be indicative of a file's contents. For example, a file containing purchase orders should have a name like *orders*, rather than *file1*. Note that filenames must be unique only within directories and need not be unique system-wide. Different directories can contain different files that have the same name. For example, there can be several files named *text* on a single system, as long as those files are each in separate directories.

When a filename contains an initial period, it is “hidden,” and it is not displayed by the `ls` command. System configuration files are often hidden. However the `ls -a` command does display hidden files. The dash (`-`) is used in specifying command options and should be avoided when naming files. In addition, the question mark (`?`), the asterisk (`*`), brackets (`[` and `]`), and all quotation marks should *never* be used in filenames, because they have special meaning to the XENIX shell. (For more information on these characters, see “Special Characters” later in this chapter.)

2.4.2 Pathnames

A pathname is a sequence of directory names followed by a simple filename, each separated from the previous name by a slash. If a pathname begins with a slash, it specifies a file that can be found by beginning a search at the *root* of the entire tree. Otherwise, files are found by beginning the search at the user's *current directory* (also known as the *working directory*). The `pwd` command is used to print the name of the working directory on the screen.

A pathname beginning with a slash is called a *full* or *absolute pathname*. The absolute pathname is a map of a file's location in the system. Absolute pathnames are unique: no two files, directories, or devices have the exact same absolute pathname. A pathname *not* beginning with a slash is called a *relative pathname*, because it specifies a path relative to the current directory.

2.4.3 Sample Names

Among the directory and file names commonly found on XENIX systems are:

/	The name of the root directory.
/bin	The directory containing most of the frequently used XENIX commands.
/usr	The directory containing each user's personal directory. The subdirectory, <i>/usr/bin</i> contains frequently used XENIX commands not in <i>/bin</i> .
/dev	The directory containing special device files.
/dev/console	The special device file associated with the system console.
/dev/ttyXX	The names of special device files associated with system ports. <i>XX</i> represents a number, such as <i>1a</i> or <i>006</i> . Most ports are assigned to terminals.
/lib	The directory containing files of "libraries" used for system development.
/usr/lib	The directory containing directories with XENIX applications.
/tmp	The directory for temporary files.
/usr/joe/run	A typical full pathname. It is the pathname of a file named <i>run</i> belonging to a user named <i>joe</i> .

`bin/script` A relative pathname. It names the file *script* in subdirectory *bin* of the current working directory. If the current directory is the root directory (`/`), it names */bin/script*. If the current directory is */usr/joe*, it names */usr/joe/bin/script*.

`file1` Name of an ordinary file in the current directory.

2

All files and directories, with the exception of the root directory, have a “parent” directory. This directory is located immediately above the given file or directory. The XENIX filesystem provides special shorthand notations for the parent directory and for the current directory:

- The shorthand name of the current directory. For example, *./filexxx* names the same file as *filexxx*, in the current directory.
- .. The shorthand name of the current directory’s parent directory. For example, the shorthand name *../..* refers to the directory that is two levels “above” the current directory.

2.4.4 Special Characters

XENIX provides a facility for specifying sets of filenames that match particular patterns. Suppose, for example, you are working on a large book. The different chapters of the book might be kept in separate files, whose names might be *chpt1*, *chpt2*, *chpt3*, and so on. You might even break each chapter into separate files. For example, you might have files named *chpt1.1*, *chpt1.2*, *chpt1.3*, and so on.

If you want to print the whole book on the lineprinter, you could enter the following command:

lp chap1.1 chap1.2 chap1.3 ...

Entering so many filenames in a command quickly becomes tedious, and will probably lead to mistakes. Fortunately, there is a shortcut. A sequence of names containing a common pattern can be specified with the use of special “wildcard” characters. The wildcard characters discussed in this chapter are:

- * Matches zero or more characters
- [] Matches any character inside the brackets
- ? Matches any single character

For example, you can enter:

lp chap*

The asterisk (*) means “zero or more characters of any type,” so this command translates into “send all files whose names begin with the word *chap* to the lineprinter.” This is a quick and effective way of printing all the files that make up your book.

This shorthand notation is not a unique property of the **lp** command. It can be used with any command. For example, you can list the names of the files in the book by typing:

lc chap*

The asterisk is not limited to the last position in a filename. It can be used anywhere in a filename and can occur several times. An asterisk by itself matches all filenames not containing slashes or beginning with periods:

cat *

This command displays all files in the current directory on your terminal screen.

The asterisk is not the only pattern-matching feature available. Suppose you want to print only chapters 1 through 4, and 9. You can enter:

lp chap[12349]*

The brackets ([and]) mean “match any of the characters inside the brackets.” A range of consecutive letters or digits can be abbreviated, so you can also do this with the following command:

```
lp chap[1-49]*
```

(Note that this does *not* try to match *chap1** through *chap49**, but rather *chap1** through *chap4** and *chap9**.) Letters can also be used within brackets: “[a-z]” matches any character in the range “a” through “z”.

The question mark (?) matches any single character:

```
lc ?
```

This command lists all files that have single-character names. The following command lists information about the first file of each chapter (i.e., *chap1.1*, *chap2.1*, ...):

```
l chap?.1
```

If you need to turn off the special meaning of any of the wildcard characters (*, ?, and [...]) enclose the entire argument in single quotation marks. For example, the following command lists only a file named “?” rather than all one-character filenames:

```
lc ‘?’
```

Pattern-matching features are discussed further in “The Shell” chapter of the *XENIX User’s Guide*.

2.5 Commands

You have already been introduced to three useful XENIX commands, **lc**, **lp**, and **cat**. The **lc** command is used to display directory contents, the **lp** command to print files and the **cat** command to display file contents.

Commands are executable programs. When you enter the name of a command, XENIX looks for a program with that name and executes the program, if it can be found. Command lines can also contain arguments that specify options or files that the program needs. The command line and command syntax are discussed in the next two sections.

2.5.1 Command Line

XENIX always reads commands from the “command line.” The command line is a line of characters that is read by the shell to determine what actions to perform. (XENIX offers three shells, the Bourne Shell, the C-Shell and the Visual Shell.) The shell reads the names of commands from the command line, finds the executable program corresponding to the name of the command, then executes that program. When the program finishes executing, the shell resumes reading the command line.



When you enter commands at a terminal, you are actually editing a line of text called the “command-line buffer.” The command-line buffer becomes a command line only when you press **RETURN**. The command-line buffer can be edited with the **BKSP** and **Ctrl-u** keys. If the **INTERRUPT** key is pressed before **RETURN**, the command-line buffer is erased. (On most keyboards, the **Del** key is the **INTERRUPT** key.)

Multiple commands can be entered on a single command line, provided they are separated by a semicolon (;). For example, the following command line prints out the current date and the name of the current working directory:

```
date; pwd
```

Commands can be submitted for processing in the “background” by appending an ampersand (&) to the command line. This mode of execution is similar to “batch” processing on other systems. The main advantage of placing commands in the background is that you can execute other commands from your terminal in the “foreground” while the background commands execute. For example, the following command outputs disk usage statistics in the directory */usr*, a fairly time-consuming operation, without tying up your terminal:

```
du /usr > diskuse &
```

The output of this **du** command is placed in the file *diskuse*, by redirecting output with the greater-than symbol **>**. (Redirection of input and output is discussed in “Input and Output” below. Background processing is discussed in “Advanced Tasks.”)

2.5.2 Syntax

The general syntax for commands is:

```
cmd [ options ][ arguments ][ filename ][ ... ]
```

2

By convention, command names are lowercase. Options are always preceded by a dash (-) and are not required. They are used to modify the command. For example, the **lc** command lists, in columnar format, the contents of a directory. The same command with the **-l** option (**lc -l**) produces a long listing of a directory's contents.

In some cases, options can be grouped to form a single option argument, as in the following command:

```
lc -rl
```

This command is really a combination of two options, where the **-rl** option selects the option that lists all files in the directory in both reverse alphabetical order and with the long format.

Sometimes multiple options must be given separately, as in the following command:

```
copy -a -v source destination
```

Here the **-a** option tells the **copy** command to ask the user for confirmation before copying *source* to *destination*. The **-v** option specifies "verbose", which causes **copy** to list the names of the files that are copied, as they are copied.

Other arguments, such as search strings, can also be given, as in the following command:

```
grep 'string of text' data.file
```

The *string of text* in this example is a single argument, and is the string for which the **grep** command searches in the file *data.file*.

2.6 Input and Output

By default, XENIX assumes that input comes from the terminal keyboard and that output goes to the terminal screen. To illustrate typical command input and output, enter:

cat

This command now expects input from your keyboard. It accepts as many lines of input as you enter, until you press **Ctrl-d**, which is the “end-of-file” or “end-of-transmission indicator.”

2

For example, enter:

**this is two lines RETURN
of input RETURN
Ctrl-d**

The **cat** command immediately outputs each line as you enter it. Since output is sent to the terminal screen by default, that is where the lines are sent. Thus, the complete session will look like this on your terminal screen:

```
$ cat
this is two lines
this is two lines
of input
of input
$
```

The flow of command input and output can be “redirected” so that input comes from a file instead of from the terminal keyboard and output goes to a file or lineprinter, instead of to the terminal screen. In addition, you can create “pipes” to use the output of one command as the input of another. (Redirection and pipes are discussed in the next two subsections.)

2.6.1 Redirection

In XENIX, a file can replace the terminal for either input or output. For example, the following command displays a list of files on your terminal screen:

lc

But if you enter the following command, a list of your files is placed in the file *filelist* (which is created if it does not already exist), rather than sent to the screen:

```
lc > filelist
```

2

The symbol for output redirection, the greater-than sign (>), means “put the output from the command into the following file, rather than display it on the terminal screen.” The following command is another way of using the output redirection mechanism:

```
cat f1 f2 f3 > temp
```

This command places copies of several files in the file *temp* by redirecting the output of **cat** to that file.

The output append symbol (>>) works very much like the output redirection symbol, except that it means “add to the end of.” The following command means “concatenate *file1*, *file2*, and *file3* to the end of whatever is already in *temp*, instead of overwriting and destroying the existing contents.”

```
cat file1 file2 file3 >> temp
```

As with normal output redirection, if *temp* doesn’t already exist, it is created for you.

In a similar way, the input redirection symbol (<) means “take the input for a program from the following file, instead of from the terminal.” As an example, you could enter the following command to send a file named *letter.txt* to several people using the XENIX **mail** facility:

```
mail adam eve mary joe < letter.txt
```

(See Chapter 6 of this tutorial for information on **mail**.)

2.6.2 Pipes

One of the major innovations of the XENIX system is the concept of a “pipe.” A pipe is simply a way to use the output of one command as the input of another, so that the two run as a sequence of commands called a “pipeline.”

For example, suppose that you want to find all unique lines in *frank.txt*, *george.txt*, and *hank.txt* and view the result. You could enter the following sequence of commands:

```
sort frank.txt george.txt hank.txt > temp1
uniq < temp1 > temp2
more temp2
rm temp1 temp2
```

2

But this is more work than is necessary. What you want is to take the output of **sort** and connect it to the input of **uniq**, then take the output of **uniq** and connect it to **more**. You would use the following pipe:

```
sort frank.txt george.txt hank.txt | uniq | more
```

The vertical bar character (|) is used between the **sort** and **uniq** commands to indicate that the output from **sort**, which would normally have been sent to the terminal, is to be used as the input of the **uniq** command, which in turn sends its output to the **more** command for viewing.

The following command is another example of a pipe. The **wc** command counts the number of lines, words, and characters in its input. The **who** command prints a list of users currently logged on, one per line. Thus, the following pipeline tells you the number of users who are logged in by counting the number of lines that come from the **who** command:

```
who | wc -l
```

Notice the difference in output between **wc -l** and **wc**. By default, **wc** tells you how many lines, words, and characters there are in the input. However, **wc -l** tells you only how many lines.

Any program that accepts input from the keyboard can accept input from a pipe instead. Any program that displays output to the terminal screen can send input to a pipe. You can have as many elements in a pipeline as you wish.

Chapter 3

Logging In

- 3.1 Introduction 3-1
- 3.2 Gaining Access to the System 3-1
 - 3.2.1 Logging In 3-1
 - 3.2.2 Logging Out 3-3
 - 3.2.3 Changing Your Password 3-3
- 3.3 Keeping Your Account Secure 3-5
 - 3.3.1 Password Security 3-5
 - 3.3.2 Good Security Habits 3-5
 - 3.3.3 Using Another User's Account 3-6
- 3.4 Changing Your Terminal Type 3-6
- 3.5 Entering Commands 3-7
 - 3.5.1 Entering a Command Line 3-7
 - 3.5.2 Erasing a Command Line 3-8
 - 3.5.3 Halting Screen Output 3-8

3.1 Introduction

This chapter explains how to perform the following basic tasks on a XENIX system:

- Log in to the system,
- Log out of the system,
- Change your password,
- Use another user's account,
- Reset your terminal type,
- Enter a XENIX command,
- Erase an incorrect command line,
- Stop and start screen output.



This chapter is designed as a tutorial. The best way to use this chapter is to read it at your terminal, entering commands as instructed in the examples.

None of the commands described in this chapter is described in great detail. For a complete explanation of each command, refer to the *XENIX User's Reference*.

3.2 Gaining Access to the System

To use the XENIX system, you must first gain access to it by logging in. When you log in, you are placed in your home directory. Logging in, changing your password, and logging out are described below.

3.2.1 Logging In

Before you can log into the system, you must be given a system "account." In most cases, your account is created for you by your system administrator. However, if you need to create the account yourself, refer to the *XENIX System Administrator's Guide* for information on creating user accounts. This section assumes that your account has already been created.

Normally, the system sits idle and the prompt “login:” appears on the terminal screen. If your screen is blank or displays nonsense characters, press the **INTERRUPT** key a few times. On most keyboards, the **INTERRUPT** key is the **Del** key.

When the “login:” prompt appears, follow these steps:

1. Enter your login name and press **RETURN**. If you make a mistake as you type, press **Ctrl-u** to start the line again (hold down the **Ctrl** key and press the **u** key). After you press **RETURN**, “Password:” appears on your screen.
2. Enter your password and press **RETURN**. The letters of your password do not appear on the screen as you enter them, and the cursor does not move. This is to prevent other users from learning your password. If you enter your login name or password incorrectly, the system displays the following message:

```
Login incorrect
login:
```

If you get this message, enter your login name and password again.

3. Depending on how your system is configured, you may or may not be prompted to enter your terminal type. If you are prompted for your terminal type, you see a line like the following:

```
TERM= (unknown)
```

Enter your terminal type if you see this line. (If you are not sure how to specify your terminal type, contact your system administrator.)

Once you have entered all the correct information, the “prompt character” appears on the screen. This is a dollar sign (\$) for Bourne Shell users and a percent sign (%) for C-shell users. The prompt tells you that the XENIX system is ready to accept commands from the keyboard.

Depending on how your system is configured, you may also see a “message of the day” after you log in.

3.2.2 Logging Out

The simplest way to log out is to enter **logout** at the % prompt for C-shell users, or **exit** at the \$ prompt for Bourne Shell users. You might also be able to logout by pressing **Ctrl-d** at the prompt. However, some systems are configured to prevent logout with **Ctrl-d**. The reason for this is that **Ctrl-d** signifies the end-of-file in XENIX, and it is often used within programs to signal the end of input from the keyboard. Since people sometimes make the mistake of pressing **Ctrl-d** several times, they often find themselves unintentionally logged out of the system. To prevent this, system administrators may disable logout with **Ctrl-d**.

Familiarize yourself with the logout procedure by pressing **Ctrl-d**, if you are currently logged in. If this does not work, log out by entering **logout** (C-shell) or **exit** (Bourne Shell). If you are not logged in, log in and then log out, experimenting with **Ctrl-d** and with **logout** or **exit**.

3.2.3 Changing Your Password

To prevent unauthorized users from gaining access to the system, each authorized user can be given a password. When you are first given an account on a XENIX system, you are assigned a password by the system administrator. Some XENIX systems require you to change your password at regular intervals. Whether yours does or not, it is a good idea to change your password regularly (at least once every two months) to maintain system security.

Use the **passwd** command to change your password. Follow these steps:

1. Enter the following command and press **RETURN**:

passwd

You see:

```
Changing password for user  
Old password:
```

Your login name appears in place of *user*.

2. Carefully enter your old password. It is not displayed on the screen. If you make a mistake, press **RETURN**. The message “Sorry” appears, then the system prompt. Begin again with step 1.
3. The following message appears after you enter your old password and press **RETURN**:

```
New password:
```

Enter your new password and press **RETURN**. It is generally a good idea to use a combination of numbers and lower-case and upper-case letters in your password.

4. You see the following message:

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

Enter your new password again. If you make a mistake, you see the following message:

```
They don't match; try again
```

Begin again with step 1 if you see this message.

When you complete the procedure, the XENIX prompt reappears. The next time you log in, you must enter your new password.

3.3 Keeping Your Account Secure

Security is ultimately the responsibility of the user. The careless use and maintenance of passwords represents the greatest threat to the security of a computer system.

3.3.1 Password Security

Here are some specific guidelines for passwords:

1. Don't use passwords that are easy to guess. Passwords should be at least six characters long and include letters, digits, and punctuation marks. (Example: frAiJ6*)
2. Passwords should not be names (even nicknames), proper nouns, or any word found in */usr/dict/words*. (Don't use a password like: terry9)
3. Always keep your password secret. Passwords should never be written down, sent over electronic mail, or verbally communicated. (Treat it like the PIN number for your instant teller card.)

3.3.2 Good Security Habits

There are simple, good security habits. Here are some general guidelines:

1. Remember to log out before leaving a terminal.
2. Use the **lock(C)** utility when you leave your terminal, even for a short time.
3. Make certain that sensitive files are not publically readable. (See the discussion of file and directory permissions in Chapter 4 of this tutorial for information on how to do this.)
4. Keep any floppies or tapes containing confidential data (program source, database backups) under lock and key.



5. If you notice strange files in your directories, or find other evidence that your account has been tampered with, tell your system administrator.

3.3.3 Using Another User's Account

You might find it necessary to work with another user's files. However, the permission settings on those files may prevent all but the files' owner from editing them. To overcome this problem, you can use the **su** command to change your current account to that of the other user. To use **su**, you must know the other user's password.

3

For example, to become user joe, enter the following command at the XENIX prompt, not at the login prompt:

```
su joe
```

When the password prompt appears, enter joe's password. To cancel the effect of the **su** command and return to your own account, press **Ctrl-d**.

3.4 Changing Your Terminal Type

On most systems, the system console is already configured for use with XENIX. However, serial terminals of various types can be connected to a XENIX system. If you are working from a serial terminal, it can be important to know how to specify your terminal type.

The terminal type is displayed each time you log in. You can change the value of the terminal type displayed by editing the *.profile* file in your home directory. If you are using the C-shell, you do not have a *.profile* file. Instead, you must edit the *.login* file in your home directory.

There are at least two reasons why you might want to change the value of the terminal type displayed:

- You might have a new terminal that is not the same model as your old terminal. If so, the terminal type displayed by your old *.profile* (*.login*) file will be incorrect.

- The terminal type displayed might be “unknown” or “ansi” or another setting which is not correct for your terminal. This would require you to type in your terminal type every time you log in. By changing the terminal type to the setting that is correct for your terminal, all you have to do is press **RETURN** when prompted for your terminal type. There is no need for you to enter the terminal type.

To change the terminal type displayed, use **vi** to edit *.profile* (*.login*). Chapter 4 of this tutorial explains how to use **vi**.

Once in **vi**, move the cursor to the line that looks like the following:

```
eval 'tset -m :?unknown -s -r -Q'
```

Change *unknown* (or whatever the value is) in this line to the terminal type of your terminal. For example, if you normally log in on a vt100 terminal, you would change the line to:

```
eval 'tset -m :?vt100 -s -r -Q'
```

Each time you log in, you would then see the message:

```
TERM = (vt100)
```

Press **RETURN** and the terminal type is set to vt100. There is no need to enter **vt100**.

3.5 Entering Commands

Before you begin working with the commands described in the rest of this tutorial, you should be familiar with three very useful XENIX features. These are character type-ahead and the special key-combinations used to erase the command line, and stop and start screen output. These features are discussed below.

3.5.1 Entering a Command Line

Entering a command line consists of typing characters and then pressing **RETURN**. Once you press **RETURN**, the computer reads the command line and executes the specified commands. No command entered on the command line is executed until **RETURN** is pressed.

You can enter as many command lines as you want without waiting for the commands to complete their execution and for the prompt to reappear. This is because XENIX supports character type-ahead. XENIX can hold up to 256 characters in the kernel buffers that read keyboard input. Experiment with this type-ahead feature by entering the following commands, one right after the other, without waiting for a previous command to finish executing. (Always press **RETURN** after entering a command. In the following example, press **RETURN** after entering each command.)

```
lc -la
du -a
lc -Fa
```

These commands generate a long listing of all the files in the current directory, then display disk usage statistics for these files, and finally list the files again, but in a different format.

3.5.2 Erasing a Command Line

Typing errors are bound to occur when you enter commands. To erase the current command line, press **Ctrl-u**. When you press **Ctrl-u**, a new prompt is displayed and no command is executed.

3.5.3 Halting Screen Output

Data often scrolls across your screen faster than you can read it. To halt scrolling temporarily, press **Ctrl-s**. To restart scrolling, press **Ctrl-q**. Experiment with **Ctrl-s** and **Ctrl-q** by entering the following command, then pressing **Ctrl-s** to stop the output and **Ctrl-q** to restart it:

```
ls /bin
```

Chapter 4

Working with Files and Directories

- 4.1 Introduction 4-1
- 4.2 Working with Directories 4-1
 - 4.2.1 Printing the Name of Your Working Directory 4-2
 - 4.2.2 Listing Directory Contents 4-2
 - 4.2.3 Changing Your Working Directory 4-3
 - 4.2.4 Creating Directories 4-4
 - 4.2.5 Removing Directories 4-5
 - 4.2.6 Renaming Directories 4-6
 - 4.2.7 Copying Directories 4-6
- 4.3 Editing Files with vi 4-7
 - 4.3.1 Entering Text 4-7
 - 4.3.2 Moving the Cursor 4-8
 - 4.3.3 Deleting Text 4-8
 - 4.3.4 Inserting Text 4-9
 - 4.3.5 Leaving vi 4-9
- 4.4 Printing Files 4-10
 - 4.4.1 Using lp 4-10
 - 4.4.2 Using lp Options 4-11
 - 4.4.3 Cancelling a Print Request 4-12
 - 4.4.4 Finding Out the Status of a Print Request 4-12
- 4.5 Working with Files 4-13
 - 4.5.1 Displaying File Contents 4-14
 - 4.5.2 Deleting Files 4-16
 - 4.5.3 Combining Files 4-16
 - 4.5.4 Renaming Files 4-17
 - 4.5.5 Moving Files 4-18
 - 4.5.6 Copying Files 4-18
 - 4.5.7 Finding Files 4-19

- 4.6 Processing Text Files 4-20
 - 4.6.1 Comparing Files 4-20
 - 4.6.2 Sorting Files 4-21
 - 4.6.3 Searching for Patterns in a File 4-21
 - 4.6.4 Counting Words, Lines, and Characters 4-22
- 4.7 Using File and Directory Permissions 4-23
 - 4.7.1 Changing File Permissions 4-26
 - 4.7.2 Changing Directory Permissions 4-28

4.1 Introduction

This chapter explains how to perform the following tasks on a XENIX system:

- Print the name of the current directory,
- List directory contents,
- Change to another directory,
- Create, remove, rename, and copy directories,
- Use the full-screen text editor `vi` to create files,
- Print files,
- Display the contents of files,
- Delete, combine, rename, move, copy, and search for files,
- Compare and sort files,
- Search for patterns in a file,
- Count words, lines and characters in a file,
- Use file and directory permissions.



This chapter is designed as a tutorial. The best way to use this chapter is to read it at your terminal, entering commands as instructed in the examples.

None of the commands described in this chapter is described in great detail. For a complete explanation of each command, refer to the *XENIX User's Reference*.

4.2 Working with Directories

Because of the hierarchical structure of the XENIX filesystem, any XENIX system has many directories and subdirectories. XENIX includes several commands that simplify working in directories. These commands are described in the following sections.

4.2.1 Printing the Name of Your Working Directory

The directory you are “in” at any given time is your “working” directory. All commands are executed relative to the working directory. The name of this directory is given by the **pwd** command, which stands for “print working directory.” To find out what your current working directory is, enter the following command:

```
pwd
```

When you first log in to the system, you are placed in your home directory.

4.2.2 Listing Directory Contents

Several related commands are used to list the contents of directories:

4

- lc** This command is a variation of the **ls** command. The **ls** command alphabetizes and displays directory contents. The **lc** command alphabetizes directory contents and displays them in columnar format.
- lf** This command does the same as **lc**, and it also marks directories with a slash (/) and executable files (computer programs) with an asterisk (*).

Enter the following command to list the contents of */usr/bin*:

```
lc /usr/bin
```

This directory contains many of the executable files with which you work in the XENIX environment. Entering **lc** with no directory name lists the contents of the current directory.

The **l** command is also useful. It is equivalent to the **ls -l** command, which produces a “long” listing of a directory’s contents. The output of this command looks like the following listing:

```
total 338
-rw-rw-r-- 1 markt pub 4448 Mar 1 09:16 1.intro.00
-rw-rw-r-- 1 markt pub 4457 Mar 1 09:29 1.intro.s
-rw-rw-r-- 1 markt pub 33836 Mar 1 09:30 2.concepts.00
-rw-rw-r-- 1 markt pub 35096 Mar 1 12:49 2.concepts.s
-rw-rw-r-- 1 markt pub 52197 Mar 1 15:09 3.basic.s
-rw-rw-rw- 1 markt pub 39835 Feb 16 11:02 4.advan.s
```

Reading from left to right, the information given for each file or directory by the **l** command includes:

- Permissions
- Number of links
- Owner
- Group
- Size in bytes
- Time of last modification
- Filename

The figure at the top lists the total number of “blocks” used on the disk to store these files. A single block is 512 bytes. 338 blocks, or 173056 bytes, are needed to store the files listed above.

4.2.3 Changing Your Working Directory

Your working directory represents your location in the filesystem. To move to a new location in the XENIX filesystem, use the **cd** command.

XENIX Tutorial

Entering **cd** with no arguments places you in your home directory. Try it. Enter **cd**. To be sure you are now in your home directory, enter **pwd**.

To move to a directory other than your home directory, you must specify that directory as an argument to the **cd** command. For example, enter the following command to move to */usr/bin*:

```
cd /usr/bin
```

Verify that you are in */usr/bin* by entering **pwd**.

Change to the “root” directory by entering the following command:

```
cd /
```

The root directory is at the “top” of the filesystem. All other directories are “below” it. Enter **ls** to examine the files and directories in the root directory. Then enter **cd** to return to your home directory. (For more information on the root directory, refer to Chapter 2 of this tutorial.)

Some shorthand notation is available to help you move quickly through the filesystem. To move up one directory from your current directory, enter:

```
cd ..
```

Enter the following command to move up two directories:

```
cd ../..
```

If you entered this latter command from your home directory, you are probably in the root directory. Verify this by entering **pwd**.

4.2.4 Creating Directories

To create a subdirectory in your working directory, use the **mkdir** command. Enter **cd** to move to your home directory and then enter the following command to create a subdirectory named *tempdir*:

```
mkdir tempdir
```

Verify that *tempdir* exists with the **lf** command. Change to *tempdir* with the **cd** command and verify that *tempdir* is empty with another **lf** command. Finally, use the **touch(C)** command to create two empty files in *tempdir*:

```
touch tempfile1 tempfile2
```

Enter **lf** one more time to verify that *tempfile1* and *tempfile2* were created.

You can only create subdirectories in a directory if you have write permission in that directory. If you do not have write permission and you use **mkdir** to create a subdirectory, you see the following message:

```
mkdir: cannot access directory_name
```

In this message, *directory_name* refers to the directory in which you attempted to create a subdirectory. Verify this by trying to create a subdirectory in the */etc* directory, a directory in which you probably do not have write permission:

```
mkdir /etc/temp
```

4

4.2.5 Removing Directories

Use the **rmdir** command to remove a directory. This command will not work if the directory has files or subdirectories in it. Verify this by moving to your home directory with the **cd** command and then entering the following command to remove *tempdir*, the directory created in section 4.2.4:

```
rmdir tempdir
```

You should see the following message:

```
rmdir: tempdir not empty
```

You must remove *tempfile1* and *tempfile2* from *tempdir* before **rmdir** deletes *tempdir*. But don't remove these files just yet. They are used in another example later in this chapter.

4.2.6 Renaming Directories

To rename a directory, use the **mv** command. For example, **cd** to your home directory and then enter the following command to rename *tempdir*, the directory created in section 4.2.4, to *newdir*:

```
mv tempdir newdir
```

Verify the name change by entering **lf**. Note that the files in *newdir* are unaffected by the change. Verify this by entering the following command:

```
lf newdir
```

4.2.7 Copying Directories

4

The **copy** command copies directories. Of course, before you can copy the contents of one directory into another, you must have write permissions on the second directory.

To copy the */newdir* directory created in section 4.2.6, and all of the files in it, to */tmp/newdir*, enter the following command:

```
copy $HOME/newdir /tmp/newdir
```

In this command, “\$HOME” is shorthand for the pathname of your home directory. You can use it wherever you would enter the pathname of your home directory.

To verify that the files in *\$HOME/newdir* were copied to */tmp/newdir*, enter the following command:

```
lf /tmp/newdir
```

Remove */tmp/newdir* by entering the following commands:

```
rm /tmp/newdir/*  
rmdir /tmp/newdir
```

The first command removes the files in */tmp/newdir*, the second command removes */tmp/newdir*. Verify that */tmp/newdir* is removed by entering the following command:

```
If /tmp
```

Remove *\$HOME/newdir* by entering the following commands:

```
rm $HOME/newdir/*
rmdir $HOME/newdir
```

4.3 Editing Files with vi

The **vi** text editor is a full-screen editor that is included in the XENIX operating system package. The sections that follow briefly explain how to use **vi**. For a more complete discussion, see the *XENIX User's Guide*.

4

4.3.1 Entering Text

Change to your home directory with the **cd** command and enter the following command to create a file called *tempfile*:

```
vi tempfile
```

A message appears indicating that you are creating a new file. You are then placed in **vi**.

There are two modes in **vi**: *Insert mode* and *Command mode*. Use *Insert mode* to add text to a file. Use *Command mode* to edit existing text in a file. Since *tempfile* is empty, press **i** to enter *Insert mode*.

Enter the following lines of text, pressing **RETURN** after each line. If you make a mistake typing, use the **BKSP** key to erase the mistake and continue typing:

```
This tutorial is very, very helpful.
It makes learning to use the XENIX system easy.
I'm glad I have this tutorial.
```

After you enter the last line, press the **ESC** key. It takes you out of *Insert mode* and places you in *Command mode*.

4.3.2 Moving the Cursor

Although many cursor-movement commands are available in **vi**, only the four basic ones are discussed here:

- h** When you are in Command mode, pressing the **h** key moves the cursor one character to the left.
- l** When you are in Command mode, pressing the **l** key moves the cursor one character to the right.
- k** When you are in Command mode, pressing the **k** key moves the cursor up one line.
- j** When you are in Command mode, pressing the **j** key moves the cursor down one line.

4

Experiment with these cursor-movement keys on the text you entered. Note that, if your keyboard has arrow keys, these usually perform in the manner of **h**, **l**, **k** and **j**.

4.3.3 Deleting Text

Deleting text with **vi** is very easy. Different commands allow you to delete characters, words and entire lines.

To delete a single character, place the cursor on that character with the cursor-movement keys and press the **x** key. Experiment with the **x** key by deleting the comma in the first line.

To delete a word, place the cursor on the first character of the word and press **dw** (press **d**, release it, and press **w**). Experiment with this by placing the cursor on the first character of “very” in the first line and pressing **dw**.

To delete an entire line, place the cursor anywhere on that line and press **dd** (press **d**, release it, and press **d** again). Experiment with this by placing the cursor on the third line and pressing **dd**. Your file should now contain the following text:

This tutorial is very helpful.
It makes learning to use the XENIX system easy.

4.3.4 Inserting Text

The **i** and **o** keys can be used to insert text. We have already used the **i** key to enter text in an empty file. To enter additional text on an existing line, move the cursor to the point where you want the new text to begin, press **i** to enter Insert mode, enter the text, and press **ESC** to return to Command mode. For example, move the cursor to the “e” in “easy” in the second line, press **i**, enter the word **very**, press the **SPACEBAR**, and press **ESC** to return to Command mode. The second line should now be:

It makes learning to use the XENIX system very easy.

The **o** key can be used to insert a new line. To use it, move the cursor to the line *directly above* the place in the file where the new line is to be inserted and press **o**. A new line is inserted, with the cursor placed at the beginning. You are also automatically placed in Insert mode. Try this by moving the cursor to the second line of *tempfile* and press **o**. Now enter more text. Press **ESC** when you are finished.

4

4.3.5 Leaving vi

Most of the time, you will want to save your file before leaving **vi**. To do this, enter Command mode and type **:x**. This command saves the file you are editing and returns you to the XENIX prompt.

In some cases, you will want to leave **vi** without saving your work. To do this, enter Command mode and type **:q!**. This command returns you to the XENIX prompt, without saving the changes that you made to your file.

Leave *tempfile* by pressing **:x**. Re-enter *tempfile* by entering the following command:

vi tempfile

Insert some text using either the **i** or the **o** key, press **ESC** and then enter **q!** to quit without saving your changes. Display *tempfile* by entering the following command:

```
cat tempfile
```

You will notice that the last set of changes you made do not appear. Remove *tempfile* by entering the following command:

```
rm tempfile
```

4.4 Printing Files

To print files, use the **lp** command. This is one of a group of commands known as the “lineprinter” commands. The lineprinter commands are easy to use and very flexible. With a few simple commands, you can print multiple copies of a file, cancel a print request, or ask for a special option on a particular printer. Check with your system administrator to find out what lineprinters and printer options are available on your system.

4.4.1 Using lp

Use **cd** to change to your home directory and enter the following command to create a file with which you can experiment:

```
cp /etc/motd $HOME/printfile
```

This command places a copy of */etc/motd* in your home directory, naming it *printfile*. The file */etc/motd* is the “message of the day file.” Its contents are displayed every time you log in to the XENIX system.

A directory must be “publicly executable” before you can use **lp** to print any of the files in that directory. This means that other users must have execute permissions on the directory. Enter the following command to make your home directory publicly executable:

```
chmod o+x $HOME
```

(See “Using File and Directory Permissions” later in this chapter for more information on **chmod(C)**.)

Enter the following command to print *printfile*:

lp printfile

This command causes one copy of *printfile* to print on the default printer on your system. A banner page might be printed along with the file. Note that you can print several files at once by putting more than one name on the **lp** command line.

When you print with **lp**, a “request ID” is displayed on your screen. A request ID might look like the following:

pr4-532

The first part (pr4) is the name of the printer on which your file is printed. The second part (532) identifies your job number. Should you later wish to cancel your print request or check its status, you will need to remember your request ID. (Cancelling and checking on print requests are discussed below.)

You can also use **lp** with pipes. For example, enter the following command to sort and then print a copy of */etc/passwd*, the file that contains system account information:

sort /etc/passwd | lp

(For more information on **sort(C)**, see “Sorting Files” later in this chapter.)

4.4.2 Using **lp** Options

The **lp** command has several options that help you control the printed output. You can specify the number of copies you want printed by using the number option, **-n**. For example, to print two copies of *printfile*, enter:

lp printfile -n2

Several different printers are often attached to a single XENIX system. With the **-d** option, you can specify the printer on which your file is to be printed. To print two copies of *printfile* on a printer named *laser*, enter:

lp printfile -n2 -dlaser

Check with your system administrator for the names of the printers available on your system.

4.4.3 Cancelling a Print Request

Use the **cancel** command to cancel a print request. The **cancel** command takes as its argument the request ID. For example, to stop printing a file with a request ID of *laser-245*, you would enter:

```
cancel laser-245
```

Experiment with **cancel** by using **lp** to print *printfile* and then using **cancel** to cancel the print request. When you are finished, enter the following command to remove *printfile*:

```
rm printfile
```

You can also use the **cancel** command to stop whatever is currently printing on a particular printer. For example, to cancel whatever file is currently printing on the printer *laser*, you would enter the following command:

```
cancel laser
```

If you cancel a file that does not belong to you, mail reporting that the print request was canceled is automatically sent to the file's owner.

4.4.4 Finding Out the Status of a Print Request

Use the **lpstat** command to check on the status of your print request. To use it, simply enter the following:

```
lpstat
```

The **lpstat** command produces output like the following:

```
prt1-121   cindym    450   Dec 15 09:30
laser-450   cindym    4968  Dec 15 09:46
```

Note that entering **lpstat** with no options displays information on your files only, not those of other users. To generate a report for all users on your computer, use **lpstat** with the **-o** option. Nothing is displayed by the **lpstat** command if the print job is complete.

The first column of the **lpstat** output shows the request ID for each of your files being printed. The second column is your login name. In the third column, the number of characters to be printed is shown, and the fourth column lists the dates and times the print requests were made.

To learn the status of a particular file, use the **lpstat** command with the file's request ID. For example, to find out the status of a file with the request ID of *laser-256*, you would enter the following command:

```
lpstat laser-256
```

The status of that file only is displayed.

You can also request the status of various printers on your system by using the **-p** option or by giving the name of the particular printer you are interested in. Enter the following command to find out the status of all the printers on your system:

```
lpstat -p
```

To find out the status of a printer named *laser*, you would enter the following:

```
lpstat -plaser
```

The request ID and status information for each file currently waiting to be printed on *laser* is displayed.

4.5 Working with Files

File manipulation (creating, deleting, displaying, combining, renaming, moving, and copying) is one of the most important capabilities an operating system provides. The XENIX commands that perform these functions are described in the following sections.

4.5.1 Displaying File Contents

The **more** command displays the contents of a file, one screenful at a time. It cannot be used to edit files. If the file contains more than one screenful of data, you see the following prompt after each screen of text is displayed:

```
--More-- (XX%)
```

XX% represents the percentage of the file displayed. Press the **RETURN** key to display another line. Press the **SPACEBAR** to display another screen.

Try the following command:

```
more /etc/rc
```

This causes the contents of */etc/rc* to display on the screen. To quit **more** before */etc/rc* is finished displaying, press **q** for quit.

The **more** command does not allow you to scroll backward, toward the beginning of the file. However, you can search forward for patterns with **more** by using the slash (/) command. For example, enter the following commands to search for a line containing "process" in */etc/rc*:

```
more /etc/rc  
/process
```

You see the following message at the top of the screen:

```
...skipping
```

If the pattern is found, it is displayed two lines below this message. If the pattern is not found, "Pattern not found" is displayed.

If you are looking at a file with **more** and decide that you want to edit the file, you can invoke the **vi** editor by pressing **v**. Of course, you must have write permission on a file before you can edit it with **vi** or any other text editor. To display the file's contents, you only need read permission.

You will often use **more** in pipes. For example, **more** is useful when you want to list the contents of a directory in long format. Enter the following command to display a long listing of the contents of */bin*, one screenful at a time:

```
l /bin | more
```

(For more information on pipes, refer to Chapter 2 of this tutorial.)

The **head** and **tail** commands display the beginning and the end of a file, respectively. With no options, they display the first or last 10 lines. Enter the following command to display the last 10 lines of */etc/rc*:

```
tail /etc/rc
```

You can specify exactly how many lines you want displayed. Enter the following command to display the first 20 lines of */etc/rc*:

```
head -20 /etc/rc
```

Enter the following command to display the last 20 lines of */etc/rc*:

```
tail -20 /etc/rc
```

The **cat** command also displays the contents of a file. Unlike **more**, **cat** continuously scrolls the file until you stop the scroll with **Ctrl-s**. **Ctrl-q** continues scrolling. Scrolling stops automatically when the end of the file is reached. To stop scrolling before the end of the file, press **INTERRUPT**, which is the **Del** key on most keyboards.

Enter the following command to display the contents of */etc/termcap*. Use **Ctrl-s** and **Ctrl-q** to stop and start the scrolling and **INTERRUPT** to halt the scrolling before the end of the file is reached:

```
cat /etc/termcap
```

4

4.5.2 Deleting Files

The **rm** command is used to delete files. We have used it throughout this chapter to delete various files. Use **cd** to change to your home directory and enter the following command to create three new files:

```
touch tempfile1 tempfile2 tempfile3
```

Delete *tempfile3* by entering the following command:

```
rm tempfile3
```

The **-i** option allows you to remove files interactively by asking you if you really want to delete each of the files specified on the command line. If you press **y** followed by a **RETURN**, the given file is removed. If you press **n**, the file is left untouched. This option is useful when removing files from a directory that contains many files. It helps you avoid erasing files accidentally that you really want to keep.

Experiment with the **-i** option by entering the following command:

```
rm -i tempfile1 tempfile2
```

Note that you can place several filenames on the **rm** command line. This is true for most XENIX commands. You can also use wildcard characters. For example, instead of entering the above command, you could enter the following:

```
rm -i tempfile*
```

(The use of wildcard characters on the XENIX command line is discussed in Chapter 2 of this tutorial.)

4.5.3 Combining Files

In addition to displaying files, the **cat** command can be used to combine several existing files into a single new file. This is done by redirecting the output of **cat** into a new file. The greater-than sign (**>**) is used for redirection. If the new file does not exist, it is created automatically. (If you are not familiar with redirection, see Chapter 2 of this tutorial.)

Use **cd** to move to your home directory and enter the following command to combine */etc/motd* and */etc/rc* into a file named *catfile*:

```
cat /etc/motd /etc/rc > catfile
```

Now display the contents of the new file *catfile* with the **more** command:

```
more catfile
```

The symbol **>>** can be used with **cat** to *append* one file to the end of another file. For example, to append the contents of */etc/motd* to *catfile*, enter the following command:

```
cat /etc/motd >> catfile
```

The contents of */etc/motd* should now be placed at the beginning and at the end of *catfile*. Verify this with the following *head* and *tail* commands:

```
head -20 catfile  
tail -20 catfile
```

4

4.5.4 Renaming Files

The **mv** command is used to move files around the XENIX filesystem and also to rename files. Use **cd** to move to your home directory. Rename *catfile*, created in section 4.5.3, to *catfile2*, by entering the following command:

```
mv catfile catfile2
```

After this move is completed, *catfile* no longer exists. The file *catfile2* exists in its place. Verify this by entering the following command:

```
lc
```

4.5.5 Moving Files

To move a file into another directory, give the name of the destination directory as the final name in the **mv** command. You do not need to specify the destination filename. For example, enter the following command to move *catfile2*, created in section 4.5.4, to the */tmp* directory:

```
mv $HOME/catfile2 /tmp
```

To be sure that *catfile2* is in */tmp* and not in the current directory, enter the following command:

```
lc . /tmp
```

(Remember that you can enter more than one argument on most XENIX command lines, and that the dot (.) stands for the current directory.)

4

Finally, move *catfile2* back to the current directory by entering the following command:

```
mv /tmp/catfile2 .
```

The **mv** command always checks to see if the last argument is the name of a directory. If it is, all files designated by filename arguments are moved into that directory. However, if you do not have write permission on the directory to which you are attempting to move files, the move fails.

4.5.6 Copying Files

The **cp** command is used to copy files. There are two forms of the **cp** command, one in which files are copied into a directory and another in which a file is copied to another file.

Use **cd** to change to your home directory. Then enter the following command to copy the contents of *catfile2*, created in section 4.5.4, to *catfile3*:

```
cp catfile2 catfile3
```

You now have two files with identical contents. To copy *catfile2* and *catfile3* to the */tmp* directory, enter the following command:

```
cp catfile2 catfile3 /tmp
```

This last command can be simplified by using a wildcard character:

```
cp catfile* /tmp
```

Like the **mv** command, **cp** always checks to see if the last argument is the name of a directory, and, if so, all files designated by filename arguments are copied into that directory. However, unlike the **mv** command, **cp** leaves the original file untouched. There should now be two copies of *catfile2* and *catfile3* on the system, one copy of each in the current directory and one copy of each in */tmp*.

4.5.7 Finding Files

A XENIX filesystem can contain thousands of files. Because of this, files can often get lost. The **find** command is used to search the filesystem for files. The command has the form:

```
find pathname -name filename -print
```

The *pathname* is the pathname of the directory that you want to search. The search is recursive; it starts at the directory named and searches downward through all files and subdirectories under the named directory.

The **-name** option indicates that you are searching for files that have a specific *filename*. The **-print** option indicates that you want to print the pathnames of all the files that match *filename* on your screen.

Enter the following command to search all directories and subdirectories for *catfile2*, the file created in section 4.5.4:

```
find / -name catfile2 -print
```

It may take a few minutes for this command to finish executing. The output of this **find** command should indicate that there are at least two occurrences of *catfile2*, one in */tmp* and one in your home directory. Remove *catfile2* and *catfile3* from */tmp* and from your home directory by entering the following command:

```
rm /tmp/catfile* $HOME/catfile*
```

4.6 Processing Text Files

XENIX includes a set of utilities that let you process information in text files. These utilities enable you to compare the contents of two files, sort files, search for patterns in files, and count the characters, words, and lines in files. These utilities are described below.

4.6.1 Comparing Files

The **diff** command allows you to compare the contents of two files and to print out those lines that differ between the files. To experiment with **diff**, use **vi** to create two files to compare. The files will be *men* and *women*. First **cd** to your home directory. Then enter the following command at the XENIX prompt:

```
vi men
```

When you are placed in **vi**, press the **i** key to enter Insert mode, and then type the following lines:

```
Now is the time for all good men to  
Come to the aid of their party.
```

Press **ESC** to return to Command mode and save *men* by entering **:w**. While still in Command mode, enter the following command to create *women*:

```
:n women
```

You see the following message:

```
"women" No such file or directory
```

You are then placed in *women*. Press **i** to enter Insert mode and then enter the following lines:

```
Now is the time for all good women to  
Come to the aid of their party.
```

Press **ESC** to return to Command mode, then **:x** to save *women* and leave **vi**. You have now created *men* and *women*.

Enter the following command to compare the contents of these two files:

```
diff men women
```

This **diff** command should produce the following output:

```
1c1
< Now is the time for all good men to
---
> Now is the time for all good women to
```

The lines displayed are the lines that differ from one another in the two files.

4.6.2 Sorting Files

One of the most useful file processing commands is **sort**. When used without options, **sort** alphabetizes lines in a file, starting with the leftmost character of each line. These sorted lines are then output to the screen, or to a file if redirection is used on the **sort** command line. This command does not affect the contents of the actual file.



Enter the following command to display an alphabetized list of all users who have system accounts:

```
sort /etc/passwd
```

The **sort** command is useful in pipes. Enter the following command to display an alphabetized list of users who are currently using the system:

```
who | sort
```

4.6.3 Searching for Patterns in a File

The **grep** command selects and extracts lines from a file, printing only those lines that match a given pattern. Enter the following command to print out the lines in */etc/passwd* that contain your login information. There will probably be only one such line:

```
grep login /etc/passwd
```


Be sure to replace *login* in this command with your login name. Your output should be similar to the following:

```
markt:0V/wE/utnUkpc:6005:104:Mark Taub, Docland:/u/markt:/bin/csh
```

Note that whenever wildcard characters are used to specify a **grep** search pattern, the pattern should be enclosed in single quotation marks ('). Note also that the search pattern is case sensitive. Searching for "joe" will not yield lines containing "Joe".

As another example, assume that you have a file named *phonelist* that contains a name followed by a phone number on each line. Assume also that there are several thousand lines in this list. You can use **grep** to find the phone number of someone named Joe, whose phone number prefix is 822, by entering the following command:

```
grep 'Joe' phonelist | grep '822-' > joes.number
```

The **grep** utility first finds all occurrences of lines containing the word "Joe" in the file *phonelist*. The output from this command is then filtered through another **grep** command, which searches for an "822-" prefix, thus removing any unwanted "Joes." Finally, assuming that a unique phone number for Joe exists with the "822-" prefix, that name and number are placed in the file *joes.number*.

Two other pattern searching utilities are available with XENIX. These are **egrep** and **fgrep**. Refer to **grep(C)** in the *XENIX User's Reference* for more information on these utilities.

4.6.4 Counting Words, Lines, and Characters

The **wc** utility is used to count words in a file. Words are presumed to be separated by punctuation, spaces, tabs, or newlines. In addition to counting words, **wc** counts characters and lines.

Use **cd** to change to your home directory. Then enter the following command to count the lines, words, and characters in the file *men*, created in section 4.6.1:

```
wc men
```

The output from this command should be the following:

```
2  16  68 men
```

The first number is the number of lines in *men*, the second number is the number of words and the third number is the number of characters. Remove *men* and *women* by entering the following command:

```
rm *men
```

To specify a count of characters, words, or lines only, you must use the **-c**, **-w**, or **-l** option, respectively. For example, enter the following command to count the number of users currently logged onto the system:

```
who | wc -l
```

The **who** command reports on who is using the system, one user per line. The **wc -l** command counts the number of lines reported by the **who** command. This is the number of users currently on the system.

4.7 Using File and Directory Permissions

The XENIX system allows the owner of a file or directory to restrict access to that file or directory. This is done with permission settings. Permissions on a file limit who can read, write and/or execute the files. Permissions on a directory limit who can **cd** to the directory, list the contents of the directory, and create and delete files in the directory.

To determine the permissions associated with a given file or directory, use the **l** command. Use **cd** to change to your home directory and then enter **l** to get a long listing of the files in this directory.

Permissions are indicated by the first 10 characters of the output of the `ls` command. The first character indicates the type of file and must be one of the following:

- Indicates an ordinary file.
- b Indicates a block special device such as a hard or floppy disk. Hard and floppy disks can be treated as both block and character special devices.
- c Indicates a character special device such as a lineprinter or terminal.
- d Indicates a directory.
- m Indicates a shared data file.
- n Indicates a name special file.
- p Indicates a named pipe.
- s Indicates a semaphore.

From left to right, the next nine characters are interpreted as three sets of three permissions. Each set of three indicates the following permissions:

- Owner permissions,
- Group permissions, and
- All other user permissions.

Within each set, the three characters indicate permission to read, to write, and to execute the file as a command, respectively. For a directory, “execute” permission means permission to search the directory for any files or directories.

Ordinary file permissions have the following meanings:

- | | |
|---|--------------------------------|
| r | The file is readable. |
| w | The file is writable. |
| x | The file is executable. |
| - | The permission is not granted. |

For directories, permissions have the following meanings:

- | | |
|---|---|
| r | Files can be listed in the directory; the directory must also have ‘x’ permission. |
| w | Files can be created or deleted in the directory. As with ‘r’, the directory itself must also have ‘x’ permission. |
| x | The directory can be searched. A directory must have ‘x’ permission before you can move to it with the cd command, access a file within it, or list the files in it. Remember that a user must have ‘x’ permission to do anything useful to the directory. |

4

The following are some typical directory permission combinations:

- | | |
|------------|--|
| d----- | No access at all. This is the mode that denies access to the directory to all users but the super user. |
| drwx----- | Limits access to the owner. The owner can list the contents of this directory and the files in it (if they have appropriate permissions), cd to the directory, and add files to, and delete files from, the directory. This is the typical permission for the owner of a directory. |
| drwxr-x--- | In addition to allowing the owner all of the above access permissions, this setting allows group members to list the contents of this directory and files within it and to cd to this directory. However, group members cannot create files in, or delete files from, this directory. This is the typical permission an owner gives to others who need access to files in his or her directory. |

`drwxr-x--x`

In addition to allowing the owner and the group all of the above access permissions, this setting allows users other than the owner or members of the group to **cd** to this directory. However, because the **r** is not set for others, other users cannot list the contents of this directory with any of the **ls** commands. This mode is rarely used, but it can be useful if you want to give someone access to a specific file in a directory without revealing the presence of other files in the directory.

The `/etc` directory contains files whose permissions vary. Examine the permissions of the files in this directory by entering the following command:

```
l /etc | more
```

4

4.7.1 Changing File Permissions

The **chmod** command changes the read, write, execute, and search permissions of a file or directory. It has the form:

chmod *instruction filename*

The *instruction* argument indicates which permissions you want to change for which class of users. There are three classes of users, and three levels of permissions. The users are specified as follows:

- u** User, the owner of the file or directory.
- g** Group, the group the owner of the file belongs to.
- o** Other, all users of the system who are not in **u** or **g**.
- a** All users of the system.

The permissions are specified as follows:

- r** Read, which allows permitted users to look at but not change or delete the file.

- w Write, which allows permitted users to change or even delete the file.
- x Execute, which allows permitted users to execute the file as a command.

Use **cd** to move to your home directory. Then enter the following command to create *tempfile*:

touch tempfile

The permissions on *tempfile* are probably:

-rw-r--r--

Verify this by entering the following command:

l tempfile

Enter the following command to give yourself (the file's owner) execute permissions on *tempfile*:

chmod u+x tempfile

Verify the permissions change with the **l** command. (Of course, since *tempfile* is neither a binary nor a script, having execute permission on it is meaningless.)

Enter the following command to give the group and other users write permission on *tempfile*:

chmod go+w tempfile

Verify the permissions change with the **l** command.

The **chmod** command also allows you to remove permissions. For example, enter the following command to prohibit others from writing to *tempfile*:

chmod o-w tempfile

Remove *tempfile* with the following command:

rm tempfile

4.7.2 Changing Directory Permissions

Directories also have an execute permission, even though they cannot be executed in the same way that a script or binary file can. For directories, the execute attribute is needed in order to do any useful work in a directory. Users who do not have execute permission for a directory cannot **cd** to the directory, list the names of files in the directory, or copy files to or from the directory.

The permissions on your home directory are probably set to the following:

```
drwxr-xr-x
```

Verify this by entering the following command:

```
l -d $HOME
```

4 You probably see output like the following:

```
drwxr-xr-x  4 markt  pub      240 Feb 10 09:09 /u/markt
```

This setting allows you, the directory's owner, to **cd** to the directory, to list the contents of the directory and of the files within it (if the file permissions also allow), and to create and delete files in the directory. This setting also allows members of the group and other users to **cd** to the directory, to list the directory's contents and also the contents of files within the directory, if file permissions allow.

To deny any useful access to others, enter the following command:

```
chmod o-x $HOME
```

Verify that the permissions were changed with the following command:

```
l -d $HOME
```

Your output should look like the following:

```
drwxr-xr--  4 markt  pub      240 Feb 10 09:09 /u/markt
```

Now, only you and members of the group have access to your directory. If you want to restore access to your home directory to other users, enter the following command:

```
chmod o+x $HOME
```


Chapter 5

Housekeeping

- 5.1 Introduction 5-1
- 5.2 Making Backups 5-1
 - 5.2.1 Formatting Diskettes and Tapes 5-2
 - 5.2.2 Using tar to Create Backups 5-2
 - 5.2.3 Listing the Contents of Backups 5-5
 - 5.2.4 Extracting Files from Backups 5-5
 - 5.2.5 Shorthand tar Notation 5-7
- 5.3 Copying Diskettes 5-9
- 5.4 Getting Status Information 5-10
 - 5.4.1 Finding Out Who Is on the System 5-11
 - 5.4.2 Determining Disk Usage 5-11
- 5.5 Controlling Processes 5-12
 - 5.5.1 Placing a Command in the Background 5-12
 - 5.5.2 Delaying the Execution of a Command 5-13
 - 5.5.3 Finding Out Which Processes are Running 5-15
 - 5.5.4 Killing a Process 5-16
- 5.6 Shell Programming 5-17

5.1 Introduction

This chapter explains how to perform “housekeeping” tasks on a XENIX system. Housekeeping tasks are maintenance tasks that you perform periodically, tasks that provide you with information about system resources, as well as tasks that let you operate more efficiently in the XENIX environment. This chapter explains how to perform the following housekeeping tasks:

- Create backups of valuable files and directories,
- Extract files from backup media,
- Make copies of floppy diskettes,
- Find out who is on the system,
- Determine how much disk space is used/free,
- Run a command in the background,
- Delay and kill the execution of commands,
- Use the shell programming language to automate tedious tasks.

This chapter is designed as a tutorial. The best way to use this chapter is to read it at your terminal, entering commands as instructed in the examples.

None of the commands described in this chapter is described in great detail. For a complete explanation of each command, refer to the *XENIX User's Reference*.

5.2 Making Backups

Backing up all the filesystems on a XENIX system is often the responsibility of the system administrator. However, individual users often find it useful to create periodic backups of the particular files with which they work. These backups are created with the **tar** command.

Floppy diskettes are the backup media used most often. Cartridge tapes are also used for backups. However, floppy diskettes and some cartridge tapes must be formatted before they can be used. The sections that follow explain how to format floppy diskettes and certain cartridge tapes and how to use **tar** to create backups.

5.2.1 Formatting Diskettes and Tapes

To format a 5.25 inch 360 kilobyte diskette (double-sided, double-density) in the first floppy drive, enter the following command:

```
format
```

You are instructed to insert a diskette into the drive and press **RETURN**.

If your first floppy drive is a 5.25 inch double-sided high-density drive, enter the following command to format a 1.2 megabyte diskette in it:

```
format /dev/rfd096ds15
```

To format a 3.5 inch 720K micro-floppy diskette in the first floppy drive, enter:

```
format /dev/rfd096ds9
```

By replacing the **0** that follows the **rfd** in these commands with a **1**, you can format diskettes in a second floppy drive on your system.

5

It is not necessary to format most cartridge tapes. However, cartridge tapes used with the *mini tape drives* must be formatted. To format one of these cartridges, enter:

```
format /dev/rctmini
```

5.2.2 Using tar to Create Backups

The **tar** command is used to create backups. The syntax of **tar** is:

```
tar [key] [files]
```

The *key* argument controls the actions of **tar**. The *files* argument specifies which files to back up.

The *keys* used most often are:

- c** Creates a backup.
- x** Extracts files from backup media.
- t** Lists the contents of backup media.
- v** Displays the name of each file being processed.
- f** Creates backups on a specified device.

Creating a Backup

The steps outlined below explain how to back up all of the files in your home directory onto floppy diskettes. Experiment with **tar** by following these steps.

To back up a different directory, merely **cd** to that directory and follow these same steps. To back up onto a tape, substitute the special device file associated with the tape, such as */dev/rctmini* or */dev/rct0*, for the floppy device file listed in these commands.

1. Log in on the console. This allows you to work within arm's reach of the floppy drive.
2. Determine how many floppy diskettes you need and format that many, using the **format** command as described in "Formatting Diskettes and Tapes" above. To determine how many diskettes to format, enter the following command:

du -a

5

Your output should look like the following:

```
12  ./1.intro.s
74  ./2.concepts.s
14  ./2.concepts.err
0   ./err
60  ./5.house.s
32  ./3.log.s
2   ./err
2   ./0.title
30  ./6.desk.s
112 ./4.files.s
12  ./4.files.err
4   ./3.log.err
356 .
```

The number at the bottom represents the total number of 512 byte blocks used by the files in the current directory. In this example, you need a total of 512 x 356 bytes, or roughly 183 kilobytes. You only need to format a single floppy disk to backup this directory.

- 
3. Enter the following command to back up all of the files in your home directory to 5.25 inch 360 kilobyte floppy diskettes in the first floppy drive:

```
tar cvf /dev/fd048ds9 .
```

If your floppy drive supports 5.25 inch 1.2 megabyte floppy diskettes, enter:

```
tar cvf /dev/fd096ds15 .
```

If **tar** requires more than 1 diskette, you are prompted to insert another “volume.” Insert another diskette if you see this prompt. The **tar** command is finished when the shell prompt reappears.

To make a backup of just a single file onto a 1.2 megabyte diskette, enter:

```
tar cvf /dev/fd096ds15 ./filename
```

Note that *filename* is preceded by a dot and a slash (./). This tells **tar** to treat *filename* as a “relative” rather than an “absolute” filename. (For more information, see **tar(C)**.)

tar will recreate, on the backup media, all subdirectories of the directory you are backing up. Thus, if you have a */bin* directory in your home directory, **tar** will create a backup of it, and all the files in it, on the backup media.

5.2.3 Listing the Contents of Backups

To list the contents of a 5.25 inch 360 kilobyte tar-floppy in the first floppy drive, enter:

```
tar tvf /dev/fd048ds9
```

To list the contents of a 5.25 inch 1.2 megabyte tar-floppy in the first floppy drive, enter:

```
tar tvf /dev/fd096ds15
```

To list the contents of a 3.5 inch tar micro-floppy in the first floppy drive, enter:

```
tar tvf /dev/fd096ds9
```

5

Experiment with this **tar** option by placing the backup of your home directory that you created in section 5.2.2 into the first floppy drive. Enter the appropriate command to list its contents.

5.2.4 Extracting Files from Backups

We recommend that you extract files from backup media into a temporary directory on the hard disk. Once in the temporary directory, you can use the **mv** command to move the extracted files to the correct location on the filesystem. The reason for using a temporary directory is that **tar** will overwrite any files on the hard disk that have the same name as the file being extracted from the backup media. This can result in files being overwritten accidentally.

To extract all of the tar-format files from a 5.25 inch 360 kilobyte diskette in the first floppy drive, enter:

```
tar xvf /dev/fd048ds9
```


XENIX Tutorial

To extract tar-format files from a 5.25 inch 1.2 megabyte floppy diskette in the first floppy drive, enter:

```
tar xvf /dev/fd096ds15
```

To extract a single file from a 1.2 megabyte floppy in the first floppy drive, enter:

```
tar xvf /dev/fd096ds15 ./filename
```

Note that *filename* is preceded by a dot followed by a slash (./). This assumes that *filename* was copied to the **tar** floppy diskette with a dot (.), as in the examples in section 5.2.2. When you copy files to a **tar** floppy with a dot, the ./ is prefixed to the filenames. Because you must enter a filename exactly as it appears on the floppy when extracting a file with **tar**, you must enter **./filename** if *filename* was copied to the **tar** floppy with a dot.

Experiment with these **tar** commands by placing the diskette containing the backup of your home directory that you created in section 5.2.2 into the first floppy drive. Follow these steps:

- 
1. Enter the following command to change to */tmp*:

```
cd /tmp
```

2. Create a subdirectory in */tmp* by entering:

```
mkdir login
```

Replace *login* with your login name.

3. Enter:

```
cd login
```

4. If you are a Bourne shell user, and if your first floppy drive is a 1.2 megabyte drive, experiment with extracting a single file by entering the following command to extract *.profile*:

```
tar xvf /dev/fd096ds15 ./profile
```

If you are a C shell user, enter:

```
tar xvf /dev/fd096ds15 ./login
```

If your floppy drive is not a 1.2 megabyte drive, enter the appropriate special device filename.

5. To check that the files were actually copied to the hard disk, enter:

```
lc -a
```

The **-a** option tells **lc** to show hidden files, those whose filenames begin with a dot (.).

6. To experiment with extracting all of the files on a **tar** floppy, enter the following command if your first floppy drive is a 1.2 megabyte drive:

```
tar xvf /dev/fd096ds15
```

If your floppy drive is not a 1.2 megabyte drive, enter the appropriate special device filename.

5

5.2.5 Shorthand tar Notation

XENIX provides shorthand notation for use with the **tar** command. This notation allows you to specify numbers in place of full special device filenames. The file */etc/default/tar* assigns numbers to the various floppy and tape devices. Enter the following to display the contents of */etc/default/tar*:

```
more /etc/default/tar
```

Your output should look similar to the following:

```
# device                block  size  tape
archive0=/dev/rfd048ds9  18     360   n
archive1=/dev/rfd148ds9  18     360   n
archive2=/dev/rfd096ds15 10    1200   n
archive3=/dev/rfd196ds15 10    1200   n
archive4=/dev/rfd096ds9   18     720   n
archive5=/dev/rfd196ds9   18     720   n
archive6=/dev/rfd0135ds18 18    1440   n
archive7=/dev/rfd1135ds18 18    1440   n
archive8=/dev/rct0        20       0    y
archive9=/dev/rctmini     20       0    y
# The default device...
archive=/dev/rfd096ds15  10    1200   n
```

This file assigns **0** to the first 360 kilobyte drive, **1** to the second 360 kilobyte drive, **2** to the first 1.2 megabyte drive, **3** to the second 1.2 megabyte drive, and so forth.

To copy all the files in the current directory to 5.25 inch 360 kilobyte diskettes in the first floppy drive, enter:

tar cv .

(The default device is device **0**. You do not have to specify the default device name in the command in order to use the default device.)

To copy all the files in the current directory to 5.25 inch 1.2 megabyte diskettes in the first floppy drive, enter:

tar cv2 .

To extract *filename* from a 3.5 inch 720 kilobyte tar micro-floppy in the first floppy drive, enter:

tar xv4 ./filename

Note that the version of */etc/default/tar* on your system may differ from that shown here. This is because your system administrator may have edited it. Before you use this shorthand notation, double-check the assignments in the */etc/default/tar* file on your system.

5.3 Copying Diskettes

To protect against the loss of data stored on floppy disks, you can use the **diskcp(C)** command to make copies of your floppy diskettes. We recommend that you use this command to make copies of your entire XENIX distribution.

You must copy information onto formatted disks. If you format floppies under XENIX, you can use them over again without reformatting. If you have disks that have been formatted under another operating system, you must reformat them under XENIX before you can use them to make copies of XENIX disks. Be aware that floppies formatted under some operating systems cannot be used under other operating systems, even with reformatting.

The **diskcp** command can format floppies before making copies. The following steps explain how to use **diskcp**:

1. Place the disk that you want to copy, the “source” floppy, in your primary floppy drive. If you created a backup of your home directory, as instructed in section 5.2.2, experiment with **diskcp** by using this backup. Place it in the floppy drive.
2. Place another floppy in the other drive. This floppy is the “target” disk. Note that any information already on the target disk will be destroyed.

If you have only one disk drive, leave the source floppy in the drive. It will be copied to the computer’s hard disk, and from there to the target floppy. You will be prompted to remove the source floppy and insert the target floppy.

3. To format the target floppy as a 360 kilobyte floppy before copying, enter the command:

diskcp -f

To format the target floppy as a 1.2 megabyte floppy before copying, enter the command:

diskcp -f -96



If you do not need to format the target floppy, and if the floppy you are copying is a 360 kilobyte floppy, enter:

diskcp

If the floppy is a 1.2 megabyte floppy, enter:

diskcp -96

If your computer has dual floppy drives, enter the following command to copy a 360 kilobyte source floppy directly on a formatted target floppy:

diskcp -d

4. Follow the instructions as they appear on your screen. Note that, with a single drive system, you are prompted to remove the source disk and insert the target disk.
5. If you made a copy of the backup of your home directory, place this floppy in the first floppy drive and verify that the files were copied correctly by entering:

tar tvf /dev/fd096ds15

If your floppy is a 360 kilobyte floppy, enter:

tar tvf /dev/fd048ds9

Note that you can use the shorthand **tar** notation in these commands, as explained in “Shorthand tar Notation” above.

5.4 Getting Status Information

Because XENIX is a large, self-contained computing environment, there are many things that you may want to find out about the system itself, such as who is logged in and how much disk space is available. The sections that follow explain how to do this.

5.4.1 Finding Out Who Is on the System

The **who** command lists the names, terminal line numbers, and login times of all users currently logged onto the system. Enter the following command to find out who is on your system:

who

Your output should look like the following:

```
arnold  tty1a  Apr7  10:02
daphne  tty1b  Apr7  07:47
elliott tty1c  Apr7  14:21
ellen   tty2a  Apr7  08:36
gus     tty2b  Apr7  09:55
adrian  tty2c  Apr7  14:21
```

The **finger** command can also be used to find out who is on the system. It provides more detailed information. To use it, simply enter **finger**.

5

5.4.2 Determining Disk Usage

The **df** command displays figures on disk free space. When used without options, it reports the number of free blocks and inodes in all the filesystems on your computer. A block is 512 bytes. Enter the following command to display disk free space figures:

df

Your output should look like the following:

```
/      (/dev/root ):  5956 blocks  1437 inodes
```

This means that in the */dev/root* filesystem, there are 5956 blocks and 1437 inodes free. 5956 blocks is roughly equivalent to 3 megabytes.

When used with the **-v** option, **df** reports on the percent of blocks used as well as the number of blocks used and free. Enter the following command:

```
df -v
```

Your output should look like the following:

Mount Dir	Filesystem	blocks	used	free	% used
/	/dev/root	80152	70192	9960	88%
/y	/dev/y	82194	34314	47880	42%
/u	/dev/u	50000	37840	12160	76%

This output indicates that on the */dev/root* filesystem 88% of the blocks, or 70192 blocks out of a total of 80152, are used. 9960 blocks are still free.

5.5 Controlling Processes

A command that is executing is considered a *process*. In XENIX, a user can run several processes at the same time, one in the *foreground* and several others in the *background*. The foreground process is the one that is currently executing on your terminal. This is the only process that can accept input from your keyboard. For instance, when you are editing with **vi**, the **vi** program is running as a foreground process.

Keyboard input cannot be sent to background processes. It is often useful to execute processes that are time consuming or require no keyboard input in the background. Controlling foreground and background processes is the subject of this section.

5.5.1 Placing a Command in the Background

Normally, commands sent from the keyboard are executed in strict sequence. One command must finish executing before the next command can begin. However, if you place a command in the background, you can continue to enter commands in the foreground, even if the background command is not finished executing.

To place a command in the background, put an ampersand (&) at the end of the command line. For example, enter the following command to create, and then count, the characters in a large file. Note that this command line is two lines long. This is made possible by placing the backslash (\) on the command line before pressing RETURN. The backslash tells the shell that the command line continues on the next line:

```
cat /etc/termcap /etc/termcap /etc/termcap > largefile; \
wc -c largefile > characters &
```

The output of the `wc` part of the command is redirected to *characters*. If you issued this command without redirecting the output, the output would print on your screen, no matter what else you might be doing. This can be very disruptive. Redirecting the output of a background command to a file is a simple way of avoiding such disruptions.

Use `cat` to display the contents of *characters*. When you are finished, use `rm` to remove both *largefile* and *characters*.

When commands are placed in the background, you cannot abort them by pressing the INTERRUPT key, as you can with foreground commands. You must use the `kill` command to abort a background process. This command is described in “Killing a Process,” below.



5.5.2 Delaying the Execution of a Command

In addition to putting commands in the background, XENIX allows you to delay command execution. This is done with the `at` command. This command allows you to set up a series of commands to be executed at a specified time in the future.

The `at` command accepts standard input. The simplest form of the command is:

```
at time day < file
```

The *file* argument is the name of the file that contains the command or commands to be executed. The *time* argument is the time of day, in digits, followed by “am” or “pm.” One- and two-digit numbers are interpreted as hours, three- and four-digit numbers as hours and minutes. You cannot use *time* arguments of more than four digits. The *day* argument is optional. It is either a month name followed by a day number, or a day of the week. If no *day* is specified, the command is executed the next time the specified *time* occurs.

XENIX Tutorial

For example, suppose that you have a large file that you want to print, but you don't want to do this during work hours because it will monopolize the printer for a long time. You could use **at** to print the file late at night, when nobody is in the office. To do so, first use **vi** to create a file containing the print command. Call the file *printfile*. This file might contain the following line:

```
lp filename
```

The *filename* argument is the name of the large file that you want to print.

After you create *printfile*, enter the following command:

```
at 11pm wed < printfile
```

There is no need to place this command in the background. Once you enter it and press **RETURN**, the XENIX prompt reappears. This causes the command in *printfile* to be executed at 11:00 p.m. on Wednesday.

Note that **at** is unaffected by logging out. To display a list of files that you have waiting to be processed with **at**, use the **at -l** command. This command also lists the following information:

5

- The file's ID number.
- The command invoking the file (**at** or **batch**).
- The date and time the file is processed.

To cancel an **at** command, first check the list of files you have waiting to be processed and note the file ID number. Then use the **at -r** command to remove the file or files from the list.

The **at -r** command has the form:

```
at -r ID-number
```

For example, the following command removes file number 504510300.a, canceling whatever commands were included in that file:

```
at -r 504510300.a
```

Note that a user can only cancel his or her own files.

The files `/usr/lib/cron/at.allow` and `/usr/lib/cron/at.deny` control who has access to the **at** command. On many systems, only the super user is allowed to use **at**. Contact your system administrator if you need to use the **at** command but are denied access to it.

5.5.3 Finding Out Which Processes are Running

The **ps** command stands for “processes status” and displays information about currently running processes. This information is crucial if you need to kill a background process.

To display information about commands that you currently have running, enter the following:

```
ps
```

Your output should look like the following:

```

      PID   TTY   TIME  COMMAND
      49    2a    0:28   sh
    11267    2a    0:00   ps
```

The PID column gives a unique process identification number that can be used to kill a particular process. The TTY column shows the terminal with which the process is associated. The TIME column shows the cumulative execution time for the process. The COMMAND column shows the actual command that is executing.

Enter the following command to display information about all the processes running on the system:

```
ps -e
```

To find out about the processes running on a terminal other than the one you are using, use the **-t** option and specify the terminal number. For example, to find out what processes are associated with terminal 2c, enter:

```
ps -t2c
```

5.5.4 Killing a Process

To stop execution of a foreground process, press your terminal's **INTER-
RUPT** key. This is often the **Del** key. Pressing this key will kill whatever foreground process is currently running. To kill all your processes executing in the background, enter the following command:

```
kill 0
```

To kill only a specified process executing in the background, you need to enter the following command:

```
kill signal_number process_ID_number
```

The *signal_number* is optional. It is sometimes needed to kill “stubborn” processes. Even stubborn processes can usually be killed with *signal_number* 9. Find out the *process_ID_number* with the **ps** command.

As an example, try killing the process associated with your shell. Note that when you log in to the system, you are placed in a shell. If you kill this shell process, you log yourself out. Enter **ps** and look at the process ID associated with either **sh** or **csh** in the COMMAND column. Suppose that this number is 4831. To kill your shell, enter the following command:

```
kill -9 4831
```

After entering this command, you see the login prompt again. Try it!

Note

Killing a process associated with the **vi** editor can result in unpredictable terminal behavior. Also, temporary files that are normally created when a command starts, and then deleted when the command finishes, can be left behind after a **kill** command. Temporary files are normally kept in the directory */tmp*. You should check this directory periodically and delete your old files.

5.6 Shell Programming

Both the Bourne shell and the C shell offer powerful programming features. If you have ever done batch programming in MS-DOS, you have some idea of what shell programming in XENIX is like. This section discusses the rudiments of shell programming. For a more complete discussion, see “The Shell” and “The C-Shell,” both in the *XENIX User’s Guide*.

In our discussion of placing a command in the background in section 5.5.1, you were instructed to enter the following command:

```
cat /etc/termcap /etc/termcap /etc/termcap > largefile; \
wc -c largefile > characters &
```

However, you could have placed these commands in a file and executed the file.

Try it by creating a file called *command.file* with **vi**. Place the following lines in *command.file*:

```
cat /etc/termcap > largefile
cat /etc/termcap >> largefile
cat /etc/termcap >> largefile
wc -c largefile > characters
```

After placing these four lines in *command.file*, type **:x** to save it and quit **vi**. Now you must enter the following command to make *command.file* executable:

```
chmod +x command.file
```

Finally, execute *command.file* in the background by entering the following command:

```
command.file &
```

After a few moments, enter the following command to verify that *command.file* executed correctly:

```
cat characters
```

XENIX Tutorial

In fact, using some of the more sophisticated shell programming features that let you control the flow of a program, you could have written *command.file* as follows:

```
for name in /etc/termcap
do
    cat $name $name $name > largefile
done
wc -c largefile > characters
```

Whenever you place XENIX commands in a file, always remember to use the **chmod** command to make the file executable. (The **chmod** command is discussed in “Using File and Directory Permissions” in Chapter 4 of this tutorial.)



Chapter 6

XENIX Desktop Utilities

- 6.1 Introduction 6-1
- 6.2 Using the System Clock and Calendar 6-1
 - 6.2.1 Finding Out the Date and Time 6-1
 - 6.2.2 Displaying a Calendar 6-2
- 6.3 Using the Mail Service 6-2
 - 6.3.1 Sending Mail 6-2
 - 6.3.2 Receiving Mail 6-4
 - 6.3.3 Writing to a Terminal 6-6
- 6.4 Using the Automatic Reminder Service 6-7
- 6.5 Using the Calculator 6-8

6.1 Introduction

The XENIX operating system includes a series of “desktop” utilities, programs that help you organize your work environment, and programs that allow you to communicate with other users on the system. This chapter describes these utilities, explaining how you can:

- Display the date, time and a calendar,
- Communicate with other users on the system,
- Use the system’s automatic reminder service,
- Use the system’s interactive calculator.

This chapter is designed as a tutorial. The best way to use this chapter is to read it at your terminal, entering commands as instructed in the examples.

None of the commands described in this chapter is described in great detail. For a complete explanation of each command, refer to the *XENIX User’s Reference*.

6.2 Using the System Clock and Calendar

XENIX includes commands that display the date and time, as well as commands that display a calendar for virtually any month or year that you choose. The following sections explain these commands.

6

6.2.1 Finding Out the Date and Time

The **date** command displays the date and time. Enter:

date

Your output should look like the following:

```
Mon Jan 25 08:26:13 PST 1988
```


6.2.2 Displaying a Calendar

The **cal** command displays the calendar of any month or year that you specify. For example, to display the calendar for March 1952, enter:

```
cal mar 1952
```

The result is:

```
March 1952

S M Tu W Th F S
          1
2 3 4 5 6 7 8
9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30 31
```

The most common month abbreviations are accepted. The month can also be expressed as a digit. To display the calendar for an entire year, leave out the month. The year must always be expressed in full. The command **cal 88** displays the calendar for the year 88, not 1988.

6

6.3 Using the Mail Service

Several programs allow you to communicate with other users on a XENIX system. Two of the most useful are **mail** and **write**. The **mail** program allows you to send a message to a user's system "mailbox." The **write** program allows you to write a message directly to a user's terminal, if the user is logged into the system. Both of these programs are described below.

6.3.1 Sending Mail

The **mail** program is a system-wide facility that permits you to exchange mail with other users. Experiment with **mail** by sending a message to yourself. To do so, enter the following command:

```
mail login
```

Be sure to replace *login* with your login name.

Depending on how your system administrator has configured your mail system, you may see the following prompt:

Subject :

If you see this prompt, enter a short description of the message to follow. In this case, enter **test**.

You can now enter your message. When you are finished, press **Ctrl-d** to terminate message entry and mail your message. However, an alternative method is available for composing a message. If you enter **~v**, (a tilde, followed by a **v**) you are placed in **vi**. Once in **vi**, you can compose your message, using all the features available in **vi**. This method of composing a message is much more flexible than the other, as it allows you to correct mistakes in your message before you send it. Correcting mistakes in a message produced only with **mail** often results in control characters cluttering up the message.

If you use **vi** to compose your message, enter **:x** when you are finished, then **Ctrl-d** to terminate message entry. Again, depending on how your mail system is configured, you may or may not see the following prompt:

Cc :

If you do see this prompt, enter the names of those users who should receive “carbon copies” of this message. It is often convenient to **Cc**: yourself. Since this is a test message to yourself, just press **RETURN**. You are now returned to the XENIX prompt.

Often you will want to send a text file to other users on the system. You can use the XENIX redirection facility to do this. Suppose that the file you want to send is named *schedule*, that it is in the current directory, and that you want to send it to users Naomi and Bea. To do so, enter the following command at the XENIX prompt:

```
mail naomi bea < schedule
```

Sending files this way is fast because you do not have to enter **mail**.

The **mail** facility has many options. These options allow you to include already composed messages in a mailing, to enclose a message you are responding to in your reply, to create mail aliases to send messages to several users at once, and so forth. These options are discussed in detail in “mail” in the *XENIX User's Guide*.



6.3.2 Receiving Mail

When you log in, you may see the following the message:

```
you have mail
```

To read your mail, enter:

mail

A list of message headers is displayed, each with a number in front of it. The list should look like the following:

```
1 john Wed Sep 21 09:21 26/782 "Notice"
2 sam Tue Sep 20 22:55 6/83 "Meeting"
3 tom Mon Sep 19 01:23 6/84 "Invite"
```

Reading from left to right, the headers tell you who the message is from, the date and time it was sent, the number of lines and characters in the message, and the message's subject.

6

To read a message, simply enter the number of the message you want to read, then press **RETURN**. For example, to read the message from sam, enter **2** and then press **RETURN**. Read the message that you sent to yourself, as described in the previous section.

There are several things you can do with your **mail** messages after you read them. You can delete them, save them, and/or respond to them.

To delete a message, press **d**, the message number and then **RETURN**. To save a message in your mailbox, press **ho**, the message number and then **RETURN**. To save a message in *filename* in the current directory, press **s**, the message number, type *filename* and press **RETURN**. If *filename* does not exist, **mail** creates it.

Two commands are available for responding to mail. These are the **r** and **R** commands. If you press **r**, followed by the number of a message, followed by **RETURN**, followed by a response, your response is sent to the author of the message. If you press **R**, followed by the number of a message, followed by **RETURN**, followed by a response, your response is sent to the author of the message plus all users who were on the **Cc:** list of the original message.

After reading a message, you might want to list your message headers again. Do so by entering **h** followed by **RETURN**. If you have more messages than will display on the screen, enter **h+** followed by **RETURN**. This will cause the next 18 message headers to display. To display the previous 18 message headers, enter **h-**.

Note that you can send mail from within the **mail** program. To send mail to a user named “joe” from within **mail**, simply enter the following command:

mail joe

Then follow the steps outlined above in “Sending Mail” to send a message to joe.

To quit **mail**, enter **q** followed by **RETURN**.

Respond to the message that you sent yourself by doing the following:

1. Enter mail by typing **mail**.
2. Enter **rnumber**, where *number* is the number of the message that you sent to yourself.
3. Press **RETURN**.
4. Compose your response. Remember that you can enter **~v** to compose your response in **vi**.
5. Press **Ctrl-d** to terminate your response. If you compose your response in **vi**, you will have to press **:x** to leave **vi** and then **Ctrl-d**.
6. Press **RETURN** to send the response.
7. Type **restart**. This will cause **mail** to display any messages that were sent to you while you were in **mail**.
8. You should see your response to the message you sent yourself. Press the number that corresponds to this response to view it.

9. When you are finished looking at your messages, press **q** to leave mail.

6.3.3 Writing to a Terminal

XENIX includes a program that allows you to send messages directly to another user's terminal, if the user is logged into the system. This is the **write** program. To write to joe's terminal, enter:

```
write joe
```

After you execute this command by pressing **RETURN**, joe sees a message like the following on his terminal:

```
Message from login tty012...
```

The *login* in this message is your login name. To respond, joe enters:

```
write login
```

Again, *login* is replaced by your login name.

6 From this point on, each line that you enter is displayed both on your own terminal screen and on joe's. Each line that joe enters is displayed on both his screen and yours. To terminate the writing of text to joe, enter **Ctrl-d** alone on a line. Joe has to do the same to terminate his **write** session with you.

A typical procedure for coordinating communication in a two-way **write** is for each party to end each line with a distinctive signal, normally (o) for "over." The last line of a message is often followed by (oo) for "over and out."

Experiment with **write** by sending a message to yourself. Do so by entering the following command:

```
write login
```

Replace *login* with your login name. You should see a message like the following:

Message from *login* tty~~nn~~...

Now simply enter your message. Since you are writing to yourself, everything you enter appears on *your* screen twice.

For example, your **write** session might look like the following:

```
Hello Mark o
Hello Mark o
Remember, we have a meeting at 12:00. o
Remember, we have a meeting at 12:00. o
Right, see you there. oo
Right, see you there. oo
```

Press **Ctrl-d** to terminate the **write** session.

6.4 Using the Automatic Reminder Service

An automatic reminder service is available to all XENIX users. Once each day, XENIX automatically searches each user's home directory for a file named *calendar*, the contents of which might look like the following:

```
1/23 David's wedding
2/9 Mira's birthday
3/30 Paul's birthday
4/27 Meeting at 2:00
9/1 Karen's birthday
10/3 License renewal
```

Each line of *calendar* is examined. Lines containing today's and tomorrow's dates are extracted and mailed to you. To look at these reminders, you must invoke **mail**.

The file *calendar* is not created for you automatically. You have to create it yourself if you want to use this reminder service. It must be in your home directory.

Use **vi** or any other XENIX text editor to create and edit *calendar*. Be sure to place each date/event entry on a separate line. Dates can be specified in a variety of formats. Any of the following is acceptable:

```
9/7
Sep. 7
september 7
```

6.5 Using the Calculator

The **bc** command invokes an interactive desktop calculator that can be used like a hand-held calculator. A typical session with **bc** is shown below. Note that the session is begun by entering **bc** at the XENIX prompt and ended by entering **quit** on a line by itself. Comments explain what action is performed after each input line.

Action	Comment
bc	Activate bc
123.456789+987.654321	Add and output
1111.111110	
9.0000000-9.0000001	Subtract and output
.0000001	
54/8	Divide and output
3	
1.12345678934*2.3	Note precision
2.58395061548	
19%4	Find remainder
3	
3^4	Exponentiation
31	
2/1*2	Note precedence
4	
2/(1*2)	Note precedence again
1	
x=46.5	Assign value to x
y=52.5	Assign value to y
x+y+1.0000	Add and output
100.0000	
obase=16	Set hex output base
15	Convert to hex
F	
16	Convert to hex
10	
54	Convert to hex
40	
255	Convert to hex
FF	
256	Convert to hex
100	
512	Convert to hex
200	
quit	Must type whole word

Also available are scaling, function definition, and programming statements much like those in the C programming language. Other features include assignment to named registers and subroutine calling. For more information, see “bc: A Calculator,” in the *XENIX User's Guide*.

Index

& *See* Ampersand (&)
* *See* Asterisk (*)
[] *See* Brackets ([])
- *See* Dash (-)
> *See* Greater-than sign (>)
. *See* Period (.)
? *See* Question mark (?)
/ *See* Slash (/)

A

-a option, described 2-12
Absolute pathname. *See* Pathname
Adding 6-9
Ampersand (&)
 background command 2-11
 background process 5-13
Append
 files 2-14, 4-17
 text. *See* Output
Argument, option. *See* Switch
Asterisk (*)
 filename, not used in 2-6
 filename wildcard 2-9
 pattern matching functions 2-9
at command 5-13
at -r command 5-14

B

Background
 command 2-11
 process 5-12, 5-16
 ampersand (&) operator 5-13
Backups
 1.2 megabyte diskettes 5-4
 360 kilobyte diskettes 5-4
 creating 5-1, 5-4
 extracting tar-format files 5-6
 floppy disks, how many to format 5-3
 listing the contents of a tar floppy 5-5
 shorthand tar notation 5-7

Backups (*continued*)
 tar command 5-2
 using 720K micro-floppies 5-5
Batch processing 2-11
bc command, calculating 6-8
/bin directory, contents of 2-7
Binary file. *See* File
BKSP key
 command-line buffer editing 2-11
Block special device 4-24
Bourne shell 2-1
Brackets ([])
 filename, not used in 2-6
 pattern-matching functions 2-10

C

cal command 6-2
Calculating, example 6-8
calendar command 6-7
cancel command. *See* lp command
cat
 command 4-15, 5-13
 files, combining 4-16
cd command
 directory change 2-4
 use 4-4
Change
 directory 2-4, 4-3
 terminal types 3-6
Character
 counting 4-22
 special device 4-24
chmod command 4-26, 4-28, 5-17
Command
 at command 5-13
 at -r command 5-14
 background submittal 2-11
 batch processing 2-11
 bc command 6-8
 cal command 6-2
 calendar command 6-7
 cat command 4-15, 4-17, 5-13
 cd command 4-4
 chmod command 5-17
 copy command 4-6
 cp command 4-18
 dash (-) use 2-6
 date command 6-1
 df command 5-11

Index

Command (*continued*)

- diff command 4-20
- diskcp command 5-9
- du command 3-8, 5-3
- executing 2-10
 - sequence 5-12
- find command 4-19
- finger command 5-11
- format command 5-2
- grep command 4-21
- head command 4-15
- kill command 5-16
- l command 4-3, 4-15
- lc command 3-8, 4-2, 4-17
- lf command 4-2
- line. *See* Command line
- lowercase letters 2-12
- mkdir command 4-4
- more command 4-14, 4-15
- multiple commands 2-11
- mv command 4-17
- options 2-12
- passwd command 3-3
- program, invoking 2-10
- ps command 5-15
- rm command 4-16, 4-19
- rmdir command 4-5
- sort command 4-21
- su command 3-6
- syntax 2-12
- tail command 4-15
- tar command 5-2, 5-5
- wc command 4-22
- who command 4-21, 4-23, 5-11

Command line

- ampersand (&) effect 2-11
- buffer defined 2-11
- defined 2-11
- entry 3-7
- erasure 3-8
- interpretation 2-11
- multiple commands 2-11
- RETURN key effect 3-7

Control characters

- filename use restrictions 2-6

Copy

- command 4-6
- directories 4-6
- files 4-18
- floppy diskettes 5-9
- See* cp command

Counting, wc command 4-22

cp command 4-18

Create

Create (*continued*)

backups

- 1.2 megabyte diskettes 5-4
- 360 kilobyte diskettes 5-4
- 720K micro-floppies 5-5
- extracting tar-format files 5-6
- floppy disks, how many to format 5-3
- listing the contents of a tar floppy 5-5
- several volumes 5-4
- shorthand tar notation 5-7
- tar command 5-2

directories 4-4

C-shell 2-1

Ctrl-d, end-of-file 3-3

Ctrl-u

- command-line buffer editing 2-11
- line kill 3-8

Current directory

- changing 4-3
- description 4-3
- printing 4-2
- shorthand name 2-8

D

d command, mail, message delete 6-4

Dash (-)

- command option use 2-6
- filename, not used in 2-6

Dash (-), permission

- denial notation 4-25
- ordinary file notation 4-24

date command 6-1

Delete

- d command 6-4
- file 4-16

/dev directory, contents of 2-7

Device pathname 2-7

Device special file. *See* Special device file

df command 5-11

diff command 4-20

Directory

See Filename

- access permission. *See* Permission

- /bin. *See* /bin directory

- changing 4-3

- command. *See* cd command

- contents of 2-4

- copying 4-6

- creating 4-4

- current directory. *See* Current directory

- described 2-4

Directory (*continued*)

- /dev. *See* /dev directory
 - diagram 2-5
 - filename
 - required 2-6
 - unique to directory 2-6
 - /lib. *See* /lib directory
 - long listing 4-3
 - parent directory. *See* Parent directory
 - pathname required 2-7
 - permission notation 4-24
 - protection 2-4
 - removing 4-5
 - renaming 4-6
 - search permission. *See* Permission
 - /tmp directory 5-16
 - user control 2-5
 - /usr. *See* /usr directory
 - working directory. *See* Current directory
- diskcp 5-9
- copying a 1.2 megabyte floppy 5-10
 - copying a 360 kilobyte floppy 5-10
 - formatting a 1.2 megabyte floppy 5-9
 - formatting a 360 kilobyte floppy 5-9
 - using two floppy drives 5-10
- Displaying a file 4-14
- Dividing 6-9
- du command 3-8, 5-3

E

- Editing files 4-7
- Exponentiation 6-9

F

File

- access
 - control 2-3
 - permissions. *See* Permission
- adding. *See* Create
- alphabetizing. *See* sorting
- appending 4-17
- attributes 2-2
- binary file 2-2
- change time 2-2
- combining 4-16
- composition 2-2
- copying 4-18

File (*continued*)

- creating
 - permission. *See* Permission
 - read permission control 2-4
 - write permission control 2-4
 - defined 2-2
 - deleting 4-16
 - write permission control 2-4
 - directory. *See* Directory
 - displaying 4-14, 4-15
 - editing 4-7
 - filename. *See* Filename
 - group 2-2
 - manipulating 4-13
 - moving 4-18
 - name. *See* Filename
 - owner 2-2
 - pathname required 2-7
 - pattern search
 - See* Pattern matching facility
 - permissions
 - changing 4-26
 - l command 4-24
 - listing 4-3
 - See* Permission
 - protection 2-3
 - removing 4-16
 - renaming 4-17
 - scratch file directory 2-8
 - size in bytes 2-2
 - sorting 4-21
 - special file. *See* Special device file
 - temporary file. *See* Temporary file
 - types 2-2
- Filename
- asterisk (*) wildcard 2-9
 - characters' use restrictions 2-6
 - described 2-6
 - example 2-8
 - long listing 4-3
 - question mark (?) use 2-10
 - required 2-2, 2-6
 - unique to directory 2-6
- Filesystem, diagram 2-5
- find command 4-19
- Finding a file 4-19
- finger command 5-11
- Floppy diskette, copying. *See* diskcp 5-9
- Foreground process 5-16
- format command 5-2
- Formatting media
 - 1.2 megabyte floppy diskettes 5-2
 - 360 kilobyte diskettes 5-2
 - 720K micro-floppy diskettes 5-2

Index

Formatting media (*continued*)

- format command 5-2
- mini tape cartridges 5-2
- Full pathname. *See* Pathname, absolute

G

Greater-than sign (>)

- file combination 4-16
- output redirection 2-14
- grep command 4-21
- Group permission. *See* Permission

H

- head command 4-15
- Home directory 2-1, 4-4

I

Inode

- number 2-4
- required for file 2-2

Input

- keyboard origin 2-13
- redirecting. *See* Redirect
- terminating 3-3

INTERLUPT key

- command-line buffer cancellation 2-11
- foreground process killing 5-16

K

- Kernel buffers, maximum keyboard input 3-8
- kill command 5-16
- Killing a process 5-16

L

- l command 4-3, 4-15
- lc command 3-8, 4-2, 4-17
- Less-than symbol (<)

Less-than symbol (<) (*continued*)

- input redirection 2-14
- lf command 4-2
- /lib directory, contents of 2-7
- Line, counting. *See* wc command
- Lineprinter
 - command 4-10
 - options 4-11
 - See* lp command
- Link, long listing 4-3
- Listing. *See* l command
- Logging in, login procedure 3-1
- Logging out, logout procedure 3-3
- Login procedure 3-2
- lp command
 - cancel 4-12
 - d option 4-11
 - lpstat command 4-12
 - n option 4-11
 - pipes, use with 4-11
 - pr command, use with 4-11
- lpstat command 4-12
 - p option 4-13

M

Mail

- composing a message 6-3
- d command 6-4
- exit, q command 6-6
- h command 6-5
- ho command 6-4
- message
 - deleting 6-4
 - displaying 6-4
 - listing 6-5
 - saving 6-4
- prompt 6-4
- q command, exit 6-6
- quitting 6-5
- reading 6-4
- reminder service 6-7
- responding to a message 6-5
- s command 6-4
- sending 6-2
- using vi 6-3
- Make directory. *See* mkdir command
- Mini tape cartridges, formatting 5-2
- mkdir command 4-4
- more command 4-14, 4-15
- Move. *See* mv command
- mv command 4-17

mv command 4-17 (*continued*)
 directory moving 4-6
 file moving 4-18

N

Name special file 4-24
 Named pipe 4-24

O

o, message end 6-6
 oo, message end 6-6
 Option
 grouping 2-12
 multiple options
 grouping 2-12
 separate listing 2-12
 regulations 2-12
 Ordinary file. *See* File
 Output
 appending
 procedure 2-14
 symbol (>>) 2-14
 control 3-8
 redirecting 4-16
 See Redirect
 resuming 3-8
 stopping 3-8
 terminal screen destination 2-13

P

Parent directory
 described 2-8
 shorthand name 2-8
 passwd command 3-3
 Password 3-4
 changing 3-4
 composition 3-4
 user accounts 3-5
 Pathname
 absolute
 example 2-7
 required 2-7
 slash (/) significance 2-7
 defined 2-6

Pathname (*continued*)

 full 2-7
 relative
 defined 2-7
 example 2-8
 structure 2-6
 Pattern matching facility
 cancelling 2-10
 characters 2-9
 described 2-8
 grep command 4-21
 Period (.)
 filename use 2-6
 working directory, changing 4-4
 Permission
 block special device notation 4-24
 denial notation 4-25
 directory permissions
 assignment 2-4
 changing 4-26
 combinations designated 4-25
 creating a file 4-25
 deleting a file 4-25
 file listing notation 4-25
 notation 4-24
 search notation 4-25
 search permission 4-28
 write permission 2-4
 execute notation 4-25
 file permissions
 creating a file 4-25
 deleting a file 4-25
 denial notation 4-25
 execute permission 4-25
 file listing notation 4-25
 file protection 2-3
 notation 4-24
 read notation 4-25
 required 2-2
 write notation 4-25
 notation 4-24
 read notation 4-25
 search notation 4-25
 symbols designated 4-24
 types 4-24
 write notation 4-25
 PID, process identification number 5-15
 Pipe
 function 2-14
 symbol (|) 2-15
 used with more command 4-15
 Pipeline, defined 2-14
 Print working directory. *See* pwd command
 Printing 4-10

Index

Process

- background. *See* Background
- foreground 5-16
- status 5-15

- Prompt character 3-3

- ps command 5-15

- pwd command 4-2

Q

- Question mark (?)

- filename, not used in 2-6

- pattern-matching functions 2-10

- single character representation 2-10

- Quotation mark, single (')

- filename, not used in 2-6

- pattern matching, cancelling 2-10

- Quotation marks, double ("")

- filename, not used in 2-6

R

- r character, read permission notation 4-25

- Redirect

- input

- procedure 2-14

- symbol (<) 2-14

- output

- procedure 2-13, 4-16

- symbol (>) 2-14

- Relative pathname. *See* Pathname

- Reminder service

- automatic 6-7

- Remove

- directory

- rmdir command 4-5

- file 4-16

- RETURN key

- command execution 3-7

- command-line buffer submittal 2-11

- mail, message display 6-4

- rm command 4-16, 4-19

- rmdir command 4-5

S

- Screen

- scrolling 3-8

- terminal 3-8

- Scrolling

- control 3-8

- screen, stopping 3-8

- Search

- file, search for 4-19

- permission. *See* Permission

- strings, example 2-12

- Security, user accounts 3-5

- Semaphore 4-24

- Semicolon (;), command separation 2-11

- Shared data file 4-24

- Shell, command interpretation 2-11

- Single quotation mark

- See* Quotation mark, single (')

- Slash (/), pathname significance 2-6

- sort command 4-21

- Special characters

- designated 2-9

- pattern matching 2-8

- Special device file, described 2-3

- Status

- command. *See* ps command

- information procedures 5-10

- su command 3-6

- Subdirectory 4-4

- Subtracting 6-9

- Super user account 2-2

- Switch. *See* Option

- System, basic concepts 2-1

T

- tail command 4-15

- tar command 5-2, 5-5

- Temporary file

- directory (/tmp) 5-16

- kill command warning 5-16

- Terminal

- changing 3-6

- screen, output 3-8

- writing to. *See* write command

- Text editing. *See* vi

- /tmp directory 2-7, 5-16

Type-ahead 3-8

U

User

- classification 4-26
- permission. *See* Permission

User account 2-1

- group identification 2-1
- home directory 2-1
- login name 2-1
- login shell 2-1
- password 2-1

/usr directory, contents of 2-7

/usr/bin directory, contents of 2-7

/usr/lib directory, contents of 2-7

V

-v option 2-12

Vertical bar (|), pipe symbol 2-15

vi

- Command mode 4-7

- cursor-movement keys 4-8

- deleting text 4-8

- entering text 4-7

- exiting 4-9

- Insert mode 4-7

- inserting text 4-9

- moving the cursor 4-8

- saving a file 4-9

- using 4-7

Visual shell 2-1

W

w character

- directory permission notation 4-25

- file permission, write notation 4-25

wc command 4-22

- word count 2-15

who command 4-21, 4-23, 5-11

- logged in users list 2-15

Words, counting. *See* wc command

write command 6-6

- message end 6-6

X

x character

- directory permission search 4-25

- file permission, execute notation 4-25

512-210-060
24837

XENIX® Tutorial