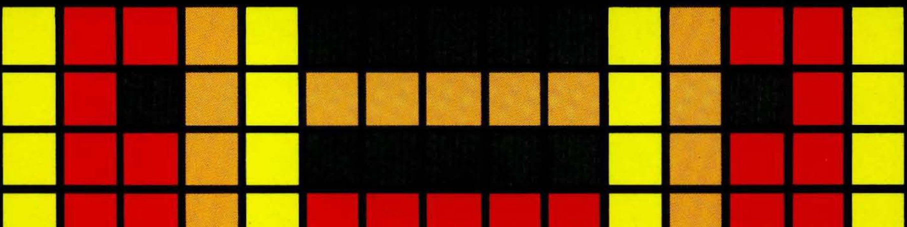
XENIX™ USER'S HANDBOOK



WEBER SYSTEMS, INC. STAFF

XENIX™ User's Handbook

XENIX™ User's Handbook

by
WSI Staff

Weber Systems, Inc.
Cleveland, Ohio

Published by:
Weber Systems, Inc.
8437 Mayfield Road
Cleveland, Ohio 44026

For information translations and book distributors outside of the United on States, please contact WSI at the above address.

Library of Congress Cataloging in Publication
Main entry under title:

XENIX user's handbook

Includes index
1. XENIX (Computer program) I. Weber Systems, Inc.
QA76.6.46 1984 001.64'2 84-050833
ISBN 0-938862-44-8

Typesetting and Layout: Tina Koran, Maria Stamoulis, Deborah McCollum, and Irma Schaeffer, Writer: David Sullivan.

Contents

INTRODUCTION

XENIX™ User's Handbook has a two-fold purpose.

- Serve as an instructional guide for the system manager to XENIX installation and upkeep.
- Serve as an introductory tutorial on XENIX system utilization for the beginning XENIX user.

The following system management topics are discussed in detail:

Installation
Configuration
Modification
System Security
System Maintenance
System Performance with Additional RAM

The user oriented topics discussed include the following:

File System
Text Editors
Electronic Mail
File Manipulation
Shell Programming
Communication Among UNIX Systems
Text Formatting

The chapters discussing system management tasks offer specific instructions on how to transfer a XENIX system from the distribution diskettes and get the system running. The chapters offering user tutorials include examples as an integral part of each discussion.

Chapter 1 of this book is meant to serve as an introduction to UNIX in general and XENIX in specific.

Chapters 2 through 7 are intended to aid the system manager. Chapter 2 covers installation of the XENIX system. Chapter 3 describes how to configure the newly installed system. Configuration tasks include adding peripheral hardware, such as terminals and printers to the system, and creating user accounts. Chapter 5 addresses system security. System maintenance tasks are detailed in chapter 6. The benefits of adding more RAM to the computer are discussed in chapter 7.

Chapters 8 through 12 are intended as an introduction to the XENIX operating system. Chapter 8 discusses general XENIX terminology and how to logon to the system. Chapter 9 describes the XENIX file system structure along with file handling utilities. Use of the XENIX text editors, **ed** and **vi,** is detailed in chapter 10. The **mail** utility is covered in chapter 11. The last chapter discusses several miscellaneous topics such as text formatting with the **nroff** utility, shell programming, file manipulation, and communications between UNIX systems.

Note that throughout this book XENIX utilities are presented in boldface. Example commands include a mixture of boldface and italics. The boldface portion of a command must be typed as it appears in the book. The italicized portion of a command must be replaced with an appropriate parameter, such as a file name.

Chapter 1 — An Overview

INTRODUCTION

In this chapter, we will present a general introduction to the UNIX operating system. A short history of UNIX is presented as well as some of the advantages and disadvantages of the system. Finally, we will introduce XENIX and compare it with the two most widely-used personal computer operating systems, CP/M and MS-DOS.

A BRIEF HISTORY OF UNIX

In the late 1960's, Bell Laboratories was partnered with the Massachusetts Institute of Technology (MIT) and General Electric in a project to develop the first interactive, multiuser/multitasking operating system for implementation on General Electric's 645 mainframe computer (GE is no longer a computer manufacturer).

An **operating system** is a set of routines that allow the user to communicate more easily with the computer hardware. Typically, the user never communicates directly with the machine. Instead, the user relays instructions to the operating system. The operating system then causes these demands to be carried out by the hardware and reports the results to the user. **Multiuser** refers to a system that allows several individuals to share the use of the computer. **Multitasking** capability enables the computer to appear as if it is working on several tasks at the same time. **Interactive** refers to a system that supports immediate dialogue with the user (as opposed to batch I/O operations involving devices such as card readers, magnetic tapes, etc.).

Although the resulting system, Multics, was a technical success in that it accomplished the project goals, the initial incarnation of Multics proved too slow and cumbersome to be an effective tool. In 1969, Bell Labs decided to pull out of the project.

Bereft of the capabilities of Multics, a young computer scientist, Ken Thompson, working in the computing science research department (Murray Hill, N.J.) of Bell Labs was forced to move his work to a more affordable minicomputer, the Digital Equipment Corporation (DEC) PDP-7™. Since at this time there was a dearth of software for the PDP-7, Thompson created not only an assembler, but a new file system and primitive operating system to control the system. In a pun on Multics, the system was dubbed UNIX by its lone inventor, Thompson. At this point, the UNIX operating system was written in assembly language and limited to use on the PDP-7. This version of UNIX could support only a single user.

The UNIX operating system got a great lift when Thompson teamed with Dennis Ritchie, a systems software engineer at Bell Labs. They developed the C programming language, and then used this language to rewrite the UNIX operating system. This action was taken in order to transport UNIX to a variety of machines. The idea of transporting a program from one machine to another is called **portability.** The first port to a significantly different machine occurred in 1977. Since then, UNIX has been ported to many different machines varying in size from the small IBM PC™ to the large Cray-II™ supercomputer.

Throughout the 1970's, AT&T did not aggressively market the UNIX operating system. However, they did grant academic licenses at minimal costs. UNIX became very popular in academic and research circles. These UNIX devotees added new features to their versions of UNIX. The Computer Science Department of the University of California at Berkeley created many such enhancements. There is even a version of UNIX called Berkeley UNIX. Also, AT&T includes many of the Berkeley enhancements in the newer versions of UNIX.

Finally, in the 1980's, AT&T started to take a more aggressive approach to marketing UNIX. Many software vendors and hardware manufacturers have made the UNIX available for a number of different computers.

GENERAL STRUCTURE

The general structure of the UNIX operating system is depicted in figure 1.1.

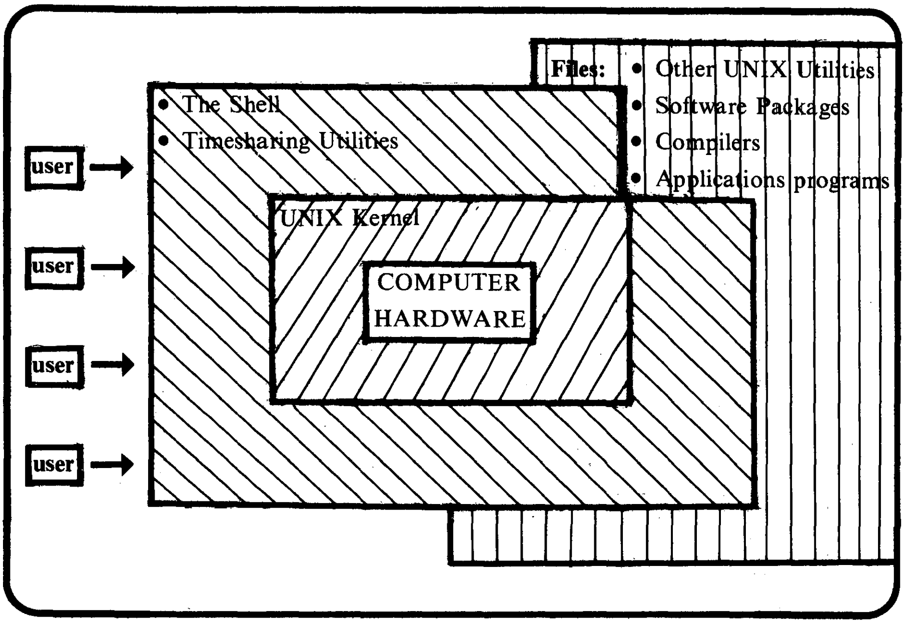


Figure 1.1. General structure of the UNIX operating system.

Note that the individual user never interfaces directly with the UNIX kernel or the computer hardware. All communication between the UNIX kernel and the shell takes place in the form of calls. This isolation of the user's application programs is part of what makes these programs portable to other UNIX systems.

The items stored in the files are accessible to the individual user (as long as he has the correct authorization) via the shell. Programs and software packages can be used just like the UNIX utilities once they have been installed into the file system.

Note that the major machine dependent functions are placed in the UNIX kernel. This fact helps make the entire UNIX system portable to various machines. Generally, only these machine dependent functions need be recoded in the new computer's native machine or assembly code.

Advantages of UNIX

We have already discussed one advantage of UNIX, its portability. Since hardware manufacturers obtain a license to use UNIX on their new products, they are spared the time and expense of developing a proprietary operating system. Furthermore, using UNIX as the operating system allows the manufacturer to introduce the new machine with a mature, tested operating system.

Software vendors can develop application packages using C and UNIX shell commands. These packages can be sold to run on a variety of machines that use the UNIX operating system with minimal modification.

By taking advantage of the UNIX standard, hardware and software vendors can hold down their development costs. This allows the consumer to enjoy a wider selection of compatible products at a reduced cost.

Another strength of the UNIX system is its file handling capabilities. Anything stored by a user is treated as a file. This includes text, data, and executable programs. The files are organized so that each file is owned by the individual who created it. Each individual is part of a group. The user population can include many groups. Access to the file can be specified in terms of read, write and execute for the individual, the group and the user population. Limiting access to certain files is important for security reasons.

A third often cited advantage of UNIX is the ease of developing new software. This facet of the UNIX operating system stems from its strong file system and its basic orientation toward programmer efficiency as opposed to machine efficiency.

Disadvantages of UNIX

One of the more common complaints made about the UNIX operating system is that it is not user friendly. The system does not automatically offer menus and prompts to the user. Also, output from the utilities is not labelled. And finally, the system will allow the destruction of files, and sometimes parts of the system itself, if care is not exercised.

These faults in the system can be corrected. The determining factor is whether the payoff from the change will justify the time spent on the modification. For example, to correct for the lack of menus, a menu driven user interface could be developed for novice users and/or special applications. Such an interface would replace the shell as the primary user interface. Users of such a development would not even need to be aware of the operating system.

Another facet of the UNIX system that draws complaints is the names given to the utilities. The complaint stems from the fact that these names are not mnemonic. That is, the name does not readily suggest the utility's function. This complaint, while valid, will become less important as the user becomes acclimated to the operating system.

XENIX

XENIX™ is a licensed version of UNIX created and marketed by the Microsoft Corporation. Microsoft has been a pioneer in the development of microcomputer languages (Microsoft BASIC) and operating systems (MS-DOS). The Microsoft XENIX operating system can be installed on microcomputers that have a 16-bit microprocessor and at least 256K of memory. XENIX functions more efficiently with computer systems which include more memory. At least 384K of memory is necessary to install the full version of XENIX.

Different computers require a slightly different implementation of XENIX. This is caused by the fact that a small portion of the operating system must be machine specific.

XENIX COMPARED TO OTHER PC OPERATING SYSTEMS

Generally, personal computers in the past have used either CP/M™ or MS-DOS™ as their operating system. Both of these systems were developed with the following constraints:

1. They had to be compact in order to fit in the machine and still have space for user applications.
2. They were to use floppy disks or cassettes as the mass storage medium.
3. They were to support only a single user.

XENIX is not compact. However, as the hardware prices fall (especially memory), the constraint of compactness becomess less of a factor.

UNIX was developed to use hard disks as the usual mass storage medium. The XENIX system will make better use of a hard disk than either MS-DOS or CP/M.

Whereas CP/M has been modified to support multiusers, UNIX was originally developed with the idea of supporting several users. A system designed for multiuser support is likely to function more efficiently than a system in which multiuser capabilities were added as an afterthought.

Finally, a UNIX system is said to drive the hardware harder than either CP/M or MS-DOS. This means that the computer is utilized a greater portion of the time. With CP/M or MS-DOS, the computer often sits idle while waiting for the single user to input a task. With XENIX, even a single user can keep the hardware busier by submitting multiple tasks. The system can run multiple tasks concurrently.

Chapter 2 — Installing the XENIX Operating System

INTRODUCTION

In this chapter we will discuss how to install XENIX. This chapter is designed to aid the individual responsible for getting a XENIX system up and running. For the purposes of this discussion, we assume the initial computer hardware to be an IBM PC XT with 256K of memory.

PREPARATION

Before trying to install the XENIX system, answer the following questions. We have formatted the text so that the answers can be recorded directly in the book. The questions and corresponding responses will provide a permanent record of how the system is installed.

Computer Hardware

☐ On which computer is the XENIX system being installed?

____

Note that the implementation of XENIX must match the computer being used. The box that contains the manuals and diskettes is clearly labeled with regard to which computer the enclosed XENIX implementation will drive.

☐ How much memory does the computer have?

____

Note that the XENIX system requires at least 256K of memory. The full version of the XENIX operating system requires 384K of memory. Before installing the smaller version of XENIX (memory less than 384K), read the last section in this chapter on modifying the XENIX installation. The modification of changing from the small version to the full version of XENIX has some undesirable side effects.

If you are unsure of the PC XT's memory, you can easily determine it. When the IBM PC XT is switched on, it executes a memory checking routine. While performing this function, it displays the amount of memory checked on the screen. The last number displayed before the operating system is booted is the amount of memory that the computer possesses. For other computers, check the manufacturer's specifications to determine how much memory is available.

Operating Systems

☐ Does the computer require another operating system, such as MS-DOS, in addition to XENIX on the hard disk?

____

This question should be considered carefully. The pertinent parameters include the amount of space each system requires and the amount of time each system will be active. Using a system other than XENIX will entail halting the XENIX operating system each time the other operating system is needed.

Also, the space on the hard disk will have to be shared by the two systems. Ten megabytes may seem like a very large amount of storage. But, the XENIX system itself occupies a large chunk of the available area. Also, the number of user files tends to grow much more quickly in a multiuser environment than in a single user environment.

Finally, even if another operating system must occasionally be used, it could be booted from a floppy diskette instead of the hard drive. This action would allow the entire hard disk to be dedicated to XENIX. The other operating system and all of its accompanying files would be stored on floppy diskettes. This operating system would never access the hard drive.

Which option is best can be determined by the number of files in both systems and the amount of usage each system will receive. If the other operating system will be used less than once a working day, it should surely not appear on the hard disk.

If the hard disk is not going to be partitioned, skip to the section on the swap area. If the hard disk is to be partitioned, it must be determined which files that are currently resident upon the disk must be saved. All files that are to be saved must be backed up on another media such as floppy diskettes.

Hard Disk Partitions

For this discussion, we assume that only two operating systems are to be resident on the hard disk. These two systems will be XENIX and MS-DOS.

☐ How much space should be devoted to the XENIX partition?

☐ How much space is left for MS-DOS?

____

Note that the developers of XENIX, Microsoft Corporation, suggest that at least 6 megabytes of the available storage be dedicated to the XENIX partition. Generally, it is advantageous to reserve as much disk space as possible for the XENIX partition. This extra storage space can be used to store additional user files. Also, system performance can be improved by defining a larger swap area. We discuss the swap area later in this chapter.

The best way to determine the partition size is to find the amount of storage required for the files that must be present in the MS-DOS partition. The MS-DOS partition should then be made slightly larger than this minimum value. The extra space is included for additional files. Three options are available if the MS-DOS partition requires more than 4 megabytes:

1. Delete some of the files in the MS-DOS partition. Seldom used files could be stored on another media such as floppy disks.

2. Back up all MS-DOS files on another media such as floppy disks. MS-DOS could then be accessed via this other media and would not appear on the hard disk.

3. Purchase another hard disk, MS-DOS would appear on one of the hard disks while XENIX would appear on the other.

If MS-DOS and XENIX can reside on the same hard disk, the amount of storage space (in bytes) must be converted to a number of cylinders. A cylinder is the unit of storage on the hard disk that is used to define partitions. The size of the fixed disk determines the number of cylinders it contains as well as the size of each cylinder. A ten megabyte disk contains 305 cylinders. Each cylinder contains 32,768 bytes of storage area.

In order to find the size of each partition in cylinders, insert the desired partition size in megabytes, into the following equations. Round the number of cylinders to the nearest whole number.

XENIX partition:

__________ megabytes ÷ 0.032768 = ______________ cylinders

MS-DOS partition:

305 - __________ XENIX cylinders = __________ cylinders

Swap Area

All the information necessary to execute concurrent tasks cannot be held in the computer's memory at one time. Data is traded between the disk and the computer memory. XENIX uses a swap area on the disk to help speed these transfers. A larger swap area allows the computer to operate more quickly. However, the amount of storage space available to the users on the hard disk is reduced as the swap area is enlarged.

Unless it is planned to add more disk space, the default value for the swap area should probably be used. If more disk space (i.e., another hard disk) is to be added to the system, more space can be allotted to the swap area.

Super-User Password

A super-user password should be created. This password controls access to the **root** account. The **root** account owns all of the XENIX files including the actual operating system stored in the root file. The super-user password is assigned to limit the use of the root account. The root account should be used only for system maintenance tasks.

An inexperienced user could unwittingly damage the system if he had access to the root account. An experienced user could access confidential files and alter them. So, the super-user password should be kept secure. Permanently store the super-user password in a safe, secure place. If the super-user password is lost, the only method of restoring it is to reinstall the entire XENIX system. We discuss the super-user password and system security in detail in chapter 5.

PROCEDURE

Once we have answered the questions on pages 23 to 27, we are ready to install the XENIX system. The XENIX system is divided into three portions. These are:

- The Operating System
- The Software Development System
- The Text Processing System

The operating system contains the XENIX programs necessary to maintain the system, manage the files, and allocate user accounts. The software development system includes the XENIX programs needed to code, compile and debug high-level and assembly language programs. The text processing system includes the XENIX programs necessary to produce documents. In this book we concentrate on the set containing the operating system.

Before beginning the installation procedure, be sure that all of the files on the hard disk that are to be preserved have been copied to another media, such as floppy disks.

The installation procedure is presented here in the form of a check list. In order to prevent confusion, check each item off as it is completed. A check box is inserted in the text only where some action must be taken.

MS-DOS

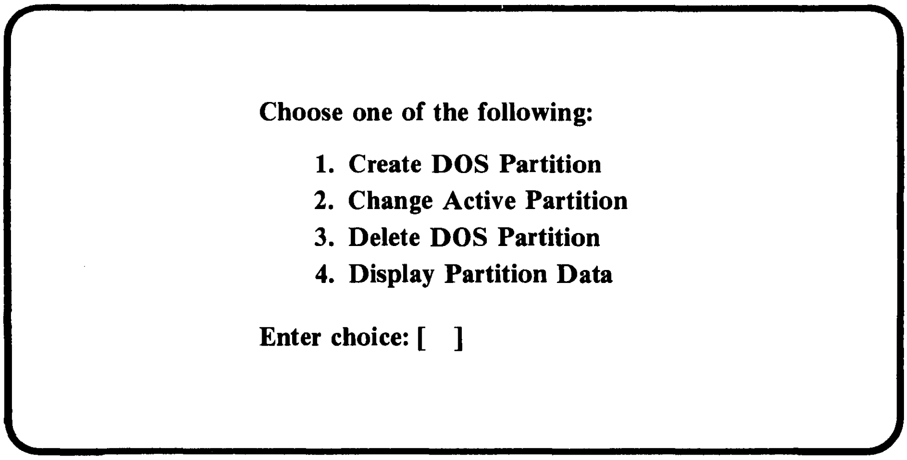
If the hard disk is to be shared with another operating system, first create and format the DOS partition. If the hard disk is to be dedicated entirely to XENIX, skip to the next section. The literature included with the XENIX system indicates that the DOS partition should be created last. However, we had trouble formatting the DOS partition when the XENIX partition was present.

☐ To begin partitioning the hard disk, insert the floppy disk containing the MS-DOS operating system into the floppy drive. Power the computer on.

☐ After MS-DOS has booted from the floppy disk and the system prompt appears on the screen, type:

FDISK

and press the Return key. The menu listed in figure 2.1 will be displayed.



```
Choose one of the following:

    1. Create DOS Partition
    2. Change Active Partition
    3. Delete DOS Partition
    4. Display Partition Data

Enter choice: [   ]
```

Figure 2.1. The FDISK menu for MS-DOS.

☐ Type **3** and press the return key. This action destroys the present DOS partition. The screen will display:

```
        Partition        Status     Type     Start     End     Size
            1               A        DOS       0       304      305



Total disk space 305 cylinders.


Warning! All data in the DOS partition will be DESTROYED.
Do you wish to continue .................... ?  [N]
```

☐ This is the last chance to save any additional DOS files. If all the necessary files have been saved, type **y** and press the Return key. Press the [ESC] key to return to the FDISK menu (figure 2.1).

☐ Type **1** and press the Return key. This action allows a new DOS partition to be set. The following prompt will appear on the screen:

```
                    Do you wish to use the entire fixed
                    disk for DOS (Y/N) .......... [Y]
```

☐ Type **n** and press the Return key. This action allows a DOS partition smaller than the entire disk to be set. The screen will display:

```
No partitions defined

Total disk space is 305 cylinders.
Maximum available space is 305
cylinders starting at cylinder 0.

Enter partition size .......... : [305]
```

☐ Type the value of the size of the DOS partition, in cylinders, as determined in the section on hard disk partitions and press the Return key. We typed 55. The screen will dispay:

```
Enter starting cylinder number .. : [          0]
```

☐ Type the value of (305 - partition size) and press the Return key. We typed (305 - 55) = 250.

☐ Press the [ESC] key. The FDISK menu (see figure 2.1) will be displayed on the screen.

☐ Type **2** and press the Return key. Option two allows the active partition to be changed. The screen will display:

```
        Partition          Status    Type     Start     End     Size
            1                N       DOS       250      304      55




Total disk space is 305 cylinders.



Enter the number of the partition you want to make
active .................... :  [  ]
```

The partition table may vary slightly, depending upon the size of the DOS partition.

☐ Type **1** and press the Return key. This action causes the first partition to be activated. Note that the status has now changed from "N" to "A".

☐ Press the [ESC] key. The screen will display the FDISK menu. Press the [ESC] key again. This action will return control to the system. The MS-DOS partition must now be formatted.

☐ Type:

```
FORMAT d:/S/V
```

Where *d* represents the hard disk's letter (generally C). This command will cause the MS-DOS partition to be formatted and a copy of DOS to be loaded into the partition. While the formatting process is taking place, the screen will display:

```
Press any key to begin formatting C:
```

☐ Press any key. The screen will display:

```
Formatting...
```

When the formatting of the DOS partition for MS-DOS is complete and the DOS has been transferred to the partition, the screen will display:

```
Formatting... Format complete
System transferred

Volume label (11 characters, ENTER for none)?
```

☐ Type the desired label or press the Return key if no label is needed.

XENIX

We are now ready to create the XENIX partition and install the XENIX system.

☐ Insert the diskette labeled “BOOT floppy” into the primary floppy disk drive. Power on the computer. The copyright message will be displayed. Several peripheral self-checks are then performed. After these tests the screen will display:

```
Boot sequence:
H
mem = xxxK
Insert root floppy and hit return
```

☐ Remove the BOOT floppy from the drive. Insert the disk labeled “ROOT floppy” in its place. Press the Return key. The screen will display:

```
Z
No single-user login present
Entering System Maintenance Mode
XENIX V3.0 Hard Disk Initialization Floppy
(backspace is ^ h, erase line is ^ u)
Use "hdinit" to initialize hard disk.
Boot Floppy
```

☐ When the above message appears, type the following:

hdinit

and press the Return key. The screen will display the following warning:

```
WARNING:  This installation program could destroy
          the present contents of your hard disk.
          Refer to DOS documentation to preserve
          an existing DOS file system

Do you wish to continue (y/n)?
```

☐ Type y and strike the Return key. A message pertaining to modifying the partition table will appear on the screen. A menu will also appear on the screen. A listing of this menu is contained in figure 2.2.

```
Select one of the following option numbers or 'q' to exit program:

            1. Create Partition
            2. Change Active Partition
            3. Delete Partition
            4. Display Partition Table
            5. Create XENIX Partition for Whole Disk

                  Enter Choice:
```

Figure 2.2. The **fdisk** menu for XENIX.

☐ If the entire hard disk is to be dedicated to XENIX, choose option 5 and skip to the box marked with an asterisk. Otherwise, choose option 4 by typing 4 and pressing the Return key. The screen will display a partition table similar to the following:

```
                                                              Device
Partition   Status    Type     Start    End     Size       "/dev/hd0"
    1         N      XENIX      000     000     000
    2         N      XENIX      000     000     000
    3         N      XENIX      000     000     000
    4         A       DOS       250     304     055
```

This display will be followed by the **fdisk** menu that is listed in figure 2.2.

☐ If the information in the partition table is correct, type **1** and press the Return key. This action allows a XENIX partition to be created.

☐ Select the first partition by typing **1** and pressing the Return key. The screen will display:

```
Starting cylinder:
```

Type **0** and press the Return key. The screen will display:

```
Size in cylinders:
```

Type in the answer determined for the XENIX partition in the section on hard disk partitions, and press the Return key. Note that we elected to devote 250 cylinders to the XENIX partition. The screen will display the **fdisk** menu (see figure 2.2).

☐ Now activate the new partition using option 2. Type **2** and press the Return key. The screen will display:

```
Type <q> <CR> to return to menu
Which partition do you want active:
```

Type **1** and press the Return key to activate the XENIX partition. The screen will display the **fdisk** menu (see figure 2.2).

☐ Select the 4 option to determine whether the partition is set correctly. Type **4** and press the Return key. The screen will display a table similar to the following partition table:

```
                                                                   Device
Partition   Status    Type     Start      End       Size      "/dev/hd0"
    1         A       XENIX     000       249       250
    2         N       XENIX     000       000       000
    3         N       XENIX     000       000       000
    4         N        DOS      250       304       055
```

This display will be followed by the **fdisk** menu (see figure 2.2).

*☐ If the partition table is correct, exit the program by typing **q** and pressing the Return key. Note that if option 5 was selected, the partition table does not need to be checked. The installation program automatically modified the partition table. The screen will display the partition table once again. The table will be followed by the prompt:

```
Do you want to update "/dev/hd0" with your changes (y/n):
```

☐ If the table is correct, type **y** and press the Return key. The screen will display:

```
Analyzing hard disk:
this procedure will take at least 5 minutes.
Scanning blocks xxxx-xxxx
```

The numbers represented by the x's will change as different blocks are checked. When the analysis is complete and if the disk is perfect, the screen will display:

```
0 had blocks in bad block table,
making hard disk bootable.
1 + 0 records in
1 + 0 records out
178 + 1 records in
178 + 1 records out
```

A message pertaining to the swap space will also appear, followed by the prompt:

```
Swap space allocation (1000 - xxxx)?
```

☐ Enter the value for the swap space determined in the section on swap space and press the Return key. We decided to use the default value of 1200. Several messages relating to the details of the installation will be displayed on the screen. Eventually the following message will appear:

```
Hard disk initialization procedure completed.
AFTER you see the message: "**Normal System Shutdown**"
please boot off the hard disk by
        opening the floppy door,
        removing the floppy,
        powering the system off,
        waiting 10 seconds,
        and powering the system on again.

**Normal System Shutdown**
```

- ☐ Remove the ROOT floppy from the floppy disk drive and power the system off.

- ☐ After waiting at least the specified ten seconds, power the system on again. The computer will perform its preliminary self-checks and commence booting in the XENIX system from the hard disk. The XENIX copyright message will appear on the screen followed by several messages pertaining to system checks. Finally, the following message will appear on the screen:

```
Install Xenix Operating System package (y/n)?
```

- ☐ Type y and press the Return key. The following message will appear on the screen:

```
For each floppy in the distribution set,
insert the floppy into the floppy disk drive,
and answer "y".
Type the letter "n" after the last floppy.

Should you ever see the message "tar: please mount new volume."
insert the next floppy, and hit the Return key."

First floppy y/n
```

- ☐ Insert the floppy disk labeled "Floppy Number: 1". Type y and press Return key. The following message will appear on the screen:

```
Extracting files from floppy
```

This message will be followed by a list of files being extracted and information about the length of each file. Eventually, the following prompt will appear on the screen:

```
Does your system have 384K RAM or more (y/n)?
```

- ☐ Answer the prompt according to the amount of memory contained in the machine. We covered memory in the section on computer hardware. We typed **n** and pressed the Return key. Several messages pertaining to the progress of the installation procedure will appear. Finally, the following prompt will appear on the screen:

```
Continue with installation...
Next Floppy (y/n)?
```

- ☐ Remove Floppy Number: 1 from the floppy drive and insert the floppy disk labeled, "Floppy Number: 2." Type **y** and press the Return key. Messages about the files being extracted from the floppy disk and installed onto the hard disk will appear.
- ☐ Continue installing the floppy disks from the distribution set. Each time the message:

```
Next floppy (y/n)?
```

appears, remove the floppy disk presently in the floppy drive and insert the disk with the next higher number. Type **y** and press the Return key. For the IBM PC XT, there are eight numbered floppies.* The first seven must be installed. The eighth disk is optional. It contains information necessary to interface your computer with other computers. If this capability is not desired, this eighth disk need not be installed. When the following message appears on the screen:

```
The next floppy contains networking programs.
These are used for communications between computers via
phone lines or direct links. If you don't need this
feature and would like to save disk space,
do not install this floppy.

Next Floppy (y/n)?
```

* This number may vary for machines that store a different amount of data on a floppy disk.

☐ Select the appropriate option. We typed **n** and pressed the Return key. The following messages will appear on the screen:

```
Xenix Operating System installation complete.
Install Software Development package (y/n)?
```

☐ The Software Development system requires 384K of RAM. Since we presently have only 256K of RAM, we opted not to install the Software Development System. If you have enough memory and purchased the package, install it now. We typed **n** and pressed the Return key.

If you choose to install the Software Development package, you will be prompted to insert the distribution floppies, one at a time, into the drive. This installation is very similar to the installation of the Operating System package. After installing (or skipping) the Software Development package, the following message will appear on the screen:

```
Install Text Processing package (y/n)?
```

☐ The Text Processing package also requires 384K of RAM. We chose not to install the Text Processing package at this time by typing **n** and pressing the Return key.

If you do decide to install the Text Processing package, installation will proceed in a fashion very similar to the installation of the operating system package.

After the Text Processing package has been installed (or skipped), a message pertaining to the system setup will appear. The system will then be shutdown as it was after the installation of the ROOT floppy. The message:

```
** Normal System Shutdown **
```

signals the shutdown.

☐ Remove the last floppy from the floppy drive. Power off the computer. Wait at least ten seconds before powering the comptuer on again.
The computer will run through its initial checks. The XENIX copyright message will then appear on the screen followed by the system check messages. Finally, the following prompt appears on the screen:

```
Type control-d to proceed with normal startup,
(or give root password for system maintenance):
```

☐ Press the Return key. Since there are no users yet, the normal procedure will not work. Since there is no root password yet, pressing the Return key allows access to the super-user mode. The following messages will appear on the screen:

```
Entering System Maintenance Mode
Term = (ibm)
```

☐ Press the Return key. Messages pertaining to the terminal for the IBM PC XT will appear followed by the Maintenance Mode prompt: #. We should now create the super-user password. This password will help protect the system from unauthorized use.

☐ Type:

```
passwd root
```

and press the Return key. The screen will display:

```
Enter new password (minimum of 5 characters)
Please use a combination of upper and lower case letters and numbers.
New password:
```

☐ Type in the super-user password and press the Return key. Note that the characters entered will not appear on the screen. The screen will display:

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

☐ Type the super-user password again and press the Return key. Note that the characters will not be displayed on the screen. If you made a mistake, the screen will display:

```
They don't match; try again.
New password:
```

The new password must be re-entered. The system will then prompt for a new entry to verify the password. If the two inputs match (remember that upper and lower case letters are treated as unrelated characters), the Maintenance Mode prompt (#) will appear on the screen.

☐ Type:

```
/etc/haltsys
```

and press the Return key. This command causes a system shutdown.

☐ Power off the computer. The installation of XENIX onto the hard disk is now complete. The system is now ready to be configured so that it can accept and execute user jobs. We discuss how to accomplish these tasks in the next chapter.

MS-DOS Files

☐ Insert the floppy disk containing a copy of MS-DOS into the floppy disk drive and power on the computer. After the system boots, change the active partition to the DOS partition. This can be accomplished using the following steps.

☐ At the system prompt, type:

```
FDISK
```

and press the Return key. The FDISK menu will be displayed on the screen. This menu is listed in figure 2.1.

☐ To activate the MS-DOS partition, type **2** and press the Return key. The screen will display a partition table and a prompt asking which partition should be activated. Type the number corresponding to the DOS partition and press the Return key. Press the [ESC] key to return to the FDISK menu. Press the [ESC] key again to return to the system level.

☐ The MS-DOS files can now be placed in the MS-DOS partition. To copy the DOS programs onto the hard disk, type:

```
COPY * * d:
```

where *d* represents the hard disk's letter (generally C). This command will cause the DOS programs on the DOS diskette to be copied onto the hard disk.

- ☐ Remove the DOS diskette from the floppy drive. Insert the first diskette containing the DOS files to be placed on the hard disk. Use the COPY command to place these files onto the hard disk.
- ☐ Repeat the above process for all diskettes containing files to be placed on the hard disk.
- ☐ When all the necessary files have been tranferred to the hard disk, remove the last diskette from the floppy drive and power off the system.

PARTITION USE

Both XENIX and MS-DOS are installed in a partition on the hard disk. When the power to the computer is turned on, the system in the currently active partition will be booted.

The FDISK program must be used to change the currently active partition. Each partition contains its own version of FDISK. When operating in MS-DOS, the XENIX system can be activated using the following steps:

1. Activate the XENIX partition using option 2 from the FDISK menu.
2. Boot the XENIX partition by resetting the system. The IBM PC XT is reset by pressing the control, alternate and delete keys simultaneously.

When operating in XENIX, MS-DOS can be activated using the following steps:

1. Activate the DOS partition using option 2 from the **fdisk** menu.
2. Shutdown the XENIX system. Use the **/etc/haltsys** command to accomplish system shutdown. Never power off the computer without first shutting down the XENIX system.
3. Power off the computer.

4. Wait at least 10 seconds.

5. Power on the computer.

MODIFYING THE XENIX INSTALLATION

Caution must be exercised when modifying the XENIX installation. Some of these installation procedures will cause the current set of user accounts and files to be deleted. Such an occurrence could be a major disaster. Before proceeding with any XENIX installation modification, check the following sections for which modifications will cause deletions from the XENIX file system. If the modification does cause deletions, be sure to back up all the necessary files. For more information on file backup, see chapter six.

More Memory

If the smaller version of the XENIX operating system was originally installed and the computer's RAM has subsequently been upgraded to 384K bytes or more, the full version of the XENIX operating system can be installed. The installation of the full version will cause all the current user accounts and files to be cleared from the system. Also, the super-user password will be destroyed. Do not proceed with the modification until the necessary files have been backed up. Use the **sysadmin** utility to create a complete system backup. We discuss how to use this utility in chapter six.

To accomplish the installation of the full version of the XENIX operating system, use the following procedure.

☐ Login as the super-user.

☐ At the super-user prompt (#), type:

```
/etc/install
```

and press the Return key. The screen will display the following:

```
Should you ever see the message "tar: please mount new volume",
insert the next floppy and hit the Return key.

First Floppy (y/n)?
```

Insert the diskette from the operating system distribution set labeled "Floppy Number: 1" into the floppy disk drive. Type **y** and press the Return key. This action will cause the names of the files stored on the diskette to be displayed as they are extracted from the diskette to the hard disk. Eventually, the screen will display the prompt:

```
Does your system have 384K or more (y/n)?
```

- ☐ Type **y** and press the Return key. The screen will display several messages about the progress of the replacement of the smaller version of XENIX by the full version of XENIX.
- ☐ When the screen displays the following:

```
Next Floppy (y/n)?
```

type **n** and press the Return key. A message stating that the installation is complete will appear on the screen. The screen will then display the super-user prompt.

- ☐ Remove the diskette from the floppy drive.

The full version of the XENIX operating system is now installed. Create a new super-user password as detailed earlier in this chapter. Restore the user accounts and files from the backup copies. The procedure to accomplish this task is detailed in chapter six.

Adding Another System

If only the XENIX operating system was installed initially and another distribution system is needed at a later date, the original installation can be modified to include the necessary addition. Three typical additions include the XENIX Text Processing System, the XENIX Development System, and the diskette from the XENIX Operating System containing the networking files.

A modification of this type will not delete any of the user accounts or files. However, both the Text Processing System and the Development System add a significant number of files to the file system. If the file system is already fairly large, this addition may adversely affect system performance.

☐ Login as the super-user.

☐ At the super-user prompt (#), type

```
/etc/install
```

and press the Return key. The screen will display the following:

```
Should you ever see the message "tar: please mount new volume",
insert the next floppy and hit the Return key.

First Floppy (y/n)?
```

☐ Insert the first diskette of the set into the floppy drive. Type **y** and press the Return key. The screen will display information about each file on the diskette as it is copied to the hard disk. When all the files have been transferred, the screen will display:

```
Next Floppy (y/n)?
```

☐ Repeat the above step for each of the diskettes in the set. Load them in order according to the numbers printed on them.

☐ After the last diskette, answer the prompt by typing **n** and pressing the Return key. A message stating that the installation is complete will appear on the screen. The super-user prompt will then be displayed.

☐ Remove the diskette from the floppy drive.

The new system is installed.

Chapter 3 — Preparing the XENIX Operating System for Use

INTRODUCTION

In this chapter we will discuss the steps that must be taken to prepare the new XENIX installation for users. Topics of discussion include the addition of peripheral devices such as printers and terminals, allocation of the necessary user accounts and creation of groups.

This chapter is intended to guide the individual responsible for readying a newly installed XENIX system for use. In the next chapter, we will discuss how to modify the system configuration to conform to changes in personnel and equipment.

This chapter is presented in a fashion similar to the last chapter. The text will provide information pertinent to the configuration of the system. We also discuss the logic motivating system configuration. The actual configuration tasks are presented in the form of check lists.

SYSTEM STARTUP AND SHUTDOWN

The XENIX system is much more complex than other popular personal computer operating systems such as MS-DOS. Starting and stopping the XENIX system is more difficult than with MS-DOS. This section details how to accomplish XENIX system startup and shutdown. The XENIX system should never be halted by simply powering off the computer. Such an action causes a system crash and endangers the file system.

Starting

In order to start the XENIX system, the XENIX partition must be active. We discussed activating partitions in the last chapter. If the XENIX partition is active, the system can be started using the following steps:

1. Power on the computer. The machine will run through its hardware checks. The XENIX system with then run through its system and file checks. After these steps are completed, the screen will display:

```
Type control-d to proceed with normal startup,
(or give the root password for system maintenance):
```

2. Proceed to either the normal startup or system maintenance by typing the appropriate response, as indicated by the message. Use the system maintenance mode only for maintenance tasks. Should the screen ever display the message:

   ```
   Proceed with cleaning (y or n)?
   ```

   Type y and press the Return key. We will discuss more about cleaning the file system in chapter 6, in which we discuss system maintenance.

Stopping

There are two ways to stop the XENIX system without causing the system to crash. One of these methods halts the system immediately. The other method provides a warning message on all currently active terminals and halts the system after a specified time. This second method should be used whenever the system is being accessed by individual users. The time between the warning and the system shutdown allows the users to save any current work and logoff.

IMMEDIATE

To accomplish an immediate system shutdown, use the following steps:

1. Login as the super-user. To complete this task, type **root** at the login prompt (xenixt!login:) and press the Return key. The system will prompt for the super-user password. Enter the password and press the Return key.
2. At the super-user prompt (#), type:

   ```
   /etc/haltsys
   ```

   and press the Return key. The screen will display:

   ```
   ** Normal System Shutdown **
   ```

3. Power off the computer.

DELAYED

To shutdown the system after a specified amount of time, use the following steps:

1. Login as the super-user. To complete this task, type **root** at the login prompt (xenixt!login:) and press the Return key. The system will prompt for the super-user password. Enter the password and press the Return key.

2. At the super-user prompt (#), type:

```
/etc/shutdown
```

   and press the Return key. The following prompt will appear on the screen:

```
Minutes to shutdown? (0-15):
```

3. Type the desired number of minutes (e.g., 5) and press the Return key. The system will display the following at each active terminal:

```
Broadcast Message from root
XENIX Shutdown in 5 minutes.
Clean up and log off.
```

   As soon as everyone has logged off, or after 5 minutes, the system will shutdown.

4. When the message:

```
** Normal System Shutdown **
```

   appears, power off the computer.

NECESSITIES

To complete some of the configuration tasks, a proficiency in the use of certain XENIX features is required. The file system and file access methods must be understood. Also, the ability to use one of the XENIX text editors is necessary.

If you have these skills, go ahead with the configuration tasks. If you do not presently have these capabilities, create a temporary user ID and learn them. The user-oriented chapters on the file system and the editors offer tutorials on these subjects. The line editor, ed, is most easily mastered. To create a temporary ID, observe the following steps:

☐ Power on the computer.

☐ When the message:

```
Type control-d to proceed with normal startup,
(or give root password for system maintenance):
```

appears on the screen, type in the root password. Recall that the screen will not display the password as it is entered.

☐ When the message:

```
Term = (ibm)*
```

appears on the screen, press the Return key.

☐ At the maintenance mode prompt (#), type:

```
mkuser
```

and press the Return key.

* This message may vary for systems other than IBM PC XT.

The screen will display:

```
                    Newuser
            Add a user to the system

  Do you require detailed instructions? (y/n/q):
```

☐ Type **n** and press the Return key. The following prompt will appear on the screen:

```
Enter new user's login name:
```

☐ Type **temp** and press the Return key. The following prompt will be displayed:

```
Enter password:
```

☐ Type in a password. The system will prompt for a verification:

```
Re-enter for check:
```

☐ Type the password again. The screen will display the following if the two passwords matched:

```
Enter Group:
```

☐ Press the Return key. The following message will appear on the screen:

```
Please enter comment: > --------------------
                      >
```

☐ Enter any comment and press the Return key. But, do not type more than 20 characters. The comment field can be left blank by just pressing the Return key. The screen will display:

```
User name is: temp
Comment field is:

Password file entry is:

                          temp : xxxxxxxxxxxxxx

Do you want to change anything (y/n/q):
```

☐ Type **n** and press the Return key. Several messages about the creation of the new user ID will appear. Finally, the following prompt will appear on the screen:

```
Do you want to add another user? (y/n/q):
```

☐ Type **n** and press the Return key. The maintenance mode prompt will appear. Type control-d (press both the control key and the d key). This begins the normal system startup. A message about the time and date will appear.

☐ Answer the prompt for the time and date. The following message will appear on the screen:

```
xenixt!login:
```

☐ Type **temp** and press the Return key. A prompt for the password will then appear on the screen:

```
Password:
```

☐ Type in the password that you just created for the **temp** ID and press the return key. If the password was typed correctly, the screen will display the welcome message. Finally, the screen will display:

```
TERM = (ibm)
```

☐ Press the Return key. When the screen displays the XENIX shell prompt ($), you can start experimenting with the XENIX utilities.

SYSTEM CONSOLE

The system console refers to the screen and keyboard attached directly to the computer on which the XENIX system is installed. The system console can be used like any terminal that is attached to the computer via a serial interface.

TERMINALS

Adding terminals to the system allows more than one person to use the computer at the same time. XENIX will support many different types of terminals. A full listing of these terminals is given in the XENIX file **/etc/termcap.**

To accomplish the addition of a new teminal, the following four requirements must be fulfilled:

- Enable a special device file
- Set the line baud rate
- Make the physical connection
- Set the terminal type.

We will discuss how to fill each of these requirements in the following sections.

Enable Special Device File

For the IBM PC XT implementation of XENIX, there are four special device files that will allow up to two terminals to be added to the system.* The system allows two serial ports to be addressed. Each of the ports is given two names, one for direct connection and one for connection via a modem. The special file names are as follows:

* Other XENIX implementations may vary slightly with regard to the actual number and names of these files.

- **/dev/tty11** - main serial interface, direct connection
- **/dev/tty12** - alternate serial interface, direct connection
- **/dev/tty13** - main serial interface, modem connection
- **/dev/tty14** - alternate serial interface, modem connection

The main serial interface is the interface included as standard equipment with the IBM PC XT. The alternate serial interface is an optional enhancement. An example of an alternate serial interface would be the serial interface included on many popular memory expansion boards.

To enable one of these special device files, use the **enable** command. For example, to enable the alternate serial interface for a direct connection to a terminal, type:

```
enable /dev/tty12
```

and press the Return key. Only the super-user is permitted to use the **enable** command.

The XENIX system documentation cautions that a full minute should be allowed to elapse between use of the enable command. Failure to do so may cause a system crash.

Set Line Baud Rate

Baud rate is a measure of how fast data is being transmitted. The baud rate indicates how many bits per second are being output. Both the sending and the receiving device must use the same baud rate. If they do not, communication becomes garbled and breaks down.

The documentation included with the XENIX operating system suggests using the **stty** utility to set the baud rate of the serial line so that it agrees with the terminal's baud rate. However, we could not get a functional terminal using this method.

Instead, we suggest the use of the following scheme to set the baud rate of the serial line.

☐ Look in table 3.1 for the desired baud rate. Note the letter associated with the desired baud rate.

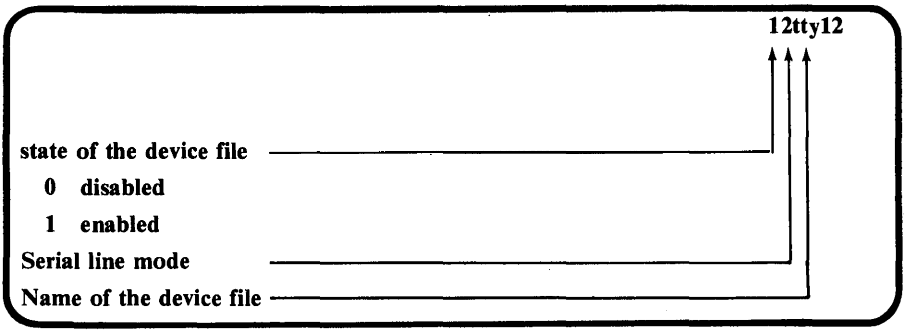
Table 3.1. Baud rate and associated letters.

Baud Rate	**Letter**
50	a
75	b
110	c
134.5	d
150	e
200	f
300	g
600	h
1200	i
1800	j
2400	k
4800	l
9600	m

☐ Edit the **/etc/ttys** file to reflect the desired baud rate. Type:

```
ed /etc/ttys
```

☐ Display the contents of the **/etc/ttys** file by typing **1,$p** and pressing the Return key. Each entry will have the following format:



☐ The serial line mode must be changed to the letter associated with the desired baud rate. Use the + and - commands to make the line containing the device file of interest the current line. Then use the **s** command to actually make the change. For example, to set the serial line mode of the **/dev/tty12** file to 300 baud, we typed:

```
-2p
```

and pressed the Return key. The screen displayed the following:

```
12tty12
```

We then typed:

```
s/12t/1gt/p
```

and pressed the Return key. The screen displayed:

```
1gtty12
```

☐ Type **w** and press the Return key. This action updates the **/etc/ttys** file with the changes.

☐ Type **q** and press the Return key to exit the line editor.

Make Physical Connection

To complete the physical connection of the terminal to the computer, a cable is required. For the IBM PC XT, the cable must have a female DB-25 connector attached to one end. This connector will be attached to the serial port in the IBM PC XT. The other end of the cable must have a connector that is compatible with the port in the terminal. Figure 3.1 diagrams the connection.

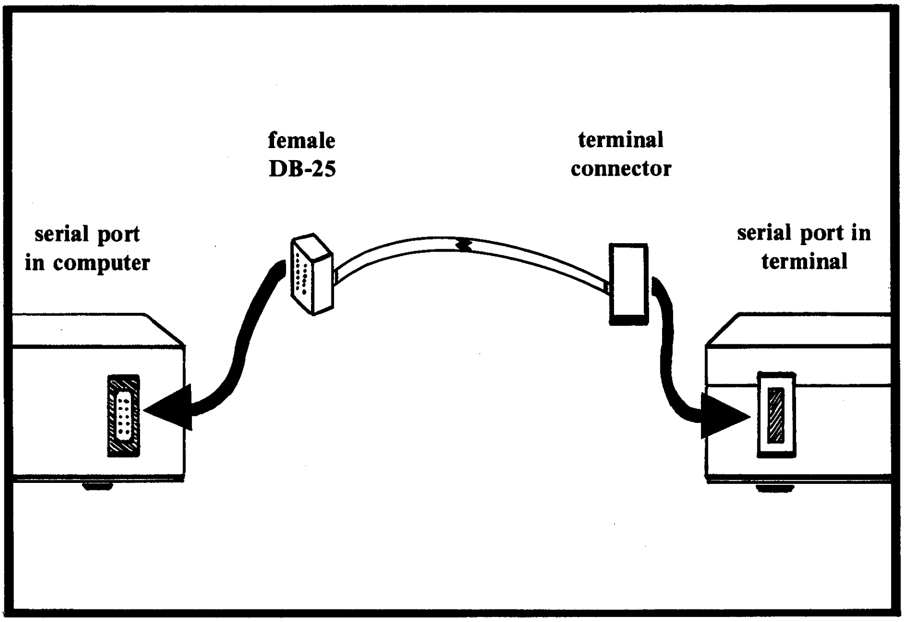


Figure 3.1. Diagram of connection between an IBM PC XT computer and a terminal.

Once the physical connection has been made, power on the terminal. Press the terminal's Return key several times. The XENIX login message should appear on the terminal's screen. If the message does not appear, check table 3.2 for possible causes and their solutions.

Table 3.2. Possible causes for no login message appearing at the terminal.

Cause	Solution
XENIX in system maintenance mode (single user)	Press control-d. Answer the prompt for date and time.
Serial connection incorrect	Check to be sure all plugs are firmly seated in their sockets. Also, check serial line pin outs for compatibility.
Serial line not enabled	Check to be sure that the correct special device file has been enabled. Switch the connector to the enabled serial line or enable the correct serial line.
Serial line baud rate incorrect.	Change baud rate so that the terminal and the computer agree.

Set Terminal Type

Generally, the terminal type should be set in the user's **.profile** file to the terminal type he most often uses. We discuss **.profile** files later in this chapter.

The **tset** utility is used to set the terminal type. The **/etc/profile** file (also discussed later) contains an example of the general use of the **tset** utility in a profile file. The specific command is:

```
eval 'tset -m :\? ibm -e -s -r'
```

Note that the terminal type is set to the IBM PC console by the appearance of **ibm.**

To change the terminal type to a different type of terminal for an individual user, the following statements should be appended to the **.profile** file:

```
eval 'tset -m :\? terminal type -e -s -r'
export TERM
```

where *terminal type* represents one of the terminal names found in the **/etc/termcap** file. Note that the question mark causes the system to prompt the user if he really wants this type of terminal. A null response (i.e., pressing the Return key) will cause the specified terminal type to be selected. Alternately, the user can specify a different terminal type.

For example, the terminal type may be set to the Digital vt100 by the appearance of the following commands in the user's **.profile** file:

```
eval 'tset -m:\? vt100 -e -s -r'
export TERM
```

Be sure to use the terminal names from the **/etc/termcap** file. We found some of these names to be incorrect in the printed system documentation.

PARALLEL PRINTERS

A parallel printer can be added to the XENIX system. This addition allows users to obtain hard copies of files and output. The XENIX system provides a spooling routine. This routine places the desired print file in a queue. The file is printed when it reaches the front of the queue. The print spool routine allows all system users to share one printer. The files to be printed must be queued otherwise different users' output would be interspersed.

The command used to create the parallel line printer within an IBM PC XT based system has two forms. The form used depends on which monitor adaptor board is installed in the system. For a system with a monochrome adapter board, the command appears as follows:

```
mknod /dev/lp c 6 1
```

A color monitor board requires the following command:

```
mknod /dev/lp c 6 0
```

Note the spacing required between the elements in the command. Also, we found that the parallel line printer was already added for a monochrome adaptor board on the new XENIX installation. Finally, the super-user must issue the above commands.

USER GROUPS

Structure

Before entering any user accounts, first determine the necessary group structure. **Group structure** is the ordering of individual accounts into sets related by the user's interests. The group structure should be determined by the tasks each group must accomplish. Group structure can be used to enhance the security of the system. We will illustrate this and other concepts in group structuring with a simple example.

EXAMPLE COMPANY

Our example will be of a hypothetical small company. The company has two officers, a president and a vice president. The company also has a design team and a secretarial pool. Figure 3.2 contains a diagram of the company's organization.

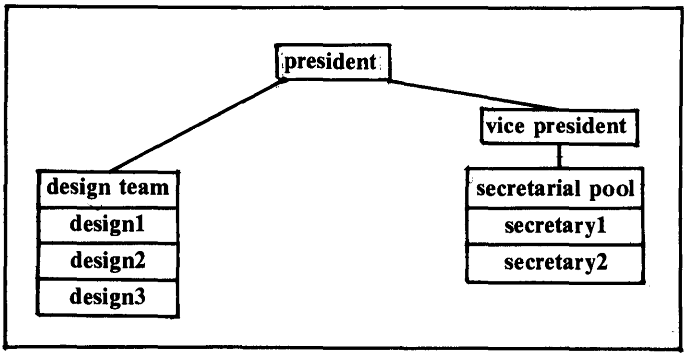
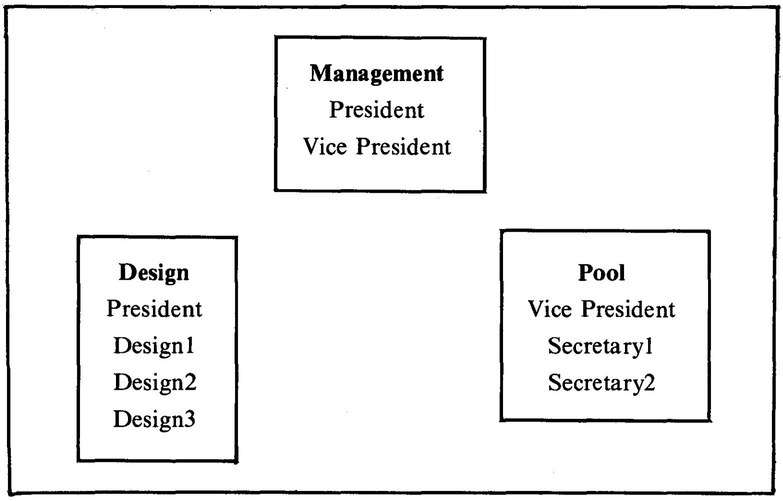


Figure 3.2. Diagram of the organization of the example company.

This structure suggests the following three groups:

1. Management
2. Design
3. Pool (for secretarial pool)

Figure 3.3. Diagram representing the group structure for the example company.



BENEFITS

This group structure allows for several advantages:

1. Confidential corporate files, such as fiscal information, can belong to the management group. Access to all other groups can be denied.

2. Files pertinent to the present design project can belong to the design group. Each member of the design team can access the files. Access to these files can be denied all other groups. Thus the design files can be kept protected from outsiders. Note that the president can monitor project progress because he is included in the design group.

3. The pool group can have access to all the correspondence and report files. Additionally, the vice president can own all confidential files, such as accounting records. The secretaries can be allowed write access in order to update these files. But, they can be denied read access.

Creating Groups

We are now ready to create the groups required by the group structure. We must create the following three groups:

- management — ID number = 100
  - president
  - vice president
- design — ID number = 200
  - design1
  - design2
  - design3
  - president
- pool — ID number = 300
  - secretary1
  - secretary2
  - vice president

☐ To create the group, first login as the super-user.

☐ Check the present group list to be sure the selected group names and ID numbers are unique. To accomplish this task, type:

```
cat /etc/group
```

This command will cause all of the entries in the group file to be displayed on the screen. Each entry will have the form:

```
group name :: group ID : users
```

☐ If all the selected group names and ID's are unique, edit the group file to include the new entries. Use the following steps to accomplish this task.

☐ Type:

```
ed /etc/group
```

and press the Return key. This action invokes the XENIX line editor. The screen will display a number and an asterisk. The asterisk is the line editor's prompt.

☐ Type **a** and press the Return key. The cursor will now be located at the line below the asterisk.

☐ Now type in all the groups, one per line. Be sure to use the following format:

```
group name :: group ID :  user1, user2,...
```

We typed:

```
management :: 100:
design :: 200: prsdnt
pool :: 300 : vcprsdnt
```

Note the lone period on the last line. The period is used to signal the end of additions to the file. The other group members will be added to the group when we create the user accounts. The screen will display the asterisk prompt again.

☐ Type:

```
1,$p
```

and press the Return key. This action will cause the entire file to be displayed on the screen. Proof the file to be sure the new group data was entered correctly and none of the old group information was changed.

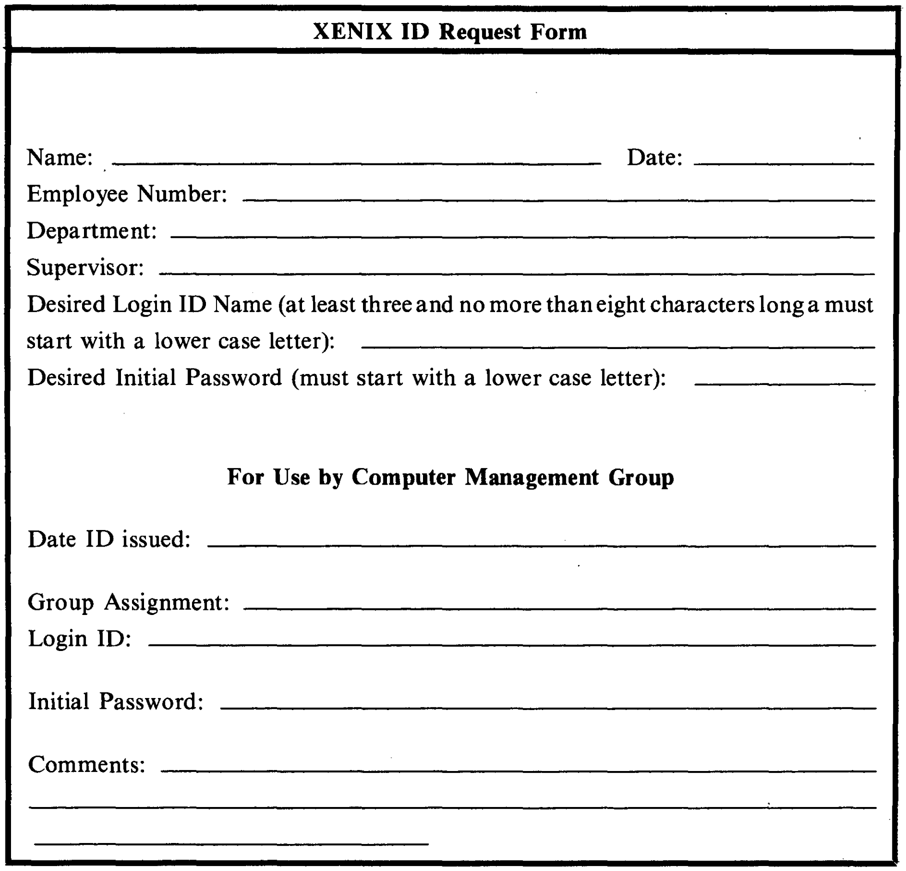
☐ If everything was correct, type **w** and press the Return key. This action causes the edited file to overwrite the copy on the hard disk.

- ☐ Type **q** and press the Return key when the asterisk prompt appears. This action causes an exit from the line editor. The screen will display the super-user prompt (#). The creation of the groups is now complete.

INDIVIDUAL USER ACCOUNTS

Collecting Account Data

Information about each necessary user account should be collected. One easy way to gather the information for the individual accounts is to have each employee who is to have an account fill out an "ID Request Form." We suggest a form similar to the one shown in figure 3.4.



XENIX ID Request Form

Name: ______________________ Date: __________

Employee Number: ______________________

Department: ______________________

Supervisor: ______________________

Desired Login ID Name (at least three and no more than eight characters long a must start with a lower case letter): ______________________

Desired Initial Password (must start with a lower case letter): __________

For Use by Computer Management Group

Date ID issued: ______________________

Group Assignment: ______________________

Login ID: ______________________

Initial Password: ______________________

Comments: ______________________

____

____

Figure 3.4. Suggested format of an ID request form.

Creating Individual Accounts

Table 3.3 lists the seven accounts that we will create. Note that the information in this table could have been collected using request forms.

Table 3.3. Individual account data for the small example company.

Name	Login ID	Initial Password	Login Group
President	prsdnt	111111	management
Vice President	vcprsdnt	222222	management
First Designer	design1	333333	design
Second Designer	design2	444444	design
Third Designer	design3	555555	design
First Secretary	scrtry1	666666	pool
Second Secretary	scrtry2	777777	pool

☐ Login as the super-user.

☐ To begin creating the new user accounts, type:

mkuser

and press the Return key. The screen will display:

```
                    New user
           Add a user to the system

  Do you require detailed instructions? (y/n/q):
```

☐ Type **n** and press the Return key. The screen will display the prompt:

Enter new user's login name:

☐ Type in the user's name (labeled "login ID" in table 3.3), and press the Return key. The following prompt will appear on the screen:

Enter password:

☐ Type in the initial password and press the Return key. We used the value from table 3.3. Type carefully because the characters will not be displayed as they are input. The screen will display a check prompt:

Re-enter for check:

☐ Type the password again. If the two entries match, the following prompt will appear:

Enter group:

☐ Type the user's login group and press the Return key. Table 3.3 contains our response. The screen will display the following:

```
Please enter comment: > ------------------
                      >
```

☐ Type a comment and press the Return key. The comment cannot exceed 20 characters. The screen will display all the new user data just input and prompt for any necessary changes.

☐ If the data is correct, type **n** and press the Return key. The screen will display:

You have 5 seconds to press DELETE if you have made an error.

After the delay, several messages pertaining to the creation of the account will be shown. Finally, the screen will display the following prompt.

```
Do you want to add another (y/n/q):
```

☐ Type **y** and press the Return key if another ID needs to be created. Create the next ID in a fashion similar to the one just detailed. When all the necessary ID's have been created, type **n** and press the Return key to exit the **mkuser** utility.

To check that the users were added to the correct groups, type:

```
cat /etc/group
```

and press the Return key. This action will cause the group information, as discussed in the last section, to be displayed on the screen. Check to be sure the users were added to the correct groups. The individual accounts are now created.

PROFILE FILES

Whenever a user logs into the XENIX system, two files are executed. The first is the **/etc/profile** file. This file sets the general system environment. The other file is the user's own **.profile** file. This file sets the individual user's personal environment. Each of these files is discussed in detail below.

The /etc/profile File

The **/etc/profile** file will include the following commands after the XENIX system has been installed onto an IBM PC XT:*

* The commands that set the terminal type may vary for systems installed on computers other than the IBM PC XT.

```
SHELL=/bin/sh
PATH=.:/bin:/usr/bin:/etc
TZ=PST8PDT
eval 'tset -m : \ ?ibm -e -s -r'
export PATH SHELL TERM TERMCAP TZ
stty -tabs
```

The first three statements in the file set system variables. System variables are used to define how the system operates. The variable **SHELL** defines which interface is to be presented to the user. In this case, the **SHELL** variable is set to the standard XENIX interface, the **sh** shell. The variable **PATH** defines where the operating system should look for commands and files. In this instance, the search path includes the following directories:

```
.
/bin
/usr/bin
/etc
```

The XENIX system will search each of these directories for a file (commands are just executable files in the XENIX system). If the file is found, the search ends successfully. If the file is not found in any of the directories encompassed by the search path, the search ends unsuccessfully. The variable **TZ** defines the time zone in which the terminal is located. In this case, the time zone is set to Pacific Standard Time, 8 hours past Greenwich Mean Time, Pacific Daylight Time.

The **eval** command in this file is used to set the terminal type. The terminal type is tentatively set to **ibm.** The user may make a null response (press the Return key only) to specify an IBM terminal. Alternatively, the user can type in a different terminal specification.

The **export** command passes the specified system variables to all shells created from the current login. The **stty** command is used to tell the system not to expand the tabs as spaces for this terminal.

The .profile File

When a new user is added to the system, the **mkuser** utility creates a **.profile** file in that user's directory. This **.profile** file contains the following:

```
# Copyright Microsoft Corporation, 1983
# User $HOME/.profile - commands executed at login time
#
PATH=.:$HOME/bin:/bin:/usr/bin:                 # set command search path
MAIL=/usr/spool/mail/'basename $HOME'           # mailbox location
umask 022                                       # set file creation mask
export PATH MAIL
```

Any text preceded by a "#" is a comment. The first three lines of the file are comments. The variable **PATH** is changed to a search path suitable to an individual user. Note that the **/etc** directory was deleted and the **$HOME/bin** directory was added. The **/etc** directory contains files that are useful to the super-user but generally are of no use to the individual user. The **$HOME/bin** directory is the user's home directory. The variable **MAIL** defines the location of the user's mailbox.

The **umask** command sets the file creation mask so that the group and others do not have write permission for new files. The **export** command once again passes the system variables to all shells. Note that the **PATH** variable is changed from the value that it has after the **/etc/profile** file was executed.

MODIFYING THE PROFILE FILES

The profile files can be changed to conform to specific needs or inclinations. The **TZ** variable should be changed to conform to the local time zone. We will also change the terminal settings in the profile files.

We return to the small example company to justify the logic behind these changes in the terminal settings. This company has the system console and two terminals available for use. Both of the terminals are Digital vt 100's. The secretaries and the vice president usually use the system console. The design team and the president generally use the terminals. Figure 3.5 illustrates this situation.

We will let the terminal setting command in the **/etc/profile** file specify the system console (**ibm**) automatically. The terminal type will be set correctly by the execution of the **/etc/profile** file for the two secretaries and the vice president. The **.profile** file for the three designers and the president will have to be modified to set the terminal type to **vt100** (represents the Digital vt100 to the XENIX system).

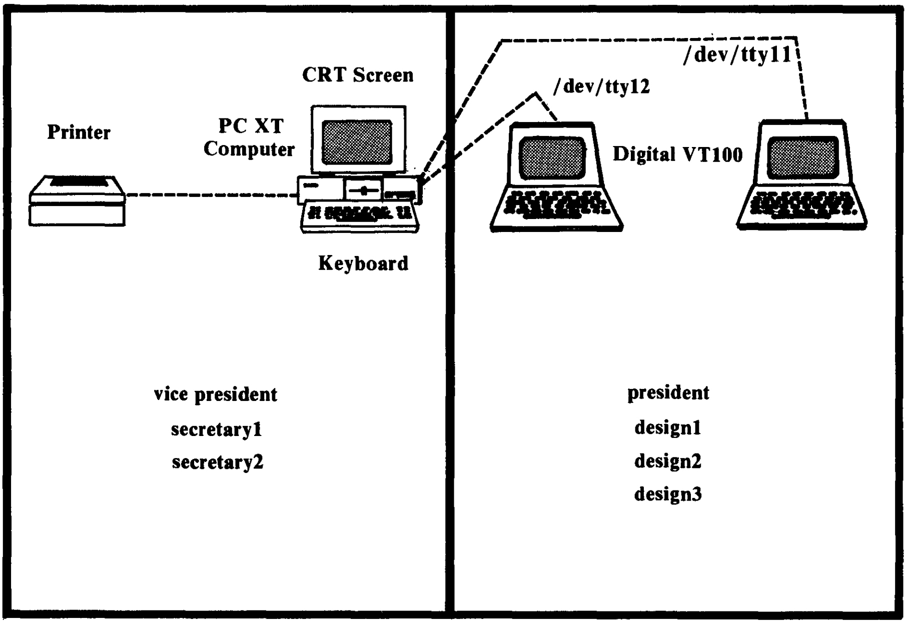


Figure 3.5. Usual usage of computer hardware within the example company.

/etc/profile

To accomplish the modification described above, perform the following steps:

☐ At the super-user prompt (#), type:

```
ed /etc/profile
```

and press the Return key.

☐ At the line editor prompt (*), type:

```
1,$p
```

and press the Return key. The screen will display the **/etc/profile** file.

☐ At the asterisk, type:

```
-3p
```

and press the Return key. The screen will display:

```
TZ = PST8PDT
```

☐ Change the value of **TZ** to the correct time zone using the **s** command. For example, in Cleveland we typed:

```
S/PST8PDT/EST5EDT
```

and pressed the Return key. This action caused the **TZ** variable to be changed to our present time zone.

☐ At the asterisk, type:

```
1,$p
```

and press the Return key. This action will cause the edited file to be displayed on the screen. Check the file for accuracy.

☐ If the file is correct, type **w** and press the Return key. This action causes the modified file to be stored into the file system.

☐ Quit the editor by typing **q** and pressing the Return key.

.profile

The four individual profiles for the users who normally use the terminals must be modified. The modification will be identical for each of the files and must set the terminal type to **vt100.** To accomplish this task, we used an editing script. The technique is detailed below:

☐ Create the script. To accomplish this task, edit a new file, **script.** Type:

```
ed script
```

and press the Return key.

☐ At the ed prompt (*), type **a** and press the Return key. Now, type the following lines, each followed by pressing the Return key:

```
1 + 4
a
eval 'tset -m :\ ?vt100 -e -s -r'
period
g/PATH MAIL/s//PATH MAIL TERM/g
w
q
.
-3
s/period/./
```

☐ At the asterisk, type **w** and press the Return key. Then, type **q** and press the Return key. The actions taken above create an editing script for the **.profile** file.

☐ At the super-user prompt, type the following lines, each followed by pressing the Return key.

```
ed /usr/prsdnt/.profile    script
ed /usr/design1/.profile < script
ed /usr/design2/.profile < script
ed /usr/design3/.profile < script
```

☐ Use the **cat** command to check the contents of each file to be certain it was altered correctly. The command to accomplish this task for the **prsdnt .profile** file follows:

```
cat /usr/prsdnt/.profile
```

Now, each time a member of the design team logs onto the system, his terminal type will be set to the Digital vt100. Note that the system will display the following prompt:

```
TERM = (vt100)
```

The user can specify the vt100 terminal simply by pressing the Return key. If a different terminal is being logged into, the user can specify that terminal by typing the terminal type name and pressing the Return key.

The /etc/rc File

The **/etc/rc** file contains the XENIX commands to initialize the system. This file is executed each time the system is started. The **/etc/rc** file can be modified to accommodate the requirements of a particular system.

One addition that should be made to the **/etc/rc** file is the commands to set and export the **TZ** variable. This action should be taken so that the system clock is set to the local time. To accomplish this task, follow the procedure outlined below.

☐ Login as the super-user.

☐ Activate a text editor and specify the **/etc/rc** file to be edited. Type:

```
ed /etc/rc
```

and press the Return key.

☐ At the asterisk prompt, type:

```
14p
```

and press the Return key. The screen should display the following:

```
PATH = /etc:/bin:/usr/bin
```

We will insert the commands to set and export the **TZ** variable into the area where other system variables are set.

☐ At the asterisk prompt, type **i** and press the Return key. Now type the commands to set the time zone and export it to the system. In Cleveland, we typed:

```
TZ = EST5EDT
export TZ
```

Note that the return key was used to get a new line. Type a period (.) on a line by itself to stop inserting lines.

☐ Type **w** and press the Return key. This action causes the updated file to be written to the disk.

☐ Type **q** and press the Return key. This causes an exit from the line editor.

Chapter 4 — Modifying the XENIX Configuration

INTRODUCTION

In this chapter, we will discuss how to modify the XENIX configuration. The configuration must be modified to reflect changes in the system hardware (terminals, printers, etc.). Also, the user accounts must be kept up-to-date. New employees must be assigned user accounts. Accounts for employees that no longer use the system must be closed.

This chapter is designed to aid the person responsible for maintaining and updating a XENIX system. It should be used as a reference guide for updating the system. The logic motivating why each task is performed is discussed. The tasks themselves are presented in the form of checklists which detail the steps necessary to complete the task.

HARDWARE

In this section we discuss how to modify the XENIX configuration to reflect changes in the system hardware. By hardware we mean the peripheral devices, other than the system console, attached to the computer.

Terminals

REMOVING

At times, a serial device other than a terminal may be needed in the system. If all of the available serial ports are in use, one of the ports must be freed by removing the device currently attached. Quite often, a terinal will be temporarily removed from the system to make room for the other device.

To accomplish the removal of the terminal, complete the following steps:

☐ Power off the terminal to be removed.

☐ Login as the super-user on a different terminal.

☐ Disable the special device file associated with the terminal to be removed by using the **disable** command. For example, to disable the terminal associated with the file **/dev/tty11,** type:

```
disable /dev/tty11
```

and press the Return key. The XENIX system manual warns to wait at least a full minute before using the command again. The warning states that failure to do so can cause a system crash.

☐ Disconnect the cable attaching the terminal to be removed from the system.

The removal of the terminal is now complete.

ADDING

A terminal can be added to the system on any of the free serial ports on the computer. Generally, four tasks must be accomplished:

- Enable the special device file
- Attach the terminal to the computer
- Set the baud rate
- Set the terminal type

See the section on terminals in chapter three for complete details of how to accomplish these tasks.

USER INFORMATION

As the personnel using the computer changes, the user accounts will need to be updated. Also, as the company changes, the groups using the computer may need to be changed or restructured. Keeping the user information up-to-date improves security by limiting access to the system to currently authorized users. Also, deleting inactive accounts saves space in the file system.

Groups

REMOVING

If it is ever necessary to remove a group, all of the users who are a part of that group must be changed to another group, or removed from the system. See the section on individual accounts to accomplish these tasks.

Once the group is emptied of accounts, the group can be removed using the following procedure.

☐ Login as the super-user.

☐ Call a text editor. Specify the **etc/group** file to be edited. Type:

```
ed etc/group
```

and press the Return key.

☐ Display the contents of the group file. Type:

```
1,$p
```

and press the Return key.

☐ Locate the group that must be removed.

☐ Make the line that contains the group to be deleted the current line. To accomplish this task, count how many lines above the last line the group to be removed lies. Type:

-*n*

and press the Return key. Where *n* is the value of the number of lines. The screen will display the line that is now the current line. If the displayed line does not contain the group to be deleted, use the + and - commands to find the correct line.

☐ When the line that contains the group to be deleted appears as the current line, type **d** and press the Return key. This action will cause the current line to be deleted.

☐ Check the file by displaying its entire contents once again. Type **1,$p** and press the Return key. The edited version of the file will appear on the screen.

☐ If this file is correct, type **w** and press the Return key. This action causes the edited file to be saved on the disk.

☐ Type **q** to quit the line editor program.

The removal of the group is now accomplished.

ADDING

A new group can be added at any time. The procedure for adding a new group is detailed in chapter three in the section on creating groups. Remember to include all users in the group who have already been allocated ID's and will be members of the new group, or will not have this particular group named as their login group.

CHANGING THE MEMBERSHIP OF AN EXISTING GROUP

New members can be added to an existing group and present members can be deleted. This action can be taken to allow or restrict an individual's access to the group's files.

To accomplish either of these tasks, the **/etc/group** file must be edited. The editing procedure is outlined below.

☐ Login as the super-user.

☐ Enter the line editor and specify the file to be edited. Type:

```
ed /etc/group
```

and press the Return key.

☐ At the line editor prompt (*), type:

```
1,$p
```

and press the Return key. This action causes the contents of the **/etc/group** file to be displayed.

☐ Locate the current line so that it contains the group to be changed. This task can be accomplished by using the + and - commands.

☐ Delete or add the necessary members. This task can be completed using the **s** command.

☐ Check the contents of the **/etc/group** file by typing:

```
1,$p
```

and pressing the Return key. This action will cause the contents of the file to be displayed.

☐ If the edited file is correct, type **w** and press the Return key. This action will cause the edited version of the file to be stored on the disk.

☐ Exit the line editor by typing **q** and pressing the Return key.

The task of changing the membership of a group is now completed.

Individual User Accounts

ID's that are no longer valid should be removed from the system. New users should be issued ID's of their own. Consistently performing these actions allows for tighter control over access to the system, improves system performance by forcing files belonging to invalid ID's to be removed, and simplifies accounting for system usage.

At times, it is necessary to modify an individual ID. The modification might be made to allow the user access to a different set of files or to change the user's login group.

REMOVING

The **rmuser** utility is used to remove user accounts. Before an account may be deleted, all of the directories and files must be removed from the account's home directory. These files may be deleted from the system, saved on floppy diskettes, or moved to other directories within the system. Chapter 9 on the file system discusses how to accomplish these tasks. When the **rmuser** program is run, it removes the account's entry from the **/etc/passwd** file, home directory, and mailbox.

To accomplish the deletion of a user account, follow the procedure outlined below.

☐ Login as the super-user.

☐ Type:

```
cd /usr/user login name
```

and press the Return key. Note that the *user login name* represents the login name associated with the account to be removed. This command causes the working directory to be changed to the user account's home directory.

☐ Check to see which files are in the home directory. Type **ls -a** and press the Return key. This command will cause the names of all the files, including invisible files, contained in the home directory to be displayed.

☐ Determine what must be done with each file. The options include: moving the file to another location within the XENIX file system, completely deleting the file, or saving the file on the back up media (usually floppy diskettes). Information on how to accomplish these tasks is given in chapter 9. The **.profile**, ., and .. files need not be removed.

☐ After the home directory of the account to be removed has been cleared, check the account's mailbox. The mailbox must be empty before the account may be removed. To check the mailbox, type:

```
cat /usr/spool/mail/user login name
```

and press the Return key. The contents of the mailbox will be displayed. If the mailbox contains any messages, type:

```
cat /dev/null > /usr/spool/mail/user login name
```

and press the Return key. This action will cause the mailbox to be cleared.

☐ Move to another directory. The account cannot be removed while it is a working directory. Type:

```
cd /
```

and press the Return key.

☐ The account can now be removed using the **rmuser** program. Type:

```
rmuser
```

and press the Return key. The screen will display a message containing instructions on the use of the **rmuser** program. The following prompt will appear after the instructions:

```
Press ENTER when you are ready:
```

☐ Press the Return key. The screen will display the following prompt:

```
Enter name of ID to be removed:
```

☐ Type the user login name of the account to be removed and press the Return key. The screen will display the following confirmation prompt:

Removing user *login name* **from the system. CONFIRM ? (y/n/q):**

☐ If the user name is correct, type **y** and press the Return key. The screen will display:

```
User sally removed from the system
Do you want to remove another user? (y/n/q):
```

☐ Answer the prompt in the appropriate fashion. A response of **n** will cause the super-user prompt to appear.

ADDING

New user accounts are added to the system using the XENIX **mkuser** utility. We detail all the facets of adding new users, from collecting the pertinent information to actually using the **mkuser** program, in chapter three.

ALLOWING ACCESS TO ANOTHER GROUP

An individual user may need to be added to a group. This action will allow the user access to the group's files. We detail how to accomplish such a task in the section on groups, included earlier in this chapter.

CHANGING A USER'S LOGIN GROUP

If a user finds that he must usually change his group from the default login group, the login group can be changed. To accomplish this task, the **/etc/passwd** file must be edited. Before starting the editing procedure, the **/etc/group** file should be viewed to find the group ID number of the new login group. To accomplish this task, type:

```
cat /etc/group
```

and press the Return key. This action will cause the contents of the group file to be displayed. Each group entry has the form:

```
group name :: group ID : users
```

Locate the group which is to be the user's new login group and note the group ID. The actual procedure to edit the **/etc/passwd** file is outlined below:

- ☐ Login as the super-user.
- ☐ Make the working directory the **/etc** directory by typing:

```
cd /etc
```

  and pressing the Return key. This action is taken for the sake of convenience.

- ☐ Make a copy of the **/etc/passwd** file by typing:

```
cp passwd passwd+
```

  and pressing the Return key. This action is taken so that a back up of the **/etc/passwd** file is immediately available. The **/etc/passwd** file contains information necessary to access the system. The loss of this file would cause major problems.

- ☐ Enter a text editor with the specification to edit the **/etc/passwd+** file by typing:

```
ed passwd+
```

- ☐ Display the contents of the **passwd+** file. At the asterisk prompt, type:

```
1,$p
```

and press the Return key.

☐ Use the + and - commands so that the current line contains the user's password entry to be changed. Each entry has the form:

```
name : xxxxxx : user ID : group ID : ....
```

☐ Use the **s** command to change the group ID to the previously noted value.

☐ Repeat the previous two steps for all user ID's that must be changed.

☐ Save the edited copy of the **passwd+** file on the disk. At the asterisk prompt, type **w** and press the Return key.

☐ Exit the line editor by typing **q** and pressing the return key at the asterisk prompt.

☐ Compare the edited version of the file with the original by typing:

```
diff -e passwd passwd+
```

This command will cause all the lines that are different in the two files to be displayed along with some other information. Check and be sure that the only differences are the changes that were intended.

☐ If the changes check out, overwrite the **passwd** file with the **passwd+** file by typing:

```
mv passwd+ passwd
```

and pressing the Return key.

The login group of the user account has now been changed.

CHANGING THE MESSAGE OF THE DAY

The message of the day is displayed after each successful logon. It is initialized to display a message similar to the following:

```
          Welcome to XENIX
           for the IBM XT

          Brought to you by
        The Santa Cruz Operation
```

This message can be changed to display another message. A typical use of the message of the day is to notify all users of scheduled system maintenance. To change the message of the day, use the following procedure:

☐ Login as the super-user.

☐ Create the new message using a text editor. Type:

```
ed
```

and press the Return key.

☐ At the asterisk prompt, type **a** and press the Return key.

☐ Type in the new message. Use the Return key to obtain a new line. Use a period (.) by itself on a line to stop adding to the message. A typical message may read as follows:

MESSAGE TO ALL USERS:

The system will be shut down at 1:00 p.m. on Friday, July 27, for normal system maintenance. The computer will be available for regular use starting the following Monday.

☐ Overwrite the old message with the new one. Type:

```
f /etc/motd
```

and press the Return key. This action tells the editor which file should be written to. The screen will display the file name. Type **w** and press the Return key to actually write the file to the disk.

☐ At the asterisk prompt, type **q** and press the Return key. This action will cause the super-user prompt to appear on the screen.

☐ Check the new message. Type:

```
cat /etc/motd
```

and press the Return key. The new message should be displayed on the screen.

Chapter 5 — XENIX System Security

INTRODUCTION

In this chapter, we discuss topics related to the security of the system. By system security we mean limiting access to the computer to authorized users only. System security also entails limiting individual user access to only certain parts of the file system. This aspect of system security is handled by the access permissions of the XENIX file system. We discuss the XENIX file system in chapter nine.

PHYSICAL SECURITY

The first step in a good security system is guaranteeing that unauthorized people do not have physical access to the computer or the attached terminals. This point may seem trivial, but it is an important link in the total security picture.

SUPER-USER PASSWORD

Privileges

The super-user password controls access to the root account. The root account is granted many privileges that none of the user accounts enjoy. Some of these special privileges are listed below:

- Read, write, and execute access to system files. Examples include execution of the **mkuser** program and editing of the **etc/group** file. User accounts are, at the most, given read access to these files.
- Access to all the user files. The super-user can read, execute, write and delete any user file in the system.
- Exclusive (single-user) control of the maintenance mode.

Confidentiality

It should be obvious that the **root** account virtually owns the entire XENIX system. So, whoever knows the super-user password, which allows access to the **root** account, can examine and alter any of the files stored within the system. For this reason, the super-user password should be kept safe and confidential. Only the person responsible for the system maintenance tasks needs to know the super-user password. However, common sense dictates that the super-user password should be recorded in a safe place. Then, if for unforseeable circumstances, the system maintenance person cannot communicate the password, it will not be lost.

Restoring a Lost Super-User Password

The only way to restore a lost super-user password is to reinstall the entire XENIX operating system. The super-user password can then be initialized. We discussed system installation in chapter two. All the individual user accounts and files would then need to be regenerated from the back up media.

Use of the root Account

System files can be inadvertently altered or lost by a few careless keystrokes while using the **root** account. For this reason, the **root** account should only be used for specific tasks, such as system maintenance, which require the root account's special privileges. A task should be planned, as much as possible, in advance. The task should then be undertaken, following the advance plan. The **root** account should be logged off immediately upon completion of the task.

Changing

The super-user password should be changed periodically. The length of the period will depend upon individual preferences and circumstances. If there is ever the slightest suspicion that the security of the super-user password has been violated, the password should be changed immediately. To change the super-user password, follow the procedure outlined below.

☐ Login as the super-user.

☐ Type:

```
passwd root
```

and press the Return key. The screen will display the following:

```
Enter new password (minimum of 5 characters)
Please use a combination of upper and lower case letters and numbers.
New password:
```

☐ Type in the new password. Type carefully because the screen will not display the characters as they are typed. Press the Return key after entering the new password.

- ☐ The system will prompt for the password to be entered again for a check. The screen will display the following:

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

- ☐ Type the password again. Remember that the characters will not be displayed on the screen. If the two entries match, the password will be changed and the super-user prompt will appear on the screen. If the two entries do not match, a prompt to start over will be displayed on the screen.

USER ACCOUNT PASSWORDS

The individual user is permitted to change his own password. Under normal circumstances, one password is allowed to stay in effect until the user decides to change it. Individual users should be instructed to keep their passwords secure and to change them periodically.

If a certain individual is working on a particularly sensitive project, he can be forced to change his password at a set interval. This technique is called **password aging.** The **pwadmin** utility is used to accomplish the task of monitoring that user's password. The procedure to control password aging is detailed below.

- ☐ Login as the super-user.
- ☐ Type:

pwadmin -min *n1* **-max** *n2 user login name*

where *n1* and *n2* represent numbers. The value of *n1* determines the minimum time interval, in weeks, that a password may be in effect. The value of *n2* determines the maximum time interval, in weeks, that a password may be in effect. After pressing the Return key, the super-user prompt will appear on the screen. Password aging is now in effect for the specified user.

Several other options of the **pwadmin** utility are detailed below:

- Disable password aging:

  pwadmin -n *user login name*

- Force a new password to be selected the next time (once only) the user logs onto the system:

  pwadmin -f *user login name*

- Information about the current settings of password aging:

  pwadmin -d *user login name*

ENCRYPTING FILES

The final measure of security that can be taken is the encryption of files. Encryption scrambles the text of the file using a key supplied by the user. The key is a string of up to 8 characters. This scrambling of the text transforms the file into gibberish. The file must be restored before it can be read. In order to restore the file, the key that was used to encrypt the file must be known.

Files can be encrypted by individual users. However, the encryption and restoration processes require a significant amount of machine time. Encryption should be reserved for files that absolutely must be kept confidential.

To encrypt a file, follow the procedure outlined below.

☐ Decide on the encryption key. The key can be any string of up to 8 characters. Do not lose the encryption key. The file cannot be restored to readable text without the key.

☐ Encrypt the file. To do this, a new file containing the encrypted version of the file must be created. The following example will encrypt a file named **critical** into a file named **critical.encrypt.** To accomplish this task, type:

```
crypt < critical > critical.encrypt
```

and press the Return key. The screen will display the following prompt:

```
Enter key:
```

- ☐ The encryption key must be entered. Type the encryption key and press Return. Type carefully because the screen will not display the characters as they are entered.
- ☐ Check the encryption. To view the encrypted file, type:

```
cat critical.encrypt
```

and press the Return key. The file should appear as gibberish when it is displayed on the screen.

- ☐ Check the key to be sure that the file can be restored. Type:

```
crypt critical.encrypt
```

and press the Return key. The screen will display the following prompt:

```
Enter key:
```

- ☐ Type in the encryption key and press Return. Remember that the characters will not be displayed as you type. If the key was correct, the screen will display the contents of the original file. If gibberish is displayed, try to restore the file one more time. If gibberish is displayed a second time, a typo was made when entering the original encryption key. Start the encryption procedure over.

☐ When the encrypted file can be restored, remove the original file. Type:

```
rm critical
```

and press the Return key.

Before an encrypted file can be read or edited, it must be restored. To restore a file, follow the procedure outlined below:

☐ Restore the file. To do this, a new file containing the readable version of the file must be created. The following example will restore the file **critical.encrypt** into the file **critical.** Type:

```
crypt < critical.encrypt > critical
```

and press the Return key. The screen will display the following prompt:

```
Enter key:
```

☐ Type the encryption key used to encrypt the file and press the Return key. Remember that the characters will not be displayed as they are typed.

☐ Check the result. Type:

```
cat crticial
```

and press the Return key. If the encryption key was entered correctly, the screen will display the original file in readable form.

Chapter 6 — Maintaining the XENIX System

INTRODUCTION

In this chapter we discuss the tasks that should be performed on a periodic basis to keep the XENIX system in good operating order. This chapter is designed as a guide for the person responsible for system maintenance. We present a schedule for the periodic maintenance tasks after describing each task in detail. Finally, we discuss how to correct certain system problems when they occur. Problems discussed include nonechoing terminals, jammed line printers and system crashes.

PERIODIC MAINTENANCE TASKS

Maintaining Free Space

Free space in a XENIX system can be defined as the space that is available throughout the file system for use. Each file that is added to the system uses a certain amount of free space. When the free space available in the file system becomes small, the XENIX operating system will operate more slowly. The system manuals recommend that 15% of the total disk space be kept free.

Space in the XENIX file system is measured in blocks. One block is 1024 bytes of data in length. To determine the total number of blocks available in the root system, look up the number of cylinders on the hard disk that are devoted to XENIX. This information was needed during the installation procedure and should be recorded in chapter two. Use the following equation to calculate the approximate total number of blocks available:

__________ # of XENIX cylinders x 32 = __________ blocks available

The total number of blocks in the file system will be necessary to calculate the percentage of blocks that are free.

CHECKING FREE SPACE

The **df** command can be used to check on the amount of free space left in the system. The free space will be reported as a number of blocks. The command has the form:

df *special file*

where *special file* represents the device file corresponding to a disk drive containing a XENIX file system. If *special file* is not specified, the **df** command will cause the free space of all normally mounted file systems to be displayed.

For example, suppose we used 250 cylinders of the hard disk for the XENIX partition. The total number of blocks available in this partition is approximately:

250 x 32 = 8,000 blocks

If we issued a **df** command by typing:

```
df
```

and pressing the Return key, the screen would display the following:

```
(/dev/root):   3186 blocks   1305 i-nodes
```

The percent free space may be calculated as:

3186/8000 x 100 = 40%

So, at the present time, there is plenty of free space in the system.

Alternatively, the critical number of blocks that must be free in order to stay above the 15% constraint could be calculated. The output from the **df** command could then be compared directly to this number. For our example, the critical number of blocks can be calculated as shown below:

15% = (critical # of blocks)/8000 x 100
(critical # of blocks) = 1200

Generally, there is enough free space in our system if the **df** command returns a value for the number of free blocks greater than 1200.

COMBATTING LACK OF FREE SPACE

If the amount of free space falls below the critical amount, some action should be taken to boost the free space back to a healthier level. The available paths of action include:

- Send a message to all users asking that all unneeded files be removed.
- Locate and remove all temporary and **core** files, clear system log files.
- Contact by mail users who are taking a large amount of space in the file system or own files that have not been accessed recently. Ask the user to clean out any unnecessary files.
- Create and mount an additional file system. This technique is detailed in chapter 9.

Each of these points will be discussed in the following sections. Note that the actions are listed in order of increasing difficulty. The simpler solutions should be tried first. If they do not work, more drastic action will need to be taken. The final option of mounting another file system should be taken only if the present system is chronically short on free space.

MESSAGES TO ALL USERS

There are two ways to send a message to all users in the XENIX system. The first is via the message of the day and the other is via the **wall** command. The first method guarantees that all users will see the message but not until the user logs on. The second method only sends the message to those users who are currently logged in. However, it has the advantage of delivering the message immediately.

The details of changing the message of the day are covered in chapter 4. The message should be similar to the following:

```
We are currently experiencing a shortage of disk space.
Please remove any unnecessary files.
Thank you for your cooperation.
```

To send a message to all currently logged in users, type:

```
wall
```

and press the Return key. Type the message. Press the Return key whenever a new line is needed. When the message is completed, press the control-d key to send it to all currently active terminals.

REMOVING TEMPORARY AND core FILES

Temporary files contain data that was created as an intermediate step by a program. A temporary file can be left in the file system if the program terminated prematurely because it contained an error or the user stopped it. A **core** file contains a copy of a terminated program. The XENIX system creates a core file for a program that terminates with an error.

Locating temporary files can be difficult because temporary files are named by the program that created them. The best method for removing temporary files is to depend on the individual users to delete temporary files. A message of the day could be displayed periodically that reminds the individual user to look for and remove temporary files.

Locating core files is a fairly straightforward procedure. Type:

```
find /usr -name core -atime +7 -print
```

and press the Return key. This command finds all core files that are older than one week and displays them on the screen. The aging period of seven days allows the individual user a chance to access the core file before it is removed from the system. The removal of the listed files can be accomplished using the **rm** command.

CLEARING SYSTEM log FILES

System log files contain information pertaining to system usage. The XENIX system appends new information to that which already exists in the file. These files can grow quite large if they are not cleared periodically. Before clearing any of the log files, be sure none of the log information is needed. The log files can be cleared using the **cat** utility. Type:

```
cat < /dev/null > file name
```

and press the Return key. Note that **/dev/null** is not a real device. It is used as a place to discard unwanted files and to get null input.

The XENIX system log files include the following files:

- **/etc/ddate** — This file records the date of each backup. If this log file is cleared, the next system backup will be a periodic backup. We discuss system backups later in this chapter.
- **/usr/adm/pacct** — This file records accounting information. When process accounting is being used, this file will grow rapidly.
- **/usr/adm/sulog** — This file records each use of the **su** command when the option to keep track of the **su** command is active.
- **/usr/adm/wtmp** — This file records user logins and logouts.
- **/usr/spool/at/past** — This file records each use of the **at** command.
- **/usr/spool/micnet/***remote machine's name***/log** — This file records transmissions between computers in a Micnet network. This file is added to only if there is an active Micnet.

Which log files should be cleared will depend on which options are currently in use within the system.

CONTACTING INDIVIDUAL USERS

Individual users who own files that seem excessively large or files that have not been accessed recently can be singled out for a message via the mail system. To decide which users should receive such messages, larger user directories and inactive files must first be identified.

FINDING LARGE USER DIRECTORIES AND FILES

The **du** command is useful for identifying those users with large directories. The command has the form:

```
du directory
```

where *directory* represents any valid XENIX directory. The directories for individual users have names with the form:

```
/usr/user login name
```

For example, to check the directory of the user with a login name of **sally,** type:

```
du /usr/sally
```

and press the Return key. The screen might display:

```
4          /usr/sally
```

This message indicates that the user with the login name of **sally** is presently using only four blocks of disk space.

Alternatively, the **quot** command could be used to accomplish the same task. Type:

```
quot
```

and press the Return key. This command will cause a summarization of file system ownership to be displayed on the screen. The output will be displayed in two columns. The first column indicates how many blocks are being used. The second column indicates which account is using these blocks. The entries will be arranged in order with the account using the largest number of blocks appearing at the top.

LOCATING UNUSED FILES

Files that have not been used recently should probably be removed from the system. If some of these files pertain to monthly or quarterly chores, they can be stored on backup media. These files will then only be loaded into the system as they are needed.

To locate a seldom used file, follow the procedure outlined below.

☐ Decide on a reasonable maximum time since a file's last use. We will use one week for this example.

☐ Locate files that are older than this maximum age. Type:

```
find /usr -atime +7 -print
```

and press the Return key. This command will cause all user files that have not been accessed in the past seven days to be displayed on the screen. The files will be listed one to a line using their complete path name. Some of the files that appear in the list may be system files. Do not remove any system files.

SENDING MAIL TO SELECTED USERS

After determining those users whose files are excessively large or inactive, send them a message via the **mail** system. The message should ask them to remove unneeded files. To accomplish the task of sending **mail,** use the procedure outlined below.

- ☐ Create two message files, one file should contain the message pertaining to the use of a large amount of file space. The other file should contain the message about unused files. Use one of the XENIX editors to accomplish this task.

- ☐ To send one of these messages to a user, type:

  mail *user login name* < *file name*

  and press the Return key. The parameter *file name* specifies the name of the file that contains the message that is to be sent. The parameter *user login name* specifies to whom the message should be sent.

System Back Up

We define system back up as the copying of important files onto a secondary storage media, such as diskettes. Comprehensive back ups can prevent catastrophic loss of important files due to a system failure, user errors, or deliberate sabotage.

The XENIX system provides two methods of creating system back ups. The first is the **sysadmin** program. This program automatically creates a system back up. The extent of the back up can be controlled by the user. The second method is a manual technique. The user specifies exactly which files are to be copied to the back up media using the **tar** utility.

STRATEGY

On large UNIX systems with many users, a partial system back up should be performed each day. This daily backup copies all files that were changed during the day. Periodically, usually once a week, the entire system should be backed up.

For the smaller XENIX system, such a thorough back up job may not be necessary. However, the extra measure of caution cannot hurt. The XENIX manuals suggest that the system be completely backed up at least once a month.

We suggest that a daily backup be done once a day. The frequency of the complete back up depends on individual circumstances. However, twice a month seems reasonable for most installations.

In our opinion, the **tar** command, used to manually choose the files to be backed up, should not be used as the primary means of creating a system back up. The **sysadmin** program should be used.

FORMATTING DISKETTES

Both the **sysadmin** and the **tar** utilities require formatted diskettes. Floppy diskettes can be formatted quite simply using the **format** utility. The procedure for formatting a diskette is outlined below.

☐ Access the **format** program by typing:

```
format
```

and pressing the Return key. The screen will display the following messages:

```
Insert a floppy in drive 0

Hit return when ready
```

☐ Insert a diskette and press the Return key. Note that formatting a diskette destroys any data previously contained on it. While the formatting is taking place, the screen will display the message:

```
formatting /dev/format0...
```

When the formatting is complete, the screen will display:

```
formatting /dev/format0... done
```

Also, the shell prompt will be displayed. If the diskette is not inserted properly or if no diskette is present in the drive, an error message will be displayed on the screen.

PROCEDURE — sysadmin

In this section, we discuss the procedure for using the **sysadmin** program to create a system backup. This program can be used to create either a complete backup or a daily backup. The daily backup will make copies of all files that have been altered during the present day. The complete back up is referred to as a periodic back up by the **sysadmin** utility. The first time the program is run, a complete back up will be performed, regardless of whether a periodic or a daily backup is specified.

To accomplish the task of making a system backup, use the procedure outlined below.

☐ Check to be sure an adequate number of formatted diskettes are available. After having backed up the system a few times, you will find it fairly easy to determine how many diskettes are needed.

☐ Login as the super-user.

☐ Access the **sysadmin** program by typing:

```
/etc/sysadmin
```

and pressing the Return key. The screen will display the following:

```
File System Maintenance

Type 1 to do daily backup,
     2 to do periodic backup,
     3 to get a backup listing,
     4 to restore a file,
     q to quit

Enter Number:
```

☐ Type the number corresponding to the backup operation which needs to be performed and press the Return key. The screen will display the following prompt:

```
PERIODIC (DAILY) BACKUP
Insert first disk in drive 0, then press ENTER
```

☐ Insert a diskette into the floppy drive and press the Return key. The screen will display various messages pertaining to the progress of the backup.

☐ The message:

```
estimated xxxxK on xx volume(s)
```

will appear. Each x represents a numeral. Check and be sure that enough formatted diskettes are on hand. The number of volumes is equivalent to the number of diskettes that will be required.

☐ When the message:

```
Change volumes
```

appears, wait until the floppy drive's light goes off. Then, remove the diskette presently in the drive and replace it with the next diskette. Press the Return key after loading the diskette into the drive. If the diskette is removed before the floppy drive's light goes off, drive error messages will appear on the screen and data transfer to the diskette will not be completed in a proper fashion. If any drive error messages are received while creating the backup, start the backup process over again.

☐ Be sure to number each diskette, starting with one, so that a particular diskette can be easily referenced at a later time.

☐ Repeat the above steps until the screen displays the message:

```
DONE
```

When this message is displayed, the system backup is complete.

☐ Check and be sure that the number of diskettes used is the same as the number of volumes displayed in the message immediately following the appearance of DONE:

```
xxxxK on xx volume(s)
```

Where each x represents a numeral. If the number of disekttes is different than the number of volumes displayed in the message, an error was made. The backup procedure must be repeated.

LISTING THE CONTENTS OF A BACKUP

A listing of the files contained on the backup diskettes can be obtained using the **sysadmin** utility. A backup listing should be obtained for each backup and stored with the backup diskettes. The procedure for obtaining a backup listing is detailed below.

☐ Login as the super-user.

☐ Enter the **sysadmin** program by typing:

```
/etc/sysadmin
```

and pressing the Return key. The file system maintenance menu, as shown in the last section on page 113 will appear.

☐ At the prompt to enter a number, type **3** and press the Return key. This action selects the option to generate a backup listing. The following messages will appear on the screen:

```
PRODUCE BACKUP LISTING
Insert first disk in drive 0, then press ENTER
```

☐ Load the first diskette of the backup into the floppy drive and press the Return key.

☐ The screen will display:

```
LIST IS IN /tmp/backup.list
```

☐ Output a hard copy of the backup list by typing:

```
1pr /tmp/backup.list
```

and pressing the Return key. This command causes the file containing the backup listing, **/tmp/backup.list,** to be printed on the line printer.

☐ After checking the copy of the backup listing, remove the file containing the backup listing from the system. The file is no longer needed. Type:

```
rm /tmp/backup.list
```

and press the Return key.

RESTORING A FILE FROM A BACKUP

If a file is lost or its contents scrambled by an error, that file must be restored. The file should be restored from the most recent backup. Any changes in the file made since the backup will not appear in the restored file. However, having this slightly outdated version of the file is better than having none.

To restore a file, the **sysadmin** program must be used. The procedure for accomplishing file restoration is outlined below.

☐ Decide which file or files must be restored.

☐ Find these files on the backup listing.

☐ Retrieve the set of diskettes relating to the backup listing on which the file or files to be restored appear.

☐ Login as the super-user.

☐ Invoke the **sysadmin** program. Type:

```
/etc/sysadmin
```

and press the Return key. The file system maintenance menu will appear on the screen. This menu is listed on page 113.

☐ Select the option to restore a file by typing 4 and pressing the Return key. The screen will display the following instructions:

```
RESTORE FILE(S)
Type full path name of files to restore.
One per line, blank line to terminate
Enter pathname:
```

☐ Type the full pathname of each file that needs to be restored. Press the Return key after each file is entered. The full pathname for each file appears in the backup listing. Pressing the Return key when the current line is blank will signal that all the required file names have been entered. When this signal is received, the screen will display:

```
Insert first disk in drive 0, then press ENTER
```

☐ Load the first diskette of the backup into the floppy drive. Press the Return key. The screen will display the following:

```
full path name : inode xxx
        .
full path name · inode xxx
        .
Mount desired dumpvolume: specify volume #:
```

where *full path name* represents the file to be restored and *xxx* represents a three-digit inode number. The XENIX system uses inode numbers internally to organize the file system.

☐ If the volume number that the file appears on is known, load the diskette with that number into the drive, type that number, and press the Return key. If the volume number is not known, leave volume one loaded in the floppy drive, type **1** and press the Return key.

☐ If all of the files are not found on the current volume, the screen will display the message:

```
Mount volume x:
```

where *x* represents the next volume number.

☐ Remove the diskette currently located in the drive and load the diskette corresponding to the requested volume number into the floppy drive. Press the Return key.

☐ Repeat the last step until all the necessary files have been found. When a file is found, the screen will display the following:

```
extract file xxx
```

where *xxx* represents one of the inode numbers shown earlier. When all the files have been extracted, the program will terminate, and the super-user prompt will be displayed on the screen. The extracted files will be stored in the directory from which the **sysadmin** program was invoked. This directory will usually be /. Each file is stored in a file having its inode number for a name. For example, a file with the full path name **/usr/john/test** and an inode number of 320 would be stored as **/320** by the restore option of the **sysadmin** program. These inode numbers appear on the backup listing. Each file must now be stored in its correct place in the file system. In the above example, file **/320** should be used to overwrite or recreate the file **/usr/john/test.**

☐ Find the full path name and inode number for each file that is being restored. This information can be found on the backup listing.

☐ Check the / directory for each file. Remember that the files will be named by their associated inode number. Type **l** and press the Return key to display directory contents.

☐ For each file being restored, type:

```
mv /inode number full path name
```

and press the return key. For example, the file **/320** referenced above would be restored by typing the following:

```
mv /320 /usr/john/test
```

and pressing the return key. This command causes the contents of the file **/320** to be moved to a file named **/usr/john/test**. The file **/usr/john/test** is created if it does not exist or overwritten if it does exist. The file **/320** no longer exists after this command.

☐ Check the / directory to be sure that all of the extracted files have been moved to their proper location.

MAINTENANCE SCHEDULE

Performing system maintenance tasks on a regular schedule is the best way to keep the system in good order. The schedule we suggest here is not particularly rigorous. However, it should be quite adequate for any XENIX installation on a personal computer.

Daily

The following tasks should be performed once each day:

UPON ARRIVAL

- Power on computer and all the peripheral devices.
- Set the correct date and time by initiating the normal start up procedure and answering the time prompt.

BEFORE LEAVING

- Create a daily backup using the **sysadmin** program. File a copy of the backup listing with the numbered and dated diskettes.
- Check the free space in the system. If it is low, take corrective action.
- Shut down the system.

Periodic

The items included in this list of tasks should be performed on a regular basis. We suggest twice a month as a reasonable interval. The fifteenth and the last day of the month could be designated as the days for periodic maintenance duties. The following list details the tasks that should be undertaken:

- Perform a periodic backup using the **sysadmin** program. File a copy of the backup listing with the numbered and dated diskettes. The periodic backup makes any previous backups superfluous. The information stored by the daily backups is also stored by the periodic backup. So, the diskettes occupied by any previous daily backups can be recycled. The diskettes containing the oldest backups should be recycled first.

- Perform the steps detailed in the section on combatting lack of free space. Taking this action twice a month can locate and correct problems before they start to affect system performance.

SYSTEM PROBLEMS

Occasionally, the XENIX system may suffer from problems. The severity of these problems can range from a user who forgot his password to a totally inoperable system. We discuss how to solve some of these problems in the following sections.

System Crash

A system crash is a breakdown which causes the computer to stop all work. A system crash can be caused by hardware malfunctions or software inconsistencies. We will not concern ourselves with system crashes caused by hardware failures. If a hardware failure is suspected, run the diagnostic routines provided by the manufacturer or have a qualified service representative check the machine.

The most common cause of a system crash is an interruption or inconsistency in the machine's power source. Other nonhardware causes of a crash include lost or damaged system files.

A system crash has occurred whenever the system displays a message beginning with "panic:" on the ystem console. Also, if the system will not respond to any input at any terminal, a system crash has taken place. Any panic messages should be recorded before any action is taken.

RECOVERY

The first step trying to recover from a system crash is to look up the meaning of any panic error messages produced just before the crash. The messages (M) section of the XENIX operating system reference manual details this information. If the cause of the error is discernible, corrective action should be taken.

The second step in trying to recover from a crash is to try to boot the system. This step should be undertaken even if no error messages were generated at the time of the crash. To accomplish this task, try to start the system in a normal fashion.

FATAL DAMAGE

If the system crashes upon being started or will not start at all, the XENIX operating system has suffered irreparable damage. The entire system must be reinstalled. Start by installing the distribution floppies as we discussed in chapter 2. After the system is reinstalled, recover the rest of the file system from the latest periodic system backup.

If the system cannot be booted from the "Boot Floppy" disk, the computer has a major hardware malfunction.

NON-FATAL DAMAGE

If the system does start and remains running, the following message will be displayed on the screen after the system check procedures have been completed:

```
The system was not shut down properly,
and the root file system should be cleaned.

Proceed with cleaning (y/n)?
```

Type **y** and press the Return key. The screen will display several messages about the progress of the cleaning operation. When the cleaning procedure is complete, the system will either be shutdown or the screen will display the message about normal system startup. At this point, restart the system or continue with the normal startup procedure.

Nonechoing Terminal

A nonechoing terminal is a terminal that does not display characters that are typed at the keyboard. The system manual suggests the use of the following routine to restore such a terminal to normal:

- ☐ Press control-j. Ignore any error messages that are displayed on the screen.
- ☐ Type **stty sane** and press the control-j key again. The terminal will not display what you type. After pressing the control-j for a second time, the system manual claims that the terminal should be restored.

We found that this procedure for freeing a nonechoing printer does not always work. If the terminal cannot be restored via the above procedure, check for a runaway process blocking the terminal. We cover runaway processes later in this chapter. If there is no runaway process, the terminal can be disabled and then enabled. Remember to allow at least one minute to elapse between uses of the **disable** and **enable** commands.

Jammed Line Printer

A jammed line printer is one that will not print the current file or any of the other files in the print queue. A common cause of printer jam up is the error caused when the printer runs out of paper.

To free the jammed line printer, use the following procedure:

- ☐ Login as the super-user.
- ☐ Find the process identification number (PID) of the print spool program, **lpd**. Type **ps -a** and press the Return key. The screen will display the following column headings:

```
PID TTY TIME COMMAND
```

☐ Locate the **lpd** command in the list following the header. Note the PID associated with the **lpd** command.

☐ Type:

```
kill PID
```

where *PID* represents the process identification number noted above. This action causes the **lpd** program to abort.

☐ Fix the error that caused the jam. For example, load the printer with paper if it is out of paper.

☐ Remove the lock file from the line printer spool. Type:

```
rm -f /usr/spool/lpd/lock
```

and press the Return key. The line printer is now free.

☐ Issue an **lpr** command to cause the printer to commence printing.

Runaway Process

A runaway process is a process that cannot be stopped from the terminal at which it was started. This occurs when an error in the program prevents what is typed at the terminal from reaching the operating system. To stop such a process, follow the procedure outlined below.

☐ Login as the super-user at another terminal.

☐ Get information about the processes that are currently active. Type:

```
ps -a
```

and press the Return key. The screen will display the following column headers:

```
PID TTY TIME COMMAND
```

The information corresponding to each header for each process listed, will appear below these headings.

☐ Locate the process that is causing the problem by using the TTY (terminal), Time (total amount of execution time the process has used so far), and COMMAND (command name and all its options) information. The runaway process must be associated with the terminal that is experiencing difficulty. The total amount of execution time associated with the process will probably be high. The user should be able to supply the name of the command that caused the problem.

☐ Once the problem process has been located, stop it by typing:

kill *PID*

and press the Return key. *PID* represents the process ID number listed by the **ps -a** command.

☐ If the **kill** command does not solve the problem within a few seconds, try typing:

kill -9 *PID*

and pressing the Return key. This version of the **kill** command is guaranteed to stop the process. However, it may leave the terminal in a nonechoing state and unneeded temporary files in the system.

Forgotten Password

If an individual user forgets his password, a new password must be made for him. To accomplish this task, follow the procedure outlined below:

☐ Login as the super-user.

☐ Invoke the **passwd** utility by typing:

passwd *user's login name*

and pressing the Return key. The screen will display the following prompt:

```
Enter new password (minimum of 5 characters)
Please use a combination of upper and lowercase letters and numbers.
New password:
```

☐ Type the new password and press the Return key. The characters will not be displayed as you type them. The system will prompt for another entry to check the password. If the two entries match, control will return to the shell. The super-user prompt will be displayed.

☐ Give the user the new value of his password.

XENIX Chapter 7 — Effect of Additional Memory on Performance

INTRODUCTION

In this chapter, we discuss how adding additional RAM to the computer will affect the performance of the XENIX system. This chapter is meant to aid the person who must decide if the cost of adding more RAM to the system is justified.

Note that the discussion concentrates on XENIX for the IBM PC XT. The specifics will be different for XENIX on other machines. However, the same general trends should still be valid.

XENIX WITH LESS THAN 384K

When the computer posesses less than 384K of RAM, the smaller implementation of XENIX must be used. We found this smaller XENIX to be frustratingly slow. Complex utilities, such as the screen editor **vi,** do not work well under the smaller XENIX.

In our opinion, the XENIX system should not be installed on a system that does not possess at least 384K of RAM. The full XENIX can be installed when at least 384K of RAM is present.

Expanding the amount of RAM past 256K requires a memory expansion board for the IBM PC XT. Most memory expansion boards allow for the addition of 384K of RAM, in 64K increments. This 384K coupled with the 256K of RAM that can be installed on the system board results in a maximum allowable memory for the PC XT of 640K.

The expansion board and two 64K increments are needed, in addition to 256K of RAM on the system board, to obtain a total of 384K RAM within the system. Many popular third party expansion boards include a serial port. This port brings the total number of serial ports to the maximum allowable of two. Inclusion of a serial port on the expansion board saves the expense of purchasing a separate serial port and makes maximum use of the available expansion slots.

XENIX WITH 384K OR MORE OF RAM

We have already concluded that at least 384K of RAM should be installed on the computer. But, how does adding memory beyond 384K enhance the performance of the system? To answer this question, we developed and ran a test routine. We tested each 64K increment up to the maximum of 640K for the IBM PC XT.

The test routine was developed only as a means of comparison. The routine uses shell programming to accomplish the necessary tasks. Note that these shell programs are inefficient and accomplish little if any useful work. However, they are quite adequate for the purposes of comparison.

The program **multiscount** was used to gather the information. The **multiscount** program executes a program named **scount** as a background process, five times. **scount** was written so that it would use a significant amount of execution time. The **scount** program follows:

```
date '+START: %H:%M:%S:' > $2
begin=0 end=10
echo "$begin" > $1
while test "$begin" -lt "$end"
do
   begin="'expr $begin + 1'"
   echo "$begin" >> $1
   sort -nr $1 > $3
   mv $3 $1
done
date '+STOP: %H:%M:%S:' >> $2
mail sully < $2
```

The **multiscount** program listing appears below:

```
date '+START: %H:%M:%S' > $1
scount count1 time1 temp1&
scount count2 time2 temp2&
scount count3 time3 temp3&
scount count4 time4 temp4&
scount count5 time5 temp5&
date '+STOP: %H:%M:%S' >> $1
```

Note that both of the programs use the **date** utility to mark the time when they begin and end execution. We executed the **multiscount** program with different amounts of RAM resident in the computer. We allowed no other jobs to be submitted for execution while the experiment was being conducted. This precaution was taken to ensure that the amount of RAM was the only variable. Table 7.1 summarizes the time required to execute the **multiscount** and **scount** programs. Figure 7.1 summarizes the data graphically.

Note that the **multiscount** program took only a short time to execute, because it simply started the **scount** program as a background process five times and then terminated.

Table 7.1. Results from simple multitask test (Execution time min:sec).

	TOTAL RAM				
	384	**448**	**512**	**576**	**640**
multiscount	00:13	00:15	00:14	00:12	00:07
scount - time 1	10:21	08:35	08:05	07:06	05:50
scount - time 2	10:21	08:04	08:40	07:00	06:07
scount - time 3	10:07	09:30	07:48	06:58	05:59
scount - time 4	10:19	08:04	07:38	06:50	06:14
scount - time 5	10:20	08:54	08:46	06:38	05:48
scount - average	10:16	08:37	08:11	06:54	06:00
	200	264	328	392	456
	RAM AVAILABLE TO THE USER				

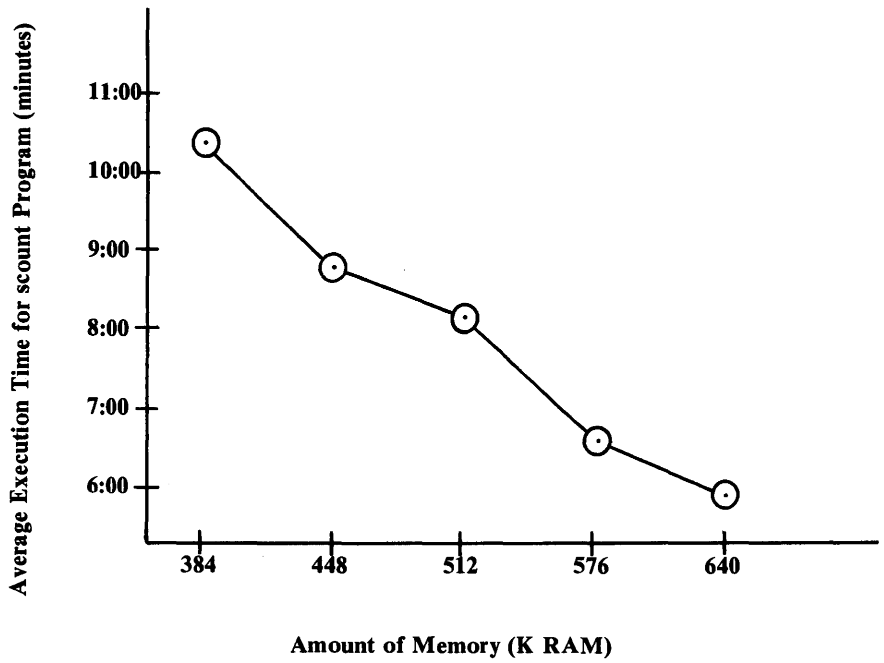


Figure 7.1. Execution time vs. memory for **multiscount** experiment.

From the results of our simple experiment, we can conclude that increasing the amount of memory decreases the execution time for a set of background processes. This effect will also help speed up the system's reaction time in a multiuser environment. Also, increasing the amount of RAM in the computer increases the amount of user available RAM. The full XENIX operating system occupies 184K. All memory added after 384K becomes available to the user.

It should be noted that adding more memory will not help cure some problems. More memory will not help if the system performance is sluggish because there is not enough free space in the file system. Also, additional memory will not reduce long file access time caused by a lengthy search path or an inefficiently organized file system.

Chapter 8 — XENIX Basics

INTRODUCTION

This chapter is intended as an introduction to XENIX for beginning users. In this chapter, we discuss XENIX terminology, the XENIX user interface, and how to start using the system.

XENIX TERMINOLOGY

In XENIX each system user is called an individual user or user. Each user has his own account. Each account is specified by a unique name called the user login name.

XENIX provides many ready-to-use programs. These programs are often referred to as utilities. There are utilities that perform many common tasks, such as sorting a file. These utilities save the user time by allowing him to concentrate on his real job instead of on how to implement that job.

The case of each letter in XENIX is important. XENIX will not recognize a name if the case is different from that specified by the system. For example, the names **distance, Distance,** and **distancE** are three separate and discrete entities in a XENIX system.

Arguments and options in XENIX commands are separated by a space. Commas or semicolons are not used as delimiters. The space is necessary, and the XENIX operating system will not recognize the command without it. For example, the command:

```
cat /etc/motd
```

causes the contents of the file **/etc/motd** to be displayed on the screen. However, the command:

```
cat/etc/motd
```

causes the following message to be displayed on the screen:

```
cat/etc/motd: not found
```

Whenever a "not found" error message appears, check the command for an omitted space or other typographical error.

While the error messages provided by the XENIX operating system may be short, they are to the point. The source of the error can often be determined with a little consideration. As experience is gained, these messages will become easier to decipher.

GETTING STARTED

The first step in getting started is to obtain a user ID. Different companies will have different policies on how to request an ID. But, all companies should provide a login name and a password when they issue an ID.

To log into the computer system, the user terminal screen must display the system login message. A typical login message is shown below:

```
xenixt!login: □
```

where the □ indicates the position of the cursor. At the login message, type the provided login name. The screen will display the following prompt:

```
Password:
```

Now type the provided password. Type carefully because the screen will not display the characters of the password as they are typed. If everything was entered correctly, the screen will display the message of the day. If some error was made, the screen will display:

```
Login incorrect
Login:
```

The login procedure must now be started again by entering the login name.

If the screen displays a prompt similar to the following:

```
TERM = (name)
```

where *name* represents a terminal name such as **ibm** or **vt100,** press the Return key.

When a login is executed, the user is placed in his own home directory. This directory is determined from the user's login name. The user's home directory is where most of an individual user's files will be stored. Generally, the user has complete control over all files appearing in his home directory. He can create, modify and destroy files in his home directory.

The XENIX system is ready to accept commands when the shell prompt is displayed. The standard shell prompt is a dollar sign ($). The shell concept is covered more extensively in the next section.

One task that may be of interest is changing the password to something that is more easily remembered. To accomplish this job, type:

```
passwd
```

and press the Return key. The screen will display the following prompt:

```
Changing the password for login name
Old password:
```

Type the present password. The screen will not display the characters of the password as they are typed. If the present password was entered correctly, the screen will display the following:

```
Enter new password (minimum of 5 characters)
Please use a combination of upper and lowercase letters and numbers.
New password:
```

Enter the new password and press the Return key. Remember that it will not be displayed on the screen. The screen will display a prompt to verify the new password:

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

Type the new password again. Recall that the case of each letter is significant. If the two entries match, control will return to the shell. The password has now been changed.

XENIX SHELLS

In XENIX, the interface the operating system presents to the user is referred to as a shell. An interface is the point of contact between the user's world and the machine's internal world. We shall discuss the **sh** shell. The **sh** shell is the standard XENIX shell command interpreter.

The major function performed by the **sh** shell is the interpretation of commands typed on the command line. The command line is the line immediately following the shell prompt, usually a dollar sign. XENIX utilities can be invoked from the command line. Utilities accessed in this manner carry out their specified tasks and return any output generated during their execution to the standard output, usually the screen. In later chapters we will discuss how to change the destination of output generated by a utility. The technique to accomplish such a change is called redirection.

INDIVIDUAL USER ENVIRONMENT

The XENIX operating system sets up an environment for each user when a login is executed. This task is accomplished by executing a special file stored in each user's home directory. The special file is named **.profile.** The **.profile** file is a hidden file. That is, it will not normally appear in the listings of the contents of the home directory. All hidden files begin with a period. The **.profile** file should never be removed from the home directory. The contents of a typical **.profile** file is shown below:

```
# Copyright Microsoft Corporation, 1983
# User $HOME/.profile - commands executed at login time
#
PATH=.:$HOME/bin:/bin:/usr/bin:             # set command search path
MAIL=/usr/spool/mail/'basename $HOME'       # mailbox location
umask 022                                   # set file creation mask
export PATH MAIL
```

There are two other hidden files that are automatically created in each user's directory. These files are the . and .. files. These two files are both used as directories. The .. directory is used by the system to find the directory preceding the present directory. The . directory is used by the system to find subdirectories of the present directory. Directories are discussed in more detail in chapter nine. Neither the . nor the .. directory should be removed.

Chapter 9 — The XENIX File System

INTRODUCTION

In this chapter, we discuss the XENIX file system. We highlight some important structural details, explain how to search for specific files, and how to move throughout the file system. File handling is covered along with file ownership.

This chapter is intended as a tutorial on using the XENIX file system for all users. Novices may want to skip the section on advanced topics. Advanced topics discussed include mounting new file systems, copying files to floppy diskettes and creation masks.

HOME DIRECTORY

Each individual user has his/her own home directory. The user is placed in his/her home directory immediately after logging on. Any files owned by the individual user will appear in his/her home directory.

STRUCTURE

Directories and Files

The XENIX file system has a tree structure made up of directories and files. Figure 9.1 contains an illustration of the XENIX file system. Files contain the information that is actually stored in the file system. Directories contain individual files or other directories. The directories are used to organize the file system while files are used to store data.

Note in figure 9.1 that files are terminal nodes in the tree. That is, no branches lead out from the file. The box labeled **xenix** is a file. Notice that a line does not link the box labeled **xenix** to any of the boxes that appear lower in the diagram.

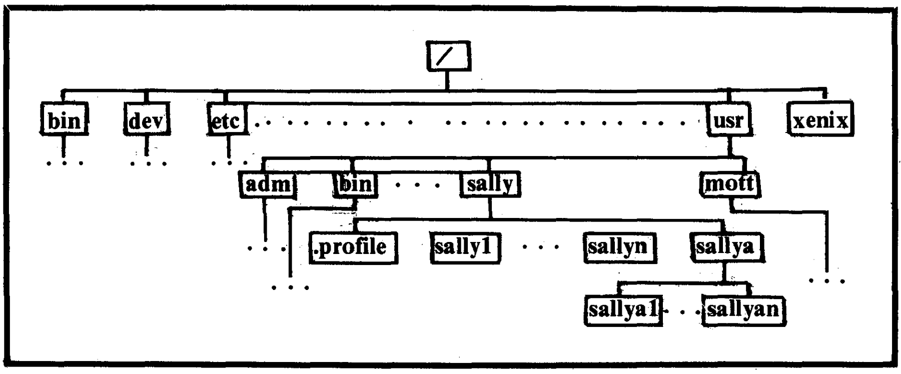


Figure 9.1. Simplified illustration of the XENIX file system.

Directories appear in figure 9.1 as boxes with branches leading out of them to boxes appearing lower in the diagram. For example, the box labeled / has branches that lead to many other boxes. The box labeled / represents the root directory.

Full Path Names

To specify a particular file or directory in the XENIX file system, a full path name is used. A full path name is an expression that uniquely specifies a file or directory. A full path name always starts from the root directory. Each directory that appears in the tree structure between the root directory and the desired file directory is then listed, successively. After the root directory, all subsequent names appearing in the full path name are separated by a slash. The slash used as a separator should not be confused with the name of the root directory, which is also /. For example, the full path name for the file represented by the box labeled **sallya1** in figure 9.1 would be expressed as follows:

/usr/sally/sallya/sallya1

Note that this convention for specifying files allows files that appear in different directories to share the same name. Study figure 9.1 carefully; note that the name **bin** is used twice. However, the two occurrences of **bin** are distinguishable from each other because they have different full path names. One has the full path name:

/bin

The other's full path name is:

/usr/bin

This distinguishability is convenient. Individual users need not worry if another person has already used a name when creating a file. As long as a user has not previously used the name for one of his own files, the file will have a unique name specified by the full path name.

Search Paths

Specifying a full path name for each file or directory that is needed could become a tedious task. Search paths are used to simplify name specifications. A search path is a list of partial full path names.

Each of the partial path names begin at the root directory. When a file or directory is specified with something other than a full path name, the first partial path name in the list is added to the specified name. This action creates a full path name. This full path name is searched for. If the specified file directory is found, the search ends, successfully. If the file or directory is not found, the next partial path name in the search path is used. If all of the partial path names in the search path are exhausted and the file or directory has still not been found, the search terminates unsuccessfully. An error message stating this fact will appear on the screen.

The search path for each individual user is specified in his **.profile** file. A search path typically includes the following partial path names:

- **.** — The current working directory.
- **$HOME/bin** — The user's file that should contain any executable (command) files.
- **/bin** — System commands available to the user.
- **/usr/bin** — More system commands available to the user.

A search path that includes the above entries allows the user to access XENIX utilities and his/her own files by merely typing the appropriate name.

DIRECTORIES

Directories can contain files and subdirectories. The current working directory is the directory in which a user presently resides. The current directory is set to the user's home directory when he or she logs on. We will discuss how to change the current directory later.

Displaying Directory Information

XENIX has three commands that will cause information about a directory to be displayed, **l**, **ls**, and **lc**. All three commands are similar. We will discuss only the **ls** command.

Type **ls** and press the Return key. This action will cause the names of all the regular files and directories contained in the current directory to be displayed. One name will be displayed on each line. If the current directory is empty, the message "total 0" will be displayed.

Information about a directory other than the current directory can be obtained by including that directory's name as an argument to the **ls** command. For example, to display the contents of the root directory on the screen, type **ls /** and press the Return key. The screen will display the following:

```
bin
dev
etc
lib
lost+found
mnt
once
stty
tmp
usr
xenix
```

Note that the names are alphabetically sorted.

More information about each file or directory can be obtained. The **-l** option causes a long version of the **ls** output to be generated. This long version will have the format illustrated in figure 9.2. The access permission for each file is broken into three categories. Each category can be allowed or denied read, write, and execute permission. A dash in the permission field indicates that the permission is denied. An "r" indicates read permission is allowed. A "w" indicates write permission is allowed. And, an "x" indicates execute permission is allowed.

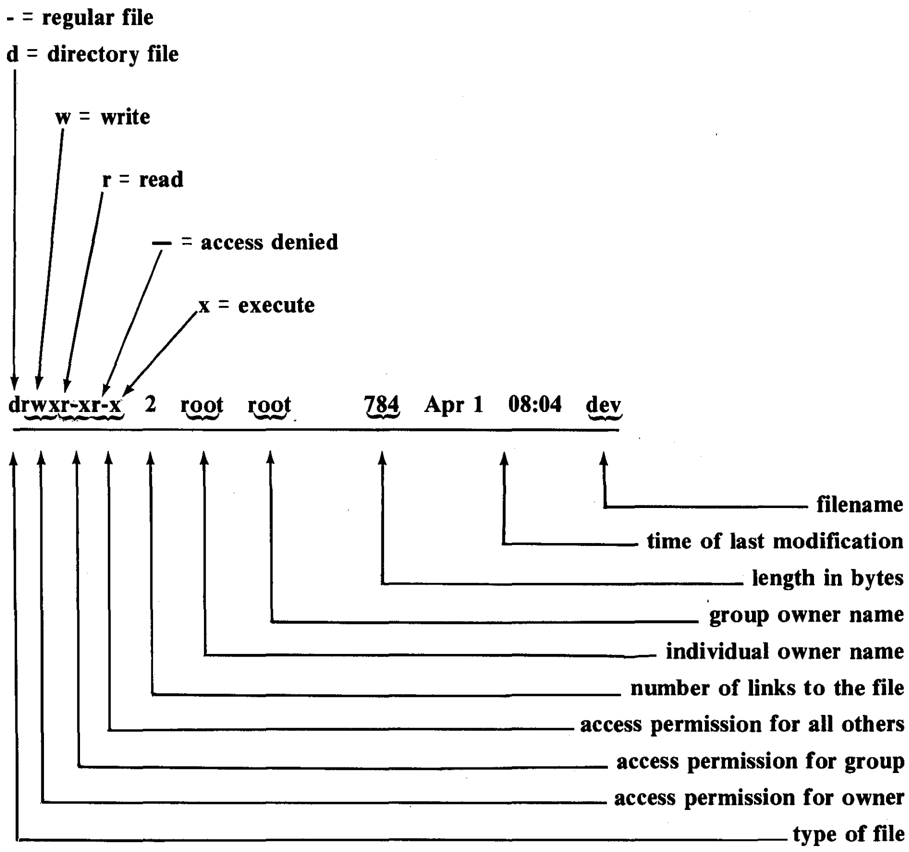

- = regular file
d = directory file
w = write
r = read
— = access denied
x = execute
drwxr-xr-x 2 root root 784 Apr 1 08:04 dev
filename
time of last modification
length in bytes
group owner name
individual owner name
number of links to the file
access permission for all others
access permission for group
access permission for owner
type of file

Figure 9.2. Format of the output from an **ls -1** command.

In the example in figure 9.2, the file is a directory. The file's owner, root, has read, write and execute permission to the file. Execute permission is equivalent to search permission for a directory. The file's group, also root, has only read and execute permission to the file. All other users also have only read and execute permission to the file, also.

To display the long version of the **ls** command, type **ls -1 /** and press the Return key. The screen will display the following:

```
total 125
drwxr-xr-x     2 bin    bin      1824 Apr   1 08:05 bin
drwxr-xr-x     2 root   root      768 Apr   1 08:04 dev
drwxr-xr-x     3 bin    bin       784 Apr   1 13:31 etc
drwxr-xr-x     2 bin    bin        48 Apr   1 07:58 lib
drwxr-xr-x     2 root   root     1328 Apr   1 08:07 lost+found
drwxrwxrwx     2 root   root       32 Apr   1 08:04 mnt
drwxrwxrwx     2 root   root       48 Apr   1 08:07 once
-rw-rw-rw-     1 root   root       31 Apr   1 17:37 stty
drwxrwxrwx     2 root   root       96 Apr   2 01:37 tmp
drwxr-xr-x    15 bin    bin       240 Apr   1 13:31 usr
-rw-r--r--     1 bin    bin    113856 Apr   1 00:00 xenix
```

In addition to the **-1** option, several other options are available. These options may be used to sort the display in a different manner (by time of last modification, reverse order, etc.), to give slightly different information (inode number, size in blocks) or to include hidden files in the display. Hidden files are those that start with a period. The **.profile** file is an example of a hidden file.

Changing Directories

Movement within the XENIX file system can be accomplished by changing the current working directory. Changing directories is most useful when extensive work is to be done in a subdirectory of the home directory. Changing the directory to a lower level will save specifying the higher level directories in the path name each time a file is accessed.

For example, in figure 9.1 **sally** has the subdirectory **sallya.** If **sally** needed to perform extensive work on the files **sallya1** ... **sallyan,** she could change the current directory to **sallya.** Now, only the file name need be specified instead of the partial path **sallya/***filename.*

The **cd** command causes the current working directory to be changed. The command has the form:

```
cd directory name
```

where *directory name* is the name of the directory to be changed to. The directory name can be either a full path name or a partial path name that can be found in the search path.

For example, to go to a higher directory, the full path name must be given. To get to the command directory, **/bin,** type **cd /bin** and press the Return key. The cd command does not generate a message on the screen. To list the contents of the new working directory, use the **ls** command. To return to the home directory, type **cd** and press the Return key. The **cd** command with no argument specified will always return the working directory to the user's home directory, no matter where in the file system the current working directory is located. To move a subdirectory such as **sallya** in figure 9.1, type **cd sallya** and press the Return key.

An important point to remember is that changing the current working directory causes the search path to be changed. Each directory has a different **.** entry. This dot entry provides the search path not included in the **/bin** and **/usr/bin** portion.

If there is ever any confusion as to which directory is the current working directory, the **pwd** utility can be used to display the present working directory on the screen. The command has the form:

```
pwd
```

The command causes one line of output. The value of the output is the full path name of the current working directory.

Creating a Subdirectory

An individual user can arrange his own directory by creating subdirectories. These subdirectories can be used to store related files. In this manner, data relating to a particular job can be isolated in a single subdirectory. For example, a person in charge of keeping records for several different stores could create subdirectories named after each store. Each subdirectory would contain the data files for that store, such as volume, personnel, and inventory files.

The **mkdir** command is used to create a directory. This command has the following form:

mkdir *directory name*

where *directory name* represents the name, unique within the user's current directory, of the new directory. For example, to create a subdirectory for a store located in Lakewood, the user could type:

```
mkdir lkwd
```

and press the Return key. The **mkdir** utility displays no messages if the directory is created.

If the user does not have write access to the current working directory and attempts to create a directory, the following error message will be displayed on the screen:

```
mkdir: cannot access.
```

The directory will not be created. Generally, individual users should create subdirectories only in their home directory.

Removing a Subdirectory

Subdirectories that are no longer needed should be removed from the system. This removal provides the system with more free storage space.

The **rmdir** utility can be used to remove an empty directory. An empty directory contains no files except the hidden files. The command has the following form:

```
rmdir directory name
```

where *directory name* represents the name of the directory to be removed. The **rmdir** command is considered the safe way to delete a directory because each individual file must be removed before the directory can be erased. A more risky command is the recursive remove:

```
rm -r directory name
```

This command causes all the files and subdirectories in the specified directory that is to be deleted. The directory itself will then be removed.

FILES

XENIX files can contain only data. This data can be executable programs, text, numbers or any other valid data type. In a working XENIX system, the files will substantially outnumber the directories.

Creating a File

Files can be created in many different ways. The most common method of creating a file is through the use of a text editor. Another common method of creating a new file is through a program that opens a file, writes data to that file, and eventually closes that file. When a file is created, it will be entered into the current working directory. If the user does not have permission to write in the current working directory, an error message will appear on the screen and the file will not be created. Generally, individual users should create files only in their home directory.

File Ownership

Each file has two levels of ownership. The first level is individual ownership. Individual ownership is assigned to a single user. That single user has the most control over the file. The other level of ownership is by the group. The users in the group that own a file often have more access to the file than other users.

When a file is created, its ownership is set. The user who created the file is the owner. That user's group is the group owner.

Changing File Ownership

The ownership of a file can be changed to a different user and group. Such an action is necessary when a person is assigned to take over a task from the individual presently performing it. The ownership of all the files pertaining to the particular task must be changed to the new person. Only the present owner or the system administrator can change the ownership of a file.

The users receiving the file ownership must cooperate with the user relinquishing ownership of the file. A general outline for accomplishing a transfer of file ownership is outlined below. Note that this task could be accomplished more easily using the **mail** utility.

1. The user receiving the file ownership changes the access permission on his . file to allow the user relinquishing the file to place the file in the receiving user's directory.

2. The user relinquishing the file ownership changes the ownership of the file.

3. The user relinquishing the file moves the file to the directory of the user receiving ownership of the file.
4. The user receiving ownership of the file changes the group ownership of the file. This step will not be necessary if the two users are in the same group.
5. The user receiving ownership of the file restores the access permission on the . file to its original configuration.

An example is now presented to clarify the details of each task. Throughout the example, the user receiving ownership of the files shall be **mott.** The user relinquishing ownership of the files shall be **sally.**

CHANGING ACCESS PERMISSION

To accomplish the job of changing the permission on the . file, the chmod utility must be used. If **mott** is in the same group as **sally, mott** would type:

```
chmod g+w .
```

and press the Return key. This command allows other members of the group to have write access to **mott's** home directory. If **mott** is in a different group than **sally, mott** would type:

```
chmod o+rwx .
```

and press the Return key. This command allows all users to have read, write, and execute permission to **mott's** home directory.

CHANGING USER OWNERSHIP

The **chown** utility is used to accomplish the job of changing the user ownership of the file. The command has the following form:

```
chown  new owner name  file name  file name  ...
```

where *new owner name* represents the new owner's login name and *file name* represents the name of the file whose ownership is to be changed. For example, to change the ownership of two files, named **sales** and **inventory,** to the user **mott,** the present owner of the files, **sally,** would type:

```
chown mott sales inventory
```

and press the Return key. The ownership of the two files has now been changed to the user **mott** from the user, **sally.** However, these files still appear in **sally's** directory. These files should be moved to **mott's** directory.

MOVING THE FILES

To accomplish this task, the **mv** utility must be used. The **mv** command to accomplish the desired move has the following form:

```
mv file name file name ... directory
```

where *file name* represents the name of the file to be moved and *directory* represents the directory to which the files should be moved. To move the two files, **sales** and **inventory** to **mott's** directory, **sally** would type:

```
mv sales inventory /usr/mott
```

and press the Return key. This command causes the files **sales** and **inventory** to be stored in **mott's** home directory and removed them from **sally's** home directory.

CHANGING GROUP OWNERSHIP

The **chgrp** utility is used to accomplish the task of changing the group ownership. This job need be undertaken only if **sally** and **mott** are in different groups. The chgrp command has the following form:

chgrp *new group name file name file name* **...**

where *new group name* represents the name of the group that is to receive ownership of the file. If **mott** and **sally** are in different groups and **mott's** group name is **acctng, mott** would type:

```
chgrp acctng sales inventory
```

and press the Return key. The group ownership would then be transferred.

RESTORING ACCESS PERMISSIONS

The transfer of ownership is now complete. For security, **mott** should now restore the access permissions on the . file. To accomplish this task, the **chmod** utility must be used, again. If **mott** is in the same group as **sally, mott** would type:

```
chmod g-w .
```

and press the Return key. This command denies other members of the group write access to **mott's** home directory. If **mott** is in a different group than **sally, mott** would type:

```
chmod o-rwx .
```

and press the Return key. This command causes access to **mott's** home directory to be denied to all users outside of **mott's** group.

Displaying File Contents

Often it is desirable to just view the contents of a file, either to see what the file contains or to check its contents. If the file is small, the contents can be viewed with the **cat** utility. The command to accomplish this task has the form:

```
cat file name
```

where *file name* is the name of the file to be viewed. This command will cause the file to be displayed on the screen. If the entire file will not fit on the screen, the file will rapidly scroll by, until the end is reached. To view a small file, type:

```
cat .profile
```

and press the Return key. This command will cause the **.profile** file of the current working directory to be displayed.

To view long files, the **more** utility provides a more satisfactory display. The **more** utility displays the first screen of data and stops. The user controls scrolling through the file. The command to use **more** utility has the following form:

```
more file name
```

where *file name* is the name of the file to be displayed. For example, to view the long file, **/etc/termcap**, type:

```
more /etc/termcap
```

and press the Return key. The beginning of the **/etc/termcap** file will be displayed on the screen. The last line will display the percentage of the file that has been viewed so far. Typing **d** will cause a scroll of eleven lines up. Typing a **q** will cause an exit from viewing the file.

Copying Files

At times, a copy of an existing file is needed. The **cp** utility is used to accomplish the job of making a copy of a file. The **cp** command has the form:

cp *sending file* *receiving file*

where *sending file* represents the name of the file to be copied, and *receiving file* represents the name of the file to receive the copy. The receiving file will be created if it does not exist. The receiving file will be overwritten if it does exist.

For example, if **joem** and **mikes** are members of the same group, **joem** can obtain a copy of **mikes'** file **results** by typing:

```
cp /usr/mikes/results copresults
```

and pressing the Return key. This command will cause a file named **copresults** to be added to **joem's** directory. Note that **joem** must have read access permission to **mikes'** files in order for the copy to be successfully made.

Linking Files

In the last example, **joem** obtained a copy of **mikes'** file **results.** If **mikes** updated his **results** file, **joem's** copy of the file, **copresults,** would become outdated. To resolve this problem, the two files can be linked.

Only one copy of a linked file is stored in the XENIX system. This file will appear under a different name in the directory of each user who owns a link to the file. Anyone who owns a link to the file can change the file.

The updated version of the file immediately becomes available to all the other users who own links to the file.

For the link to work properly, the access permission of the file must be changed so that the users with links appearing in their directory can modify the file. The **chmod** utility can be used to accomplish the task of modifying the access permission. To allow his other group members to link to the file **results, mikes** should type:

```
chmod g+w results
```

and press the Return key. If this step is not taken, only **mikes** will be able to change the linked file.

The **ln** utility is used to establish a link. This command has the following form:

ln *original* *new*

where *original* represents the name of the file to be linked and *new* represents the name of the new link to that file. For example, **joem** would type:

```
ln /usr/mikes/results mresults
```

and press the Return key to obtain a link to **mikes'** result file in his home directory. The link would appear as **mresults** in **joem's** home directory.

Now, whenever either **joem** or **mikes** changes the linked file, both users will have access to the new, updated version.

Removing a File

When a file is no longer needed, it should be removed. Removing unneeded files frees storage space for other users.

The **rm** command is used to remove files. The user trying to remove a file must have write access for the directory that contains that file. If he does not, an error message will be generated, and the file will not be deleted. Generally, an individual user should remove only files in his home directory. The **rm** command has the following form:

```
rm file name
```

where *filename* represents the name of the file to be removed. More than one file name can be specified. Each file name should be separated by a blank. For example, the following command:

```
rm sales inventory
```

would cause the files **sales** and **inventory** to be removed from the current working directory.

ADVANCED TOPICS

The material that we discuss in this section can be skipped by the novice user. Knowledgeable users will find the information in this section useful for completing specific tasks. Topics included in this section include the use of the creation mask, how to create and mount another file system, how to format diskettes, and how to store specific files on a diskette.

Creation Mask

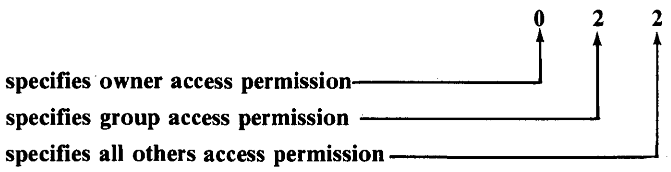
The XENIX system uses the file creation mask to set the access permissions for a file when it is created. The default value for the mask is set in the **.profile** file. The default value of the file creation mask assigns the following access permissions:

files **rw-r--r--**

directories **rwxr-xr-x**

For files, these permissions represent read and write for the file owner and read only for both the group and all others. For directories, these permissions represent read, write and search for the directory owner and read and search for the group and for all others.

The file creation mask can be modified so that a different set of access permissions will be assigned. The user accessible part of the creation mask contains three octal digits. An octal digit can have a value from 0 to 7. One of the three digits controls access permission for the owner, another for the group, and the last for all others. Table 9.1 lists the access permissions related to each octal digit. The default creation mask that is specified in the user's **.profile** file is:


0 2 2
specifies owner access permission
specifies group access permission
specifies all others access permission

This mask specifies read and write permission for the owner and read permission for the group and others.

The **umask** utility allows the individual user to specify a new creation mask. The current value of the creation mask can be checked by typing:

```
umask
```

and pressing the Return key. This command will cause four digits to be displayed on the screen. The rightmost three digits specify the file creation mask as we discussed earlier. The command to change the file creation mask has the following form:

```
umask ugo
```

where *u* represents the access permission for the owner, *g* represents the access permission for the group and *o* represents the access permission for all others.

For example, if the file creation mask has not been changed, typing **umask** and pressing the Return key will cause the screen to display:

```
0022
```

This is the default creation mask that we discussed earlier. If it is desirable to disallow users outside of the group (others) from access to the file, type:

```
umask 027
```

and press the Return key. This new setting of the creation mask will disallow others from accessing any files and directories created while it is in force.

An important point to remember when using the **umask** command is that the command affects the access permissions assigned to files as they are created. The **umask** command does not change the access permission of files already in existence. The **chmod** utility will change the access permission of existing files. Examples showing how to use the **chmod** utility were presented earlier in this chapter.

Table 9.1. Access permissions related to the octal digits in the creation mask.

Octal Digit		Access Permission	Description
0	file	rw-	read and write
	directory	rwx	read, write, and execute
1	file	rw-	read and write
	directory	rw-	read and write
2	file	r--	read
	directory	r-x	read and execute
3	file	r--	read
	directory	r--	read
4	file	-w-	write
	directory	-wx	write and execute
5	file	-w-	write
	directory	-w-	write
6	file	---	none
	directory	--x	execute
7	file	---	none
	directory	---	none

Storing Information on Diskettes

Information that is used infrequently should be stored on floppy disks. This infrequently used data can then be removed from the XENIX file system. This procedure helps free more storage space for use within the file system.

Also, a special diskette containing certain files could be made up to satisfy a specific need. For example, a diskette could be prepared to contain all the files pertaining to a completed project. This diskette could be stored with the project records. The files could then be removed from the system.

FORMATTING DISKETTES

Before a diskette may be used to store files by the XENIX system, the diskette must be initialized. This initialization procedure is known as formatting the diskette. Formatting a diskette destroys any data that was previously stored on the diskette.

The **format** command is used to initialize a diskette. To format a diskette, type:

```
format
```

and press the Return key. The screen will display:

```
Insert floppy in drive
Hit return when ready.
```

The diskette should now be inserted into the floppy drive. After closing the drive door, press the Return key. The screen will display:

```
formatting /dev/format 0 ...
```

When the formatting process is complete, the screen will display:

```
formatting /dev/format 0 ... done
```

and the shell prompt will appear on the screen.

The diskette is now ready to be used to store information.

COPYING SPECIFIC FILES TO A DISKETTE

Specific files can be selected to be copied to a diskette. Before files can be stored on the diskette, a file system must be created on it. The file system is necessary to organize the files on the diskette. The **mkfs** utility is used to create the file system on the diskette. This command has the following form:

/etc/mkfs *device file* *blocksize* *gap* *block*

where *device file* represents the special device file name, *blocksize* represents the size of each block, *gap* represents the gap number for the disk, and *block* represents the block number for the disk. On the IBM PC XT, the name of the floppy drive is **/dev/fd0.** For the standard floppy drive in an IBM PC XT, the command to create a file system on a diskette is:

```
/etc/mkfs /dev/fd0 320 2 8
```

Once the file system has been created, it must be mounted into the XENIX root file system. This action must be taken so that the operating system can assign path names from the **root** directory to the new file system. Without path names, the new file system is inaccessible.

The **mount** utility is used to mount a file system into the XENIX root file system. The documentation manuals indicate that the **mount** utility along with the corresponding **umount** utility (used for unmounting file systems), can be invoked by individual users. However, we found that the **mount** and **umount** utilities are restricted to use by the super-user. Talk to the system manager to find out if the access permissions have been changed.

If the access permissions have been changed to allow individual users to execute the **mount** and **umount** utilities, mount the new file system as outlined below. Otherwise, have the system manager mount it.

A new file system must be mounted into an empty directory. An empty directory is a directory that contains only the hidden files **.** and **..**. Also, since the diskette may have to be used by a variety of people, all files created on the diskette should allow access to every system user. To create an empty directory, use the **mkdir** utility presented earlier in this chapter. To allow access to every system user, use the **umask** utility to change the file creation mask. Specify a creation mask of **000.**

The command to mount a file system has the form:

/etc/mount *device name* *directory name*

where *device name* represents the name of the special device file for the device that contains the new file system and *directory name* represents the full path name of the empty directory through which the new file system is to be accessed. For example, the following command:

```
/etc/mount /dev/fd0 /usr/sally/tempdir
```

would cause the file system on the diskette loaded into the floppy disk drive 0 to be mounted into the directory **tempdir** in the user **sally**'s home

NOTE FOR THE SYSTEM MANAGER

The following two choices regarding the use of the **mount** and **umount** utilities are available:

1. Each time a file system must be mounted or unmounted, login as the super-user and perform the steps necessary to mount or unmount the file system. These steps are detailed in the text.

2. Change the access permission on the **/etc/mount** and **/etc/umount** files so that individual users can execute these files. To accomplish these changes, login as the super-user and issue the following commands:

```
chmod o+x /etc/mount

chmod o+x /etc/umount
```

directory.

The new file system can now be used exactly like any other portion of the XENIX file system. The directory in which the file system is mounted provides the entry point to the new file system. Files can be created on the diskette using text editors, moved to or from the diskette using the **mv** utility, and copied to or from the diskette using the **cp** or **copy** utilities.

When all the tasks using the new file system have been completed, the new file system should be unmounted. The **umount** utility is used to unmount file systems. The command to unmount a file system has the following form:

/etc/umount *device name*

where *device name* represents the name of the special device file for the device that contains the mounted file system. For example, the following command:

```
/etc/umount /dev/fd0
```

would cause the file system contained on the diskette loaded into floppy drive 0 to be unmounted.

After the file system has been unmounted, the file creation mask should be changed back to its original configuration.

Chapter 10 — XENIX Text Editors

INTRODUCTION

A text editor is a program that allows the user to change a file. The process of changing a file is called editing. XENIX users can choose between three text editors. They are named **ex, ed,** and **vi.** The **ex** utility contains elements of both the **ed** and **vi** editors. The **ex** editor is the most difficult of the three to learn. We will not discuss the use of the **ex** editor.

The **ed** utility is a line editor. A line editor allows only one line of the file to be edited at a time. The **ed** utility is the easiest to master. It is useful for small editing jobs and for completing repetitious editing tasks on a group of files.

The **vi** utility is a full screen editor. A full screen editor allows an entire screen, typically 20 to 25 lines, to be viewed at once. Any of the displayed data can be edited. The **vi** editor is more difficult to learn to use. However, the payoff is quicker completion of tasks, in comparison to **ed,** once proficiency in the use of **vi** is gained.

ed — A LINE EDITOR

Invoking ed

The line editor **ed,** can be entered in either of two manners. The first is to simply type **ed** and press the Return key. This command will cause the editor to be invoked. The screen will display an asterisk (*). The asterisk is the line editor prompt. It has a meaning, similar to that of the dollar sign in the standard XENIX shell. All the line editor commands must be entered immediately following the asterisk prompt. This prompt signifies that the **ed** utility is waiting for some command from the user.

The line editor can also be invoked with a specification that a specific file be edited. The command to enter the **ed** utility in this manner has the following form:

ed *file name*

where *file name* is the name of the file to be edited. When **ed** is invoked in this fashion, a number will be displayed on the line before the asterisk prompt. This number represents the number of characters in the file that was specified for editing. If the specified file does not exist, a zero will appear for the number of characters in the file.

One point that is important to note is that the editor does not work directly on the contents of a file. The editor makes a temporary copy of the file being edited in a buffer. A buffer is a temporary storage area. The file being edited is not actually altered until a command to write the changes is issued. In the case where a new file is being created, the file does not exist on the disk file system until such a write command is given.

For example, the following command:

```
ed results
```

would cause the line editor to be invoked and a copy of the file **results** to be loaded into the editing buffer.

Exiting ed

The quit command is used to exit from the **ed** utility. The quit command is issued by typing **q** and pressing the Return key. If changes have been made on the current file and not saved, the screen will display the following message:

```
?
warning: expecting 'w'
```

Issuing another quit command will cause an exit from the **ed** utility without the changes in the file being saved.

The warning message is a safety feature. It helps the user to remember to save all the editing changes made during the editing session. The feature helps to avoid accidental loss of completed work. Note that if a new file is currently being edited and the double quit is used to exit the line editor without saving the file, the new file will not appear in the XENIX file system. The file was never actually created. It existed only as a temporary file in the editor's buffer area.

Saving Edited Files

The write command is used to save the edited copy of the file. When a write command is issued, the file stored in the editor's buffer is used to overwrite the file stored in the XENIX file system. To execute a write command, type **w** and press the Return key. This causes the contents of the buffer to overwrite the currently specified file. When the line editor is invoked with a file to edit specified, that file becomes the currently specified file.

Issuing a write command when no file is currently specified will cause the following error to be displayed on the screen:

```
?
illegal or missing filename
```

FINDING THE CURRENTLY SPECIFIED FILE

The name of the currently specified file can be displayed on the screen using the file command. Type **f** and press the Return key. The screen will display the name of the currently specified file. If there is no currently specified file, no file name will be displayed.

CHANGING THE CURRENTLY SPECIFIED FILE

The currently specified file can be changed to a different file using the file command. This usage of the file command has the following form:

```
f file name
```

where *file name* represents the name of the file that should be made the currently specified file.

For example, the following commands:

```
f newresults
w
```

would cause the currently specified file to be changed to **newresults.** The contents of the editing buffer would then be used to create or overwrite the file **newresults.** When the next write command is issued, the action taken depends on whether the file existed before the write command was executed.

Moving Within the File

The **ed** utility keeps track of which line in the buffer is the current line. Generally, a line is the text or data that is displayed on one line of the screen. The current line is the line that will be affected by any editing commands that are issued. The current line is represented by a period (.). The current line is often called line dot or simply dot. Initially, the current line is set to the first line in the file. This first line is line number one.

Subsequent lines are numbered sequentially. The line number of the current line can be displayed by typing . = and pressing the Return key.
The next line can be accessed by pressing the Return key.

MOVING TO A SPECIFIC LINE NUMBER

The current line can be changed to a different line number by typing a line number and pressing the Return key. The current line will be changed to the specified line number, and the contents of that line will be displayed on the screen. If a line number greater than the total number of lines is entered, the screen will display the following error message:

```
?
line out of range
```

For example, the following command:

```
10
```

would cause the current line to be changed to line 10. Also, the contents of line 10 would be displayed on the screen.

MOVING TO THE LAST LINE

The last line in the buffer can be accessed without knowing the last line number. A dollar sign is used to represent the last line number. Typing a dollar sign and pressing the Return key will cause the current line to be changed to the last line in the buffer. Also, the contents of the last line will be displayed on the screen. The value of the last line number can be displayed on the screen by typing an equal sign and pressing the Return key.

For example, the following command:

```
=
```

would cause the value of the last line number in the file to be displayed on the screen. And, the command:

```
$
```

would cause the current line to be changed to the last line in the file.

MOVING A SPECIFIC NUMBER OF LINES

The + and - commands are used to move a specific number of lines up or down in the buffer. The + command moves the current line closer to the end of the file. The - command moves the current line closer to the beginning of the file. The form of the + and - commands is as follows:

+*number*
or
-*number*

where *number* represents the number of lines the current line is to be changed by. Both the + and - commands also cause the new current line to be displayed on the screen.

If the + command causes the line number to become larger than the last number or the - command causes the line number to become less than one, the following error message will be displayed on the screen:

```
?
line out of range
```

For example, the following command:

```
+5
```

would cause the current line to be changed to the line five lines forward in the file. If the current line was originally line 22, the command given above would cause the current line to be changed to line 27. Similarly the following command:

```
-7
```

would cause the current line to be changed to the line seven lines backwards in the file. If the current line was originally line 30, the command given above would cause the current line to be changed to line 23.

Displaying the File

The print command is used to display the file currently located in the editing buffer. The most useful form of the print command follows:

*first line,last line***p**

where *first line* represents the line number of the first line to be displayed and *last line* represents the line number of the last line to be displayed. All lines between the first line and the last line will also be displayed. The value of *first line* must be less than the value of *last line* for the command to work. Also, the value of the current line is changed to the value of the *last line.* An especially useful version of this command follows:

```
1,$p
```

This command will cause the entire contents of the editing buffer to be displayed on the screen. Another example of the print command follows:

```
-2,+1p
```

This command would cause the following four lines to be displayed on the screen:

- The current line
- The two lines prior to the current line
- The line following the current line

Adding Lines to a File

Lines can be added to a file by either appending them after the current line or by inserting them before the current line. The simplest manner in which to use these two commands is to first locate the current line at the appropriate position in the file. Then the append or insert command may be instituted to add the necessary lines.

APPENDING

The append command adds lines to a file after the current line. Any lines that were in the file after the current line are moved down so that the new material is sandwiched between the old material. To start appending lines to a file, type **a** and press the Return key. The cursor will be moved to the beginning of the next line on the screen. Type the material to be appended. Pressing the Return key will access a new line. To stop appending material to the file, type a period (.) by itself at the start of a new line. The current line will be set to the last appended line.

For example, the following sequence:

```
a
We are adding two lines. This is the first.
And, this is the second.
.
```

would cause the two lines of text to be placed into the file immediately after the current line. Suppose the file originally read:

```
This is the current line.
This is the line following the current line.
```

After the above append command was executed, the file would read:

```
This is the current line.
We are adding two lines. This is the first.
And this is the second.
This is the line following the current line.
```

Note that the current line would actually be set to the last line that was appended to the file. In this case, the current line would read:

```
And this is the second.
```

INSERTING

The insert command functions in a fashion identical to the append command except that the new material is inserted before the current line instead of after. To start inserting lines into a file, type **i** and press the Return key. The cursor will be moved to the beginning of the next line on the screen. Now, text is added as it was with the append command. The lone period will once again signal the end of additional material.

For example, the following sequence:

```
i
We are inserting two lines. This is the first.
And, this is the second.
.
```

would cause the two lines of text to be inserted into the file immediately before the current line. Suppose the file originally read:

```
This is the line before the current line.
This is the current line.
```

After the above insert command was executed, the file would read:

```
This is the line before the current line.
We are inserting two lines. This is the first.
And, this is the second.
This is the current line.
```

Note that the current line would actually be set to the last line that was inserted into the files. In this case, the current line would read:

```
And this is the second.
```

Deleting Lines from a File

The delete command is used to remove unwanted lines from the file. The delete command works in a manner similar to the print command.

Typing **d** and pressing the Return key causes the current line to be removed. The command to remove more than one line has the form:

*first line,last line***d**

where *first line* represents the line number of the first line to be deleted and *last line* represents the number of the last line to be deleted. All lines between the first and the last line will also be deleted. The value of *first line* must be less than the value of *last line* in order for this command to work.

For example, the following command:

d

causes the current line to be deleted. The current line will be changed to the next line in the file. When the last line in the file is deleted, the current line is changed to the previous line, which is the new last line. Also, the following command:

-6,+3d

would cause ten lines to be deleted. The current line, the six lines before the current line, and the three lines after the current line would be deleted.

Replacing Lines of a File

The change command is used to replace lines of a file with new lines of text. The action this command performs is equivalent to deleting the lines to be replaced and inserting the new lines of text.

The command to replace only one line of the file has the form:

line number **c**

where *line number* represents the line to be replaced. After the Return key is pressed to execute this command, the cursor will be located at the beginning of the next line. As many lines as necessary can be added by typing the text as with an append command. Pressing the Return key accesses a new line. A lone period at the beginning of a line signals that no more lines need be added. Note that one line can be replaced be several lines.

The command to replace several lines of the file has the form:

*first line,last line***c**

where *first line* represents the number of the first line to be replaced and *last line* represents the number of the last line to be replaced. All lines between the first line and the last line are also replaced. After the Return key is pressed to execute this command, the new lines that are to replace the specified lines are entered as in the command to replace only one line.

For example, the following sequence:

```
-1,+2c
Replace four lines with this one line.
```

would cause the current line, the line before the current line and the two lines after the current line to be changed to the single line of text. Suppose the file originally read:

```
This will stand.
Some line.
This is the current line.
Some line.
Some line.
This will stand, too.
```

After the above change command was executed, the file would read:

```
This will stand.
Replace four lines with one line.
This will stand, too.
```

Note that the current line would be set to the last line changed. In this case, the current line would read:

```
Replace four lines with one line.
```

Rearranging Lines Within a File

There are two methods of moving lines within the file being edited. The lines can be actually moved to a new location, or a copy of the lines can be transferred to the new location while they also remain at the original location.

MARKING A LINE

The move and transfer commands each require three different lines to be addressed. The simplest procedure to handle all the addresses is to mark the appropriate lines. The lines can then be referred to by the mark instead of by line number. The command to mark a line has the following format:

*line***k***character*

where *line* represents the line number of the line to be marked and *character* represents the lowercase letter that will mark the line. The marked line can then be referred to by the following:

'*character*

where *character* represents the lowercase letter that was used to mark the line.

For example, the following command:

ka

would cause the current line to be marked with an **a.** The marked line can now be referred to by **'a** instead of a line number.

MOVING LINES

The move command causes the specified lines to be moved from their present position to a new location. The move command has the following form:

first line,*last line***m***new line*

where *first line* represents the line number of the first line in the group of lines to be moved, *last line* represents the line number of the last line in the group of lines to be moved, and *new line* represents the line number of the line after which the lines being moved will be inserted.

For example, suppose we have the following file in the editing buffer:

```
one
two
three
four
five
six
seven
eight
nine
ten
eleven
twelve
```

Also suppose we want to move the following lines:

```
two
three
four
```

so that they appear after the line depicted below:

```
eleven
```

First, we will mark the necessary lines. We will then use the marked lines to specify the line numbers in the move command. The following table lists the necessary commands with a short explanation of their purpose:

Command	Purpose
2kb	Mark the line containing "two" with the letter "b".
4ke	Mark the line containing "four" with the letter "e".
11kr	Mark the line containing "eleven" with the letter "r".
'b,'em'r	Move the lines containing "two", "three", and "four" to between the lines containing "eleven" and "twelve".

The resulting file would appear as follows:

```
one
four
five
six
seven
eight
nine
ten
eleven
two
three
four
twelve
```

In a longer file where the line number cannot be easily determined, the necessary lines can be marked by first making the line of interest the current line by using the + and - commands. The marks can then be made using the current line as the default for the line number in the mark command.

TRANSFERRING LINES

The transfer command causes the specified lines to be copied at a specific location. The lines that were copied will appear at their current location as well as the new one. The transfer command has the following form:

first line,*last line*t*new line*

where *first line* represents the line number of the first line in the group of lines to be transferred, *last line* represents the line number of the last line in the group of lines to be transferred, and *new line* represents the line number of the line after which the lines being transferred will be inserted.

For example, suppose we wanted to accomplish the same task as with the example for the move command, except as a transfer. The file would be marked in the same manner. In place of the move command, we would issue the following transfer command:

'b,'et'r

As a result of this command, the file would appear as follows:

```
one
two
three
four
five
six
seven
eight
nine
ten
eleven
two
three
four
twelve
```

Note the difference in the results of the two commands.

JOINING TWO LINES

Contiguous lines may be joined to form one line. This task is accomplished by removing the new line character. The join command has the following form:

first line,*last line***j**

where *first line* represents the line number of the first line to be joined and *last line* represents the line number of the last line to be joined.

For example, suppose we have a file that contains the following lines:

```
one
two
three
```

Issuing the following command:

```
1,3j
```

will cause the file to be changed to the following:

```
onetwothree
```

Note that no space was provided for between the joined lines. We will discuss how to remedy this problem later.

Substitutions Within the Current Line

The substitute command allows an individual character or a string of characters to be changed within a line. The command to accomplish this task has the following form:

s/*search string***/***replacement string***/**

where *search string* represents the character string to be replaced and *replacement string* represents the replacement character string.

For example, if the current line contained the following:

```
Lead is a precious metal.
```

Issuing the following command:

```
s/Lead/Gold/p
```

would cause the line to be changed and displayed on the screen as follows:

```
Gold is a precious metal.
```

Note that a **p** included after the final slash causes the updated version of the line to be displayed on the screen.

Be sure to read the next section, which discusses special characters, before using the substitute command extensively. The special characters provide easier methods for accomplishing some tasks.

Special Characters

Before we can fully explain the use of commands that search the file for a piece of text, we must first discuss a few special characters. These characters carry a special meaning to the line editor program. We do not consider commands typed at the asterisk prompt to be special characters. The following characters can carry a special meaning:

^	the caret
.	the period
$	the dollar sign
[	the left square bracket
]	the right square bracket
*	the asterisk (or star)
/	the slash
\	the backslash

The context in which these characters are used can change their meaning. We will discuss the details of special characters, in the following sections.

THE CARET (∧)

The caret is used to represent the beginning of the line. The caret is interpreted in this fashion only when it is the first character of a search string.

For example, if the current line contained the following:

```
Improved Potato Chips
```

The following command:

```
s/∧/New,/p
```

would cause the line to be changed and displayed on the screen as follows:

```
New, Improved Potato Chips
```

Note that the caret has no special meaning when it is in the replacement string. If the current line is the line shown above, the following command illustrates this fact:

```
s/I/∧/p
```

This command would cause the current line to be changed and displayed as follows:

```
New, ∧mproved Potato Chips
```

Also, the caret has no special meaning if it is included somewhere other than at the start of the search string. If the current line is the one shown above, the following command illustrates this fact:

```
s/w,^/w,I/p
```

This command would cause the current line to be changed and displayed as follows:

```
New, Improved Potato Chips
```

THE DOLLAR SIGN ($)

The dollar sign is used to represent the end of the line when it is used at the end of a search string. Do not confuse this usage of the dollar sign with the dollar sign used the represent the last line of a file.

If the current line contained the following:

```
New, Improved Potato Chips
```

The following command:

```
s/$/with Ridges/p
```

would cause the line to be changed and displayed on the screen as follows:

```
New, Improved Potato Chips with Ridges
```

As with the caret, the occurrence of a dollar sign in a replacement string or in some position of the search string other than last will have no special meaning.

THE PERIOD (.)

A period is used to represent any single character in a search string. By any single character, we mean that the character in the position corresponding to the period can have any value and a match will still be found if the other characters match.

The utility of the period used as a special character is best demonstrated with a search for multiple items.

THE ASTERISK (*)

The asterisk is used to represent as many consecutive reoccurrences of the preceding character as possible. The asterisk can be used to locate and eliminate unnecessary repetitions of a character.

For example, suppose the following line is the current line:

```
Then, we                    started to
```

The following command:

```
S/we *started/we started/p
```

would cause the line to be changed and displayed on the screen as follows:

```
Then, we started to
```

THE SLASH (/)

The slash is used as the delimiter for search and character strings. The first slash signifies the beginning of the search string. The second slash signifies the end of the search string and the beginning of the replacement string. The third slash signals the end of the replacement string.

THE BACKSLASH (\ )

The backslash is used to defeat the special meaning of a special character. The backslash is used before a special character if the ordinary character is needed.

For example, suppose the current line contains the following:

```
The / and * are special characters.
```

The following command:

```
s/ \/and \*/slash and asterisk/p
```

would cause the current line to be changed and displayed on the screen as follows:

```
The slash and asterisk are special characters.
```

Substitutions on Multiple Lines

If the same substitution must be made in several lines, the global form of the substitute command is used. The global substitute has the following form:

*first line,last line***s/***search string***/***replacement string***/g**

where *first line, last line, search string,* and *replacement string* retain the same meaning as in the regular substitute command.

Recall the example where we joined several lines into one line (see page 182). The resulting line's words all ran together. To insert the correct spaces, we could have used the global substitute command before joining the lines together. We need to add a space at the beginning of each of the lines that are to be joined to the first. Suppose that we are starting with the original file:

```
one
two
three
```

To insert the necessary spaces, the following command is used:

```
2,$s/∧/ /g
```

The result of joining all of the lines in the file would now be:

```
one two three
```

Searching a File

Files can be searched for the occurrence of a particular character key. The **ed** utility provides several utilities for searching. The search can be conducted on a seek only basis, seek and edit basis or interactive seek and edit.

FORWARD SEARCH

A forward search scans forward in the file until the first match of the search string is found. The search will automatically wrap around from the end of the file to the beginning. The search will then be carried to the starting point, which is the current line. If the search is unsuccessful, the following message is displayed on the screen:

```
?
search string not found
```

If the search is successful, the current line is changed to the line number that contains the match and that line is displayed on the screen. The command to accomplish a forward search has the following form:

/search string/

where *search string* is the character string being sought.

BACKWARD SEARCH

The backward search functions identically to the forward search except that the file is scanned in reverse. The wraparound, if it is necessary, will be from the beginning of the file to the end of the file. The command to initiate a backward search has the following form:

?search string?

where *search string* represents the character string being sought. On small files, it makes little difference whether the search is conducted forward or backward. On large files, search time can be reduced if it is known where in relation to the current line the line being sought is located.

SEARCH AND EDIT

If it is known that each occurrence of a character string must be edited in an identical fashion, then the search and edit command can be used. The search and edit command searches a specified group of lines. Each occurrence of the search string causes the current line to be set to the line containing the match. Then, the editing commands in the command list are executed. The search and edit command has the following form:

```
first line,last lineg/search string/command \
command \
command
```

where *first line* represents the line number the search will start at, *last line* represents the line number the search will end at, *search string* represents the character string being sought and each occurrence of *command* represents an editing command that is to be carried out. These commands comprise the command list. Note that the backslash is used to indicate that there is another command in the list.

Suppose a text file had been labeled with "NEEDS INFO" each time an unavailable piece of data was referenced by the text. When that piece of data became available, it could be inserted into the text using the following search and edit command:

```
g/NEEDS INFO/s/NEEDS INFO/HEADER/ \
a \
text \
text \
.
```

This command would cause each occurrence of "NEEDS INFO" to be changed to "HEADER." It would also insert the two lines of text after the header. Note that when the lines to be searched are not specified, the search and edit command acts upon the entire file.

REVERSE SEARCH AND EDIT

The reverse search and edit command is identical to the search and edit command except that lines do not contain a match to the search string are acted upon. The reverse search and edit command has the following form:

*first line,last line*v/*search string*/*command* \
command \
command

where *first line, last line, search string,* and *command* have the same meaning as in the search and edit command.

INTERACTIVE SEARCH AND EDIT

The interactive search and edit command marks each occurrence of the search string that is found in the specified group of lines. Then, the first marked line is displayed on the screen. The value of the current line is set to that line. One editing command can be executed on that line. Pressing the Return key causes a null command to be executed. After a command is executed, the next marked line is displayed. After the last marked line has been dealt with, the asterisk prompt returns to the screen.

The interactive search and edit command has the following form:

*first line,last line*G/*search string*/

where *first line* represents the line number at which the search is to start, *last line* represents the line number at which the search is to end, and *search string* represents the character string being sought. Note that if no line numbers are specified, the interactive search and edit command will act upon the entire file.

REVERSE INTERACTIVE SEARCH AND EDIT

The reverse interactive search and edit command is identical to the interactive search and edit command except the lines that do not contain a match to the search string are acted upon. The reverse interactive search and edit has the following form:

*first line,last line*V/*search string*/

where *first line, last line,* and *search string* have the same meaning as in the interactive search and edit command.

Bringing Files into the Editing Buffer

Files from the XENIX file system can be brought into the editing buffer. These files can either be added to the file presently in the buffer or replace the file currently in the buffer.

READ

The read command is used to add a file to the file currently in the editing buffer. The read command has the following form:

*line*r *file name*

where *line* represents the line number after which the new file should be placed and *file name* represents the XENIX file that is to be added to the editing buffer. If *line* is omitted, the new file will be placed immediately after the file currently in the editing buffer.

For example, the following command:

15r table

would cause the file **table** to be inserted into the file presently in the buffer. The file, **table,** would be inserted after line 15 but before line 16.

EDIT

The edit command is used to start editing a new file. If changes made on the current file in the editing buffer have not been saved, the following warning will appear on the screen:

```
?
warning: expecting 'w'
```

A second edit command will cause the new file to be moved into the editing buffer without saving the present contents of the buffer.

The edit command has the following form:

e *file name*

where *file name* represents the name of the XENIX file to be updated. For example, the following command:

```
e memo
```

would cause the file **memo** to replace the current contents of the editing buffer.

Undoing Editing Errors

The undo command allows the last change made on the editing buffer to be undone. Specifically, the undo command reverses the effect of the most recently executed command from the following list:

- append (**a**)
- change (**c**)
- delete (**d**)
- search and edit (**g**)
- reverse search and edit (**v**)
- interactive search and edit (**G**)
- reverse interactive search and edit (**V**)
- insert (**i**)
- join (**j**)
- move (**m**)
- read (**r**)
- substitute (**s**)
- transfer (**t**)
- undo (**u**)

The undo command has the following form:

```
u
```

By checking the result of each editing command and undoing mistakes, much time can be saved. The results from commands that affect many lines, such as the search and edit commands, should be checked with particular care.

Running Shell Commands from ed

Shell commands can be run without exiting from **ed**. This facility can be useful to find and view files to add to the editing buffer. The command to run a shell command from the **ed** utility has the following form:

```
!shell command
```

where *shell command* represents the XENIX shell command to run.

For example, the following command:

```
!l
```

would cause the contents of the current working directory to be displayed. And, the following command:

```
!cat table
```

would cause the contents of the file **table** to be displayed on the screen.

Editing Scripts

Editing scripts are used to accomplish a repetitious editing task on a group of files. The technique is to create a file that contains the necessary editing commands. This file is then stored under a name such as **script.** Now each file is edited using the commands stored in the **script** file. The command to accomplish this task has the following form:

```
ed file name < script
```

where *file name* is the name of the file to be edited. Such a command would have to be issued for each file that needed changing. For a practical example on the use of an editing script, see page 77 in chapter three. A normal user will not be able to run the example because he does not have the proper access permissions to the files being edited. However, the example demonstrates the technique of editing several files with an editing script.

Summary

Table 10.1 summarizes the **ed** commands. Note that we did not discuss a few of the listed commands. These commands were included in the table for the sake of completeness. The commands that we did not cover will be of interest only for special applications. For more information on these commands, refer to the system documentation.

Table 10.1. Summary of **ed** commands

Command Name	**Command Format**	**Description**	**Page**
append	**a** text .	Appends text after current line.	172-173
change	line, line **c** text .	Changes specified lines to the given text.	175-177
delete	line, line **d**	Deletes the specified lines.	174-175
edit	**e** file name	Edit a different file.	194
file	**f** file name	Changes the currently specified file.	168
search and edit	**g**/*search string*/ *command list*	Executes commands starting at the lines that contain the search string.	191
interactive search and edit	**G**/*search string*/	Finds each occurrence of search string and allows manual editing on each.	192
help	**h**	Provides explanation of errors.	--
insert	**i** text .	Inserts text before the current line.	173-174
join	line, line **j**	Joins specified lines.	182-183

Table 10.1. (cont.) Summary of **ed** commands.

Command Name	Command Format	Description	Page
mark	line **k** character	Marks specified line with specified character.	177-178
list	line,line**l**	Lists specified lines; includes unprintable characters.	--
move	line,line**m**line	Moves specified line to new location.	178-181
number	line,line**n**	Displays specified lines, preceded by their line number.	--
print	line,line**p**	Display specified lines.	171-172
prompt	**P**	Causes asterisk prompt to be displayed/not displayed.	--
quit	**q**	Causes exit from ed. Checks if write has been done.	167
	Q	Causes exit from ed. Does not check if write has been done. Same as a two **q** commands.	--
read	line **r** file name	Reads specified file into the editing buffer after the specified line number.	193
substitute	line,line**s** /search string/ replacement string/	Searches specified lines for the first occurrence of the search string and replaces it with the replacement string.	183-184
global substitute	line,lines /search string/ replacement string/ **g**	Searches specified lines for every occurrence of the search string and replaces it with the replacement string.	188-189
transfer	line,line**t**line	Transfers a copy of the specified lines to the specified location. Leaves the original location unaltered.	181-182
undo	**u**	Undoes the effect of the last command that modified the editing buffer (i.e., a, c, d, g, i, j, m, r, s, t, v, G, or V).	195
reverse search and edit	**v**/search string/ command list	Executes the command string starting at lines that do not contain the search string.	192

Table 10.1. (cont.) Summary of **ed** commands.

Command Name	**Command Format**	**Description**	**Page**
reverse interactive search and edit	V/search string/	Finds each line that does not contain the search string and allows manual editing on each.	193
use **crypt** with ed	X	Uses the **crypt** utility when reading and writing files.	--
display line	=	Displays the number of last line.	170
number	. =	Displays line number of the current line.	169
shell command	! command	Runs a shell command from **ed.**	196

vi - A SCREEN EDITOR

Any text file can be created and edited using the **vi** utility. The **vi** utility is a full screen editor. A full screen editor allows the user to view and modify text displayed on the entire screen of the terminal. Generally, one screen contains approximately 25 lines. If the entire file will not fit on the screen, only a portion of the file is displayed. The other portions of the file can be viewed with the use of the scrolling commands.

The screen editor allows the user to position the cursor at any point on the screen. Screen oriented editing commands generally affect the file at the point where the cursor is positioned. In addition to the screen oriented commands, **vi** also provides line oriented commands. These commands generally affect the current line of the file. The current line is the line that presently contains the cursor.

In this section, we will discuss both the screen and line oriented commands. In addition, we cover how to enter and exit the **vi** utility and how to move the cursor.

Entering vi

The **vi** utility can be invoked in several different manners. How the utility is invoked depends upon the task at hand.

SIMPLEST METHOD

The easiest method of invoking the screen editor utility is to simply type **vi** and press the Return key. When the **vi** utility is invoked in this fashion, the screen editor is opened but no particular file is specified. New text can now be created and modified. When the editor is exited, a file to write to must be specified in order to save the text created during the editing session.

The screen will appear as follows:

```
[ ]
~
~
~
~
~
~
~
```

Note that each tilde represents a potential line. A potential line will not appear in the file. A potential line becomes an actual line when text or a carriage return is added to it. We represented only a few of the potential lines. In reality, the potential lines would fill the screen. The pair of square brackets represent the position of the cursor.

WITH A FILE SPECIFIED

A more useful method of entering the **vi** utility is to open the editor and specify the file to edit at the same time. The command to accomplish this task has the following form:

vi *file name*

where *file name* represents the name of the file to be edited. If that file already exists, it will be loaded into the editing buffer and the text, beginning at the first line, will be displayed on the screen.

The concept of the editing buffer is important to understand. The editor does not act directly on the specified file. Instead, a copy of the file is placed in the editing buffer. All changes made during the editing session are made on the copy of the specified file. The file itself is not changed until a write command is issued. A write command causes the contents of the editing buffer to overwrite the contents of the specified file.

For example, to enter the screen editor with the file **tryit** specified, the following command would be used:

```
vi tryit
```

As a result of this command, the file **tryit** would be loaded into the editing buffer and the **vi** utility would be entered. The screen display might resemble the following:

```
[T]his is a practice file.
Presently it only has a few lines.
~
~
~
~
~
~

"tryit" 2 lines, 60 characters
```

Note that we only represented a few of the potential lines, which are marked by the tildes. The present position of the cursor is on the first letter in the file, which is a "T". The position of the cursor is indicated by the square brackets that enclose the "T". The last line of the display is the status line. The status line provides information about the specified file. We will see later that the status line is also used to display line oriented and other special commands.

One important case to consider is when a nonexistent file is specified as the file to edit. Suppose the file **empty** does not exist. The following command:

```
vi empty
```

would result in the following screen display:

```
[ ]
~
~
~
~
"empty" No such file or directory
```

The file **empty** does not presently exist in the user's search path. The file will be created when it is written from the **vi** utility. Note that the file will not be created if no text is added to the editing buffer before the write command is issued. Specifying a file that does not exist is a convenient method of creating a new file.

STARTING AT A SPECIFIED LINE WITHIN THE FILE

The **vi** utility may also be invoked so that the cursor is placed at a character other than the first character in the specified file. There are three basic variations in this method of entering **vi.**

The most useful is the command to place the cursor at the beginning of the last line of the file. This command has the following form:

vi + *file name*

where *file name* represents the name of the specified file.

Another method of invoking the **vi** utility allows a specific line number to be indicated. The cursor will be placed at the beginning of the specified line number. This command has the following form:

vi +*line number* *file name*

where *line number* represents the number of the line in which the cursor is to be placed, and *file name* represents the name of the file to be edited.

Finally, the occurrence of a specific string can be searched for. The cursor will be placed at the start of the specified string. This command has the following form:

```
vi +/search string   file name
```

where *search string* represents the specified string and *file name* represents the name of the file to be edited. Note that if the search string is not found, the cursor will be placed at the beginning of the last line in the file.

RECOVERING AN UNSAVED FILE AFTER A SYSTEM CRASH

If the system crashes while a file is being edited, the changes made during the editing session are not necessarily lost. The **vi** utility can be invoked in a special fashion in order to attempt to recover the file. This command has the following form:

```
vi -r file name
```

where *file name* represents the name of the file that was being edited at the time of the system crash.

Even with the above safety feature, it is a good practice to periodically write the file being edited. This action provides insurance from inadvertent destruction of the file in the editing buffer.

Modes

The **vi** utility has different modes of operation. The mode determines how input from the keyboard is interpreted. The **vi** utility has three different modes.

COMMAND MODE

In the command mode, any input from the keyboard is interpreted as an editing command. If the computer's speaker sounds when the Escape key is depressed, then the command mode is active. The command mode can be made active by holding the Escape key in the depressed position until the speaker sounds. While in the command mode, depressing a key that is not defined as part of an editing command will cause the speaker to sound. Note that the **vi** utility is placed in the command mode upon being invoked. Also, the commands are never displayed on the screen. All of these commands are very short. Their input causes an immediate action.

INSERT MODE

In the insert mode, any input from the keyboard is placed into the editing buffer. The insert mode is used to actually create new text within the file. The insert mode can be activated by any of the following **vi** editing commands: insert, append, open, substitute, change, or replace. The insert mode is exited by pressing the Escape key.

ESCAPE MODE

In the escape mode, input from the keyboard is interpreted as a special command. These special commands allow file manipulation and the use of line oriented editing commands. Generally, these commands start with either a colon (:) or a slash (/) and are terminated by a carriage return. The text of these commands will appear on the status line. Recall that the status line is the bottom line of the screen display.

Exiting vi

The changes made to a file are usually saved before the **vi** utility is exited. However, it is possible to quit the editor without saving the contents of the file. The **vi** utility can be temporarily exited to execute shell commands. After the necessary shell commands have been completed, control is returned to the **vi** utility. Also, we include a discussion of how to write a file without quitting the editor. All of the commands to exit the screen editor must be issued from the command mode of the **vi** program.

EXITING AND SAVING

The command most often used to quit the screen editor causes the contents the editing buffer to overwrite the specified file before exiting the **vi** utility. The specified file is the file that was named when **vi** was invoked unless a different file is specified in the quit command.

The command to write to the currently specified file and quit the screen editor has the following two forms:

```
ZZ
or
:x
```

In the case of the **:x** command, **vi** will change to the escape mode. The colon and the "**x**" will be displayed on the status line. In both cases, the status line will display the name of the file being written to, the number of lines in the file and the number of characters in the file. After this information is displayed, the shell prompt will appear on the screen, indicating that **vi** is closed.

The command to write to a file other than the currently specified file and exit the screen editor has the following form:

:x *file name*

where *file name* is the name of the file to be written to. This command is useful if the contents of the original file should not be overwritten. For example, typing the following from the command mode of **vi:**

```
:x newfile
```

and pressing the Return key would cause the current contents of the editing buffer to be stored in the file **newfile.** Note that the original file would not be altered. Also, if **newfile** did not previously exist, it would be created in the current working directory.

SAVING WITHOUT EXITING

In some cases it may be desirable to write to a file without exiting the **vi** utility. Either the currently specified file or a different file can be written to. One reason to write to a file without exiting **vi** is to save all the previous editing changes during a long editing session as a precautionary measure.

The command to write to a file without quitting the screen editor has the following form:

:w *file name*

where *file name* is the name of the file to be written to. If *file name* is omitted, the currently specified file will be written to. Note that specifying *file name* affects only this particular write command. The currently specified file is not changed. This means that if the file **original** is the currently specified file, the following command:

:w newfile

would cause the file **newfile** to be created. But, if the following command was then issued:

:x

the file **original** would be overwritten with the contents of the editing buffer.

EXITING WITHOUT SAVING

If for some reason it is decided that the contents of the editing buffer are not useful, **vi** can be exited without saving the buffer. Once such an action is taken, the contents of that buffer are irretrievably lost.

The command to quit the screen editor without saving the contents of the editing buffer has the following form:

```
:q!
```

Use this command with care. Be sure the contents of the editing buffer are unnecessary before issuing the command.

EDITING ANOTHER FILE

If the editing tasks on one file are completed but another file also must be edited, the next file can be called into the editing buffer. The current contents of the editing buffer should first be saved using a **:w** command.

The command to replace the current contents of the editing buffer with another file has the following form:

:e *file name*

where *file name* represents the name of the file to be placed in the buffer. If the current contents of the editing buffer has not been saved, the following message will appear on the status line:

```
No write since last change (:edit!overrides)
```

As the message states, the following command will allow another file to be placed in the editing buffer without saving the current contents of the buffer:

```
:e! file name
```

where *file name* is the name of the file to be placed in the editing buffer.

EXECUTING A SHELL COMMAND

It is possible to execute a shell command without leaving the **vi** utility. The command to accomplish this task has the following form:

```
:!shell command
```

where *shell command* represents any valid XENIX shell command. For example, typing

```
:!cat results
```

and pressing the Return key would cause the **results** file to be displayed on the screen. At the end of the display, the following message will be displayed:

```
[Hit return to continue]
```

Pressing the Return key will cause the screen to be redrawn. The new display will exhibit the same information that appeared on the screen before the shell command was executed.

Scrolling

Scrolling is used to access part of the file that is not presently displayed on the screen. The **vi** editor allows for both forward and backward scrolling by increments of either a full screen or a half screen. A scroll always accesses contiguous lines in the file. A forward scroll, sometimes called "scrolling down", always moves the display toward the end of the file. A backward scroll, sometimes called "scrolling up", always moves the display toward the beginning of the file. The scroll commands only function in the command mode.

FORWARD

The control-d combination causes a scroll of one-half screen forward in the file. The control-f combination causes a scroll of a full screen forward in the file.

BACKWARD

The control-u combination causes a scroll of one-half screen backward in the file. The control-b combination causes a scroll of a full screen backward in the file.

Moving the Cursor

The cursor can be moved about within the file. If a command is issued that causes the cursor to be moved to a point that is not currently displayed on the screen, the screen display will be redrawn so that the cursor is included in the display.

Generally, if a number is input immediately before a command that causes a cursor movement, the cursor will be moved that number of spaces, lines, words, etc. Examples included with the discussion of the commands will help clarify this point. All cursor movement commands must be input while **vi** is in the command mode.

BY THE SPACE

The cursor can be moved by single spaces both to the right and the left. The commands that cause the cursor to be moved by single spaces will not move the cursor from its present line. If the command indicates that the cursor should be moved past the beginning or the end of the line, the cursor will be moved to the beginning or end of the line, respectively. The cursor will not be moved beyond the current line, nor will it wrap around to the next line.

Pressing the **l** key or the space bar causes the cursor to be moved one space to the right. To move six spaces to the right, type **6l.**

Pressing the **h** key or the backspace key (labelled **←** on an IBM PC XT) causes the cursor to be moved one space to the left. To move nine spaces to the left, type **9h.**

TO THE BEGINNING OF THE CURRENT LINE

The cursor can be moved to the beginning of the current line. To accomplish this task, type **0.**

TO THE END OF THE CURRENT LINE

The cursor can be moved to the end of the current line by typing **$.**

TO A SPECIFIC COLUMN

The | command allows the cursor to be moved to any column in the current line. A column is a one character space in a line. Generally, terminal screens include 80 columns. To move to the fortieth column, type **40|**. The | command will not cause the cursor to move beyond the beginning or the end of the current line.

BY THE WORD

The cursor can either be moved forward (to the right) to the beginning of the next word or backward, to the beginning of the previous word. If the cursor is presently in the middle of a word, a command to move the cursor backward by one word will cause the cursor to be repositioned at the beginning of that word. For example, suppose the cursor were presently located at the indicated position:

wo[r]d

A command to move one word backward would cause the cursor to be respositioned so that it rested on the "w".

When counting words, punctuation can either be included or excluded from the word count. Which forms of the cursor movement by the word commands are used depends on personal preference and circumstances.

To move forward either the w or W keys are used. Pressing the w key causes the cursor to be moved one word to the right. Puctuation marks are counted as words. Pressing the W key causes the cursor to be moved one word to the right. But, punctuation marks are not counted as words. Unlike moving the cursor by the single space, the command to move the cursor by the word will cause the cursor to wrap around to the next line. Finally, a command such as **7W** will cause the cursor to be moved forward to the beginning of the seventh word. The cursor will be moved forward as many lines as necessary to find the seventh word.

The b key and the B key are used to move backward by the word in the file. The b command is the counterpart to the w command while the B command is the counterpart to the W command.

The cursor can also be moved to the end of the next word forward in the file. The e and E keys are used in a fashion identical to the w and W keys, respectively. The cursor is placed at the end of the word instead of the beginning.

BY THE LINE

The cursor can be moved by the line both backward and forward in the file. The cursor can either be placed at the beginning of the line it was moved to or in the same column as it occupied in the line it was moved from.

The + and Return keys cause the cursor to be moved forward one line in the file and placed at the beginning of the line. The j and control-n keys cause the cursor to be moved forward one line while maintaining the cursor's column position.

To move the cursor several lines at once, type the desired number immediately before typing the command key. For example, typing **6+** will cause the cursor to be repositioned at the beginning of the sixth line forward in the file.

The k and control-p keys cause an identical action to the j and control-n keys except the cursor is moved backward in the file. The - key causes an identical action to the + and Return keys except the cursor is moved backward.

Inserting Text

Text can be inserted at any position in the file. The **vi** utility has several insertion commands. Each command starts inserting text at a slightly different point in the file. The actual point of insertion is determined in relation to the position of the cursor.

All of the insertion commands must be given while **vi** is in the command mode. All of these commands cause the mode to be changed to the insert mode. All characters input via the keyboard are placed into the editing buffer and displayed on the screen. These characters are placed at the position indicated by the cursor. Pressing the Return key will cause a new line to be accessed. The insert mode is exited by pressing the escape key after all necessary insertions have been completed. When the insert mode is exited, **vi** returns to the command mode.

BEFORE THE CURSOR

The **i** command is used to start insertion of text before the current position of the cursor. To initiate an **i** command, simply press the i key while **vi** is in the command mode. Recall that **vi** can be forced to the command mode by holding the Escape key down until the terminal speaker sounds.

The **i** command is quite often used to start inputting text into an empty editing buffer. As much text as is necessary can be added using the **i** or any of the other insert commands.

For example, suppose the cursor was positioned on a line as shown below:

cur[s]or

where [s] indicates the cursor is on the 's'. Issuing an **i** command, typing a 1 and a 2, and pressing the Escape key would cause the line to be changed as follows:

cur1[2]sor

BEFORE THE FIRST CHARACTER OF THE CURRENT LINE

The **I** command is used to start insertion of text before the first character in the current line. Recall that the current line is the line in which the cursor presently appears. To initiate an **I** command, press the I key while **vi** is in the command mode.

For example, suppose the cursor was positioned on a line as shown below:

cur[s]or

Issuing an **I** command, typing a 1 and a 2, and pressing the Escape key would cause the line to be changed as follows:

1[2]cursor

AFTER THE CURSOR

The **a** command is used to start insertion of text after the cursor. To initate an **a** command, press the a key while **vi** is in the command mode.

For example, suppose the cursor was positioned on a line as shown below:

cur[s]or

Issuing an **a** command, typing a 1 and a 2, and pressing the Escape key would cause the line to be changed as follows:

cursl[2]or

AFTER THE LAST CHARACTER OF THE CURRENT LINE

The **A** command is used to start insertion of text after the last character of the current line. To initate an **A** command, press the A key while **vi** is in the command mode.

For example, suppose the cursor was positioned on a line as shown below:

cur[s]or

Issuing an **A** command, typing a 1 and a 2, and pressing the Escape key would cause the line to be changed as follows:

cursorl[2]

AT THE BEGINNING OF A NEW LINE BELOW THE CURRENT LINE

The **o** command is used to start insertion of text at the beginning of a new line directly below the current line. To initiate an **o** command, press the o key while **vi** is in the command mode.

For example, suppose the cursor was positioned as shown below:

```
Line above the current line.
cur[s]or
Line below the current line.
```

Issuing an **o** command, typing a 1 and a 2, and pressing the Escape key would cause the display to be changed as follows:

```
Line above the current line.
cursor
1[2]
Line below the current line.
```

AT THE BEGINNING OF A NEW LINE ABOVE THE CURRENT LINE

The **O** command is used to start insertion of text at the beginning of a new line directly above the current line. To initiate an **O** command, press the O key while **vi** is in the command mode.

For example, suppose the cursor was positioned as shown below:

```
Line above the current line.
cur[s]or
Line below the current line.
```

Issuing an **O** command, typing a 1 and a 2, and pressing the Escape key would cause the display to be changed as follows:

```
Line above the current line.
1[2]
cursor
Line below the current line.
```

Deleting Text

Text can be deleted at any position in the file. The **vi** utility has several deletion commands. Each command deletes a different amount of text or starts deleting the text from a different point in the file. What part of the editing buffer is actually deleted is determined by the position of the cursor.

All of the deletion commands must be executed while **vi** is in the command mode. All of the commands cause the specified text to be removed from the editing buffer. The mode is never changed from the command mode.

DELETING CHARACTERS FORWARD

The **x** command is used to delete characters in the forward direction (to the right). The **x** command always starts deleting with the character underneath the cursor. More than one character can be deleted by specifying a number immediately before the **x** command. The **x** command has the following form:

number **x**

where *number* represents the number of characters to be deleted.

For example, suppose the cursor was positioned on a line as shown below:

```
cur[s]or
Line below the current line.
```

Issuing the following command:

2x

would cause the line to be changed as follows:

```
cur[r]
Line below the current line.
```

Note that the **x** command will only delete characters on the current line. So, if the following command had been issued in place of **2x**:

34x

The result would have been as follows:

```
cu[r]
Line below the current line.
```

DELETING CHARACTERS BACKWARD

The **X** command is used to delete characters in the backward direction (to the left). The **X** command always starts deleting with the first character to the left of the character the cursor is currently resting upon. More than one character can be deleted by specifying a number immediately before the **X** command. The **X** command has the following form:

*number***X**

where *number* represents the number of characters to be deleted.

For example, suppose the cursor was positioned on a line as shown below:

```
Line above the current line.
cur[s]or
```

Issuing the following command:

```
2X
```

would cause the line to be changed as follows:

```
Line above the current line.
c[s]or
```

Note that the **X** command will only delete characters on the current line. So, if the following command had been issued in place of **2X:**

```
34X
```

The result would have been as follows:

```
Line above the current line.
[s]or
```

DELETING WORDS

The **dw** command is used to delete words. If the cursor is located in the middle of a word, it will delete from the character underneath the cursor to the end of the word. If a number is specified immediately before the **dw** command, then that number of words will be deleted. Punctuation marks are counted as words. The **dw** command has the following form:

*number***dw**

where *number* represents the number of words to be deleted.

For example, suppose the current line appears as follows:

```
The cur[s]or is in the current line.
The line following the current line.
```

The following command:

```
dw
```

would cause the following changes:

```
The cur[i]s in the current line.
The line following the current line.
```

Also, starting from the original configuration, the following command:

```
9dw
```

would cause the following changes:

```
The cur[f]ollowing the current line.
```

Note that the period was counted as a word. Also, a **dw** command can affect lines other than the current line. Finally, the **dw** command causes two lines to be joined when words are deleted from two or more lines by a single command.

DELETING FROM THE BEGINNING OF A LINE

The **d0** command is used to delete from the beginning of the line up to the character before the character beneath the cursor. The **d0** command has the following form:

d0

For example, suppose the current line appears as follows:

The cur[s]or in the current line.

Issuing the following command:

d0

would cause the following changes:

[s]or in the current line.

DELETING TO THE END OF A LINE

The **d$** command is used to delete from the characters beneath the cursor to the end of the current line. The **d$** command has the following form:

```
d$
```

For example, suppose the current line appears as follows:

```
The cur[s]or in the current line.
```

Issuing the following command:

```
d$
```

would cause the following changes:

```
The cu[r]
```

DELETING ENTIRE LINES

The **dd** command is used to delete the entire current line or a group of lines starting with the current line. The **dd** command has the following form:

*number***dd**

where *number* represents the number of lines to be deleted.

For example, suppose part of a file appears as follows:

```
The curren[t] line
two
three
four
five
```

Issuing the following command:

```
dd
```

would cause the file to be changed as follows:

```
[t]wo
three
four
five
```

Also, starting with the original file, issuing the following command:

```
4dd
```

would cause the file to be changed as follows:

```
[f]ive
```

Substituting Text

A substitution accomplishes the action of a deletion of some text and the insertion of other text. Varying amounts of text can be substituted to any point in the editing buffer. The part of the file actually affected by a substitution is determined by the position of the cursor.

All of the substitution commands must be executed while **vi** is in the command mode. Generally, these commands cause **vi** to enter the insert mode. Recall that the insert mode is exited by pressing the Escape key.

REPLACE A SINGLE CHARACTER

The **r** command is used to replace a single character. The character the cursor is resting upon is changed to the character that is input immediately following the **r** command. The **r** command has the following form:

```
r
```

For example, suppose the current line appears as follows:

```
Current line contains an err[e]r.
```

The following command will correct the spelling error:

```
r
```

followed by typing an "o". The current line would appear as follows after the command shown above was executed:

```
Current line contains an err[o]r.
```

Note that this command does not cause the insert mode to be entered.

REPLACING MORE THAN ONE CHARACTER

The **R** command is used to replace more than one character. The replacement starts with the character beneath the cursor and proceeds to the right. The **R** command has the following form:

R

For example, suppose the current line appears as follows:

Current line contains [a]n error.

The following command:

R

followed by typing “no” and pressing the Escape key would cause the current line to appear as follows:

Current line contains n[o] error.

CHANGING WORDS

The **cw** command is used to change one or more words to one or more different words. Punctuation marks are counted as words. The replacement starts at the character beneath the cursor. The **cw** command has the following form:

*number***cw**

where *number* represents the number of words to be deleted.

For example, suppose the current line appears as follows:

```
The current line contains no [e]rrors.
```

Issuing the following command:

```
cw
```

typing "mistakes" and pressing the Escape key would cause the current line to appear as follows:

```
The current line contains no mistake[s].
```

Now, suppose the current line appears as follows:

```
The current line c[o]ntains no mistakes.
```

Issuing the following command:

```
3cw
```

typing "leaned up" and pressing the Return key would result in the current line appearing as:

```
The current line cleaned u[p].
```

CHANGING LINES

The **cc** command is used to change the entire current line or a group of lines starting with the current line. The number of new lines inserted is not required to equal the number of old lines deleted. The **cc** command has the following form:

*number***cc**

where *number* represents the number of old lines to be deleted.

For example, suppose part of a file appears as follows:

```
The curren[t] line
two
three
four
five
```

The following command:

cc

typing "one" and pressing the Escape key would cause the file to appear as:

```
on[e]
two
three
four
five
```

Now, the following command:

```
3cc
```

typing:

```
The current line.
The line after the current line.
The third line.
```

and pressing the Escape key would cause the file to appear as:

```
The current line.
The line after the current line.
The third line[.]
four
five
```

CORRECTING EDITING MISTAKES

The **u** command is used to correct editing blunders. The **u** command undoes the last command that changed the contents of the editing buffer. If an editing command does not have the desired effect, it is often easiest to use the **u** command to undo the command. A different editing approach may then be attempted. The **u** command has the following form:

```
u
```

For example, if after the final example command in the previous section, a **u** command had been issued, the file would have been returned to the following state:

```
one
two
three
four
five
```

Note that only the last editing command was undone. Serious editing mistakes can be avoided by carefully examining the result of each editing command and undoing commands with undesirable effects.

Line Oriented Commands

Line oriented commands are available in **vi.** Line oriented commands are commands such as those found in the **ed** utility. Generally, the commands found in the **ed** utility are available in **vi.** To access an **ed** command from **vi,** first type a colon while **vi** is in the command mode. The colon will appear on the status line. After the colon, type the **ed** command. For more on line oriented commands, see the section on the line editor, presented earlier in this chapter.

Summary

The following tables group the related **vi** commands. All of the commands that we discussed are listed in these tables. We have also included a few commands that we did not discuss. The individual user might find these useful in specific instances. For more information on these commands, consult the system documentation.

Finally, some of the commands of the **vi** program are not included. These commands embrace more advanced or more obscure topics and are beyond the scope of a simple introduction. Many of these commands deal with customizing the **vi** environment. The **vi** environment is changed by specifying options. The **vi** utility has approximately 40 environment options. These options typically deal with parameters such as whether or not error messages should be displayed on the screen and whether or not line numbers are displayed. The **vi** utility works well using the default settings of these options. Also, there are some additional editing commands, but in our opinion the majority of the important basic features of the **vi** utility were covered in this chapter.

Table 10.2. Commands used to enter the **vi** utility.

Command Function	**Command Form**	**Page**
Invoke **vi** with no file to edit specified.	**vi**	200-201
Invoke **vi** with file specified.	**vi** *file name*	201-202
Invoke **vi** with file specified and position cursor at beginning of the last line in the file.	**vi** + *file name*	203
Invoke **vi** with file specified and position cursor at the beginning of the specified line.	**vi** +*line number file name*	203
Invoke **vi** with file specified and position cursor at the beginning of the first occurrence of the search string	**vi** +/*search string file name*	204
Invoke **vi** for an attempt to recover a file being edited when the system crashed.	**vi -r** *file name*	204

Table 10.3. Commands used to save the editing buffer and exit the **vi** utility.

Command Function	Command Form	Page
Write contents of the editing buffer to the currently specified file and exit the screen editor.	**ZZ** or **:x**	206
Write contents of the editing buffer to a file other than the currently specified file and exit **vi.**	**:x** *file name*	206
Write contents of the editing buffer to the specified file. Do not exit **vi.**	**:w** *file name*	207
Append contents of the editing buffer to the end of specified file. Do not exit **vi.**	**:w >>** *file name*	--
Exit **vi** without saving the contents of the editing buffer.	**:q!**	208
Start editing a new file. Check to see if current editing buffer saved.	**:e** *file name*	208
Start editing a new file. Do not check to see if current editing buffer saved.	**:e!** *file name*	209
Run a shell command.	**:!** *shell command*	209
Start a new shell.	**:!sh**	--

Table 10.4. Scroll commands in the **vi** utility.

Command Function	Command Form	Page
Scroll one-half screen forward.	Press control-d	210
Scroll full screen forward.	Press control-f	210
Scroll one-half screen backward.	Press control-u	210
Scroll full screen backward.	Press control-b	210

Table 10.5. Commands that cause cursor movement in the **vi** utility.

Command Function	Command Form	Page
Move cursor forward by the space.	*number* **l** or *number* press space bar	211
Move cursor backward by the space.	*number* **h** or *number* press backspace key	211
Move cursor forward by the word. Punctuation marks count as a word.	*number* **w**	212
Move cursor forward by the word. Punctuation marks do not count as a word.	*number* **W**	212
Move cursor backward by the word. Punctuation marks count as a word.	*number* **b**	212
Move cursor backward by the word. Punctuation marks do not count as a word.	*number* **B**	212
Move cursor forward to the end of word. Punctuation marks count as a word.	*number* **e**	212
Move cursor forward to the end of word. Punctuation marks do not count as a word.	*number* **E**	212
Move cursor to a specific column.	*number* |	211
Move cursor forward by the line to the beginning of the line.	*number* + or *number* Return	212
Move cursor forward by the line. Maintain same column position.	*number* **j** or *number* control-n	212
Move cursor backward by the line to the beginning of the line.	*number* -	213
Move cursor backward by the line. Maintain same column position.	*number* **k** or *number* control-p	213
Move cursor forward by the sentence.	*number* **)**	--
Move cursor backward by the sentence.	*number* **(**	--
Move cursor forward by the paragraph.	*number* {	--
Move cursor backward by the paragraph.	*number* }	--

Table 10.6. Text insertion commands for the **vi** utility.

Command Function	Command Form	Page
Insert text before the cursor.	**i**	213-214
Insert text before the first character of the current line.	**I**	214
Insert text after the cursor.	**a**	214-215
Insert text after the last character of the current line.	**A**	215
Insert text at the beginning of a new line below the current line.	**o**	215-216
Insert text at the beginning of a new line above the current line.	**O**	216

Table 10.7. Text deletion commands for the **vi** utility.

Command Function	Command Form	Page
Deleting characters forward.	*number* **x**	217
Deleting characters backward.	*number* **X**	218-219
Deleting words.	*number* **dw**	219-221
Deleting words from the beginning of a line.	**d0**	221
Deleting from the end of a line.	**d$**	222
Deleting entire lines.	*number* **dd**	222-223

Table 10.8. Text substitution commands for the **vi** utility.

Command Function	Command Form	Page
Replace a single character.	**r** + *character*	224
Replacing more than one character.	**R**	225
Changing words.	*number* **cw**	225-226
Changing lines.	*number* **cc**	227-228
Undoing the last change made in the editing buffer.	**u**	228-229

XENIX CHAPTER 11 — MAIL SERVICE

INTRODUCTION

The **mail** utility allows users to communicate via messages. The **mail** services are based on the concept of physical mail. A user can send copies of a message to any number of other users. Those other users can respond to the message if necessary.

The **mail** utility maintains a list of all messages received by each user. Until a particular message is actually disposed of by the receiving user, it remains in the system.

In this chapter, we will touch upon all facets of the **mail** facility. We will detail how to create and send messages via the **mail** system and read and dispose of incoming messages. We will also discuss special uses of the **mail** utility.

MODES

The **mail** utility has two distinct modes. These are the command mode and the compose mode. The mail utility displays the same prompt (the underline character) in each mode. The user must keep track of which mode is active.

Command Mode

The command mode is used to manipulate received messages. Received messages can be viewed, saved, or deleted. The command mode is also used to set the **mail** utility's options and for miscellaneous tasks such as running shell commands from the **mail** utility.

While in the command mode, input from the keyboard is interpreted as **mail** utility commands. We will cover the individual commands later. A help menu that displays information about these commands is available. Typing **?** and pressing the Return key from the **mail** command mode causes the screen to display the command menu.

Compose Mode

The compose mode is used to prepare messages to be mailed. Generally, input from the keyboard is added to the message being created. Special compose mode escape commands are used to specify message headers and who should receive copies of the message. A help menu that displays information about the escape commands is available. Typing **~?** and pressing the Return key from the **mail** compose mode causes the screen to display the escape command menu.

SENDING MAIL

It is important to distinguish the parts of a message in order to best use the **mail** utility. We discuss the different parts that comprise a message before covering how to actually send a message via the **mail** system.

Anatomy of a Message

All messages sent via the **mail** utility consist of two parts, the header and the body. The header is used to specify information pertinent to the message. The body contains the main text of the message.

THE HEADER

The information contained in the header tells the **mail** utility where to deliver the message. The header also informs the recipient about the message. The header contains the following fields:

- From
- To
- Date
- Subject
- Carbon Copies (Cc)
- Blind Carbon Copies (Bcc)
- Return Receipt To

The **From** field is used to specify the sender of the message. The **mail** utility automatically sets this field to the logon name of the user who sent the message.

The **To** field specifies the recipients of the messsage. The users who are to receive the message must be specified by the sender. More than one person can be specified as a receiver. At least one person must be specified in the **To** field.

The **Date** field is used to display the date and time that the message was received. The **mail** utility automatically sets the date field.

The **Subject** field is used to describe the message. The **Subject** field is optional and must be provided by the sender.

The **Carbon Copies** field is used to specify who receives a copy of the message. The **Cc** field is optional and must be specified by the sender.

The **Blind Carbon Copies** field is also used to specify who receives a copy of the message. However, the **Bcc** field is not displayed in the message header. People receiving a message do not know who has received a blind carbon copy. The **Bcc** field is optional and must be provided by the sender.

The **Return Receipt To** field is used to specify users who will receive an automatic acknowledgement of the message. The return receipt message simply tells who successfully received the message. The action caused by the **Return Receipt To** field is analogous to sending a registered letter. A record showing that the letter was delivered is obtained; however, no reply message from the recipient is automatically returned.

Note that only those fields that are needed must be specified. The only field that is required by the **mail** utility is the **To** field. The other fields may be used at the sender's discretion.

THE BODY

The body of the message conveys the text of the message. The body can be created using the **mail** utility's compose mode. Alternatively, the body can be made up of an existing file. Such a file could be created using one of the text editors. Generally, short, unformatted messages are created using the compose mode. Messages that are longer or require formatting are usually created using a text editor.

Composing a Message

In this section we discuss how to actually compose a message. Our discussion includes the setting of the header fields and inputting the body of the message.

ENTERING THE mail UTILITY

The command to enter the mail utility and start composing a message has the following form:

mail *user name   user name ...*

where *user name* represents the logon name of the user who is to receive the message.

For example, the following command:

```
mail helen tom
```

would cause the mail utility to be entered. The compose mode would be active. The users **helen** and **tom** would be specified to receive the message.

Alternatively, the **mail** utility can be entered by issuing the following command:

```
mail
```

The **m** command is then used to start composing a message. The **m** command has the following form:

m *user names*

where *user names* represents a list of one or more users.

For example, the following command:

```
m helen tom
```

would cause the compose mode to be activated. The users **helen** and **tom** would be specified to receive the message.

SETTING THE TO FIELD

All of the header fields are set using compose mode escape commands. Compose mode escape commands start with a tilde (~) and are terminated by pressing the Return key. The **~ t** command is used to add recipients to the **To** field. The **~ t** command has the following form:

```
~ t user name   user name ...
```

where *user name* represents the logon name of a person who is to receive the message.

For example, typing the following:

```
~ t kim
```

and pressing the Return key would cause the user **kim** to be added to the list of people to receive the message.

SETTING THE SUBJECT FIELD

The **~ s** command is used to set the **Subject** field. The **~ s** command has the following form:

```
~ s text
```

where *text* represents the contents of the **Subject** field. The **~ s** command causes the previous contents of the **Subject** field to be overwritten.

For example, typing the following:

```
~ s New summer hours.
```

and pressing the Return key would cause the subject field to be changed to read "New summer hours."

SETTING THE Cc FIELD

The **~c** command is used to add users to the **Cc** field. The **~c** command has the following form:

```
~ c user name   user name ...
```

where *user name* represents the logon name of the people to receive copies of the message.

For example, typing the following:

```
~ c john laura
```

and pressing the Return key would cause the users **john** and **laura** to be added to the **Cc** field.

SETTING THE Bcc FIELD

The **~b** command is used to add users to the **Bcc** field. The **~b** command has the following form:

```
~ b user name   user name ...
```

where *user name* represents the logon name of the people to receive blind copies of the message.

For example, typing the following:

```
~ b pat scottie
```

and pressing the Return key would cause the users **pat** and **scottie** to be added to the **Bcc** field.

SETTING THE RETURN RECEIPT TO FIELD

The **~R** command is used to add users to the **Return Receipt To** field. The **~R** command has the following form:

```
~R user name   user name ...
```

where *user name* represents the logon name of a person to receive the receipt message. Note, users should usually only specify themselves to receive the return receipt message.

For example, typing the following:

```
~R sully
```

and pressing the Return key would cause the user **sully** to be added to the **Return Receipt To** field.

EDITING ALL OF THE HEADER FIELDS

The **~h** command is used to set or change all of the header fields. The **~h** command has the following form:

```
~h
```

This command causes the current setting of each of the header fields to be displayed, one at a time. The current contents of the displayed field can be added to or changed. Pressing the Return key will cause the next field to be displayed.

For example, if the example commands from the preceding sections to set each individual field had been issued, each field would contain the following:

```
To: helen tom kim
Subject: New summer hours.
Cc: john laura
Bcc: pat scottie
Return - receipt -to: sully
```

Typing the following:

```
~h
```

and pressing the Return key would cause the following line to be displayed on the screen:

```
To: helen tom kim
```

To add more users to the list, simply type their logon names at the tail of the list. To remove someone from the list, simply backspace to the proper name and retype the list. When the Return key is pressed, the screen would display the following:

```
Subject: New summer hours.
```

Once again, the contents of the field can be added to or changed. When the Return key is pressed again, the screen will display:

```
Cc: john laura
```

The Cc field can be added to or changed. When the Return key is pressed, the screen will display:

```
Bcc: pat scottie
```

The Bcc field can be added to or changed. When the Return key is depressed the following line will be displayed on the screen:

```
Return - receipt - to: sully
```

The last time the Return key is depressed, the screen will display:

```
(continue)
```

The editing of the header fields is complete. The compose mode is still active.

CREATING THE BODY OF THE MESSAGE

The body of the message is created by simply typing the message. The Return key is used to access a new line. The control-d combination is used to signal the end of the message. When control-d is pressed, the screen will display the following:

```
(end of message)
```

Also, the message is sent to the specified users and the **mail** utility is exited.

USING A PREVIOUSLY CREATED FILE

The body of the message can be taken from a previously created file. Such a file will generally have been created using one of the XENIX text editors.

The ~**r** command is used to read file into the body of the message. The ~**r** command has the followng form:

~**r** *file name*

where *file name* represents the name of the file to be placed into the body of the message.

For example, the following command:

~ r memo

would cause the contents of the file **memo** to be placed in the body of the message.

Using the ~**r** command allows a complicated message that was created using the **ed** or **vi** utility to be sent via the **mail** utility. Any of the necessary header fields can be specified before the message is sent.

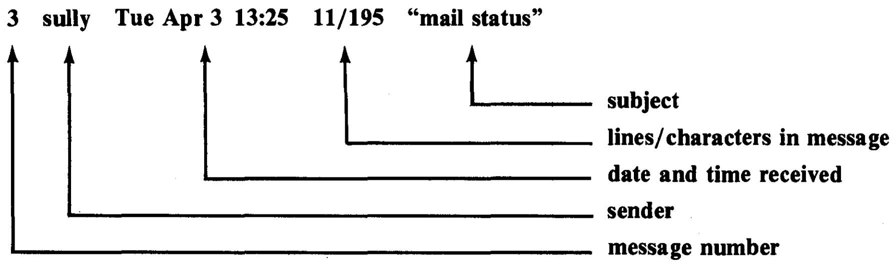
MANIPULATING RECEIVED MAIL

Mail received by the individual user should be read and disposed. Important messages can be saved. Other messages should be deleted. Allowing the mail to stack up is a poor practice because it becomes difficult to find the important messages among the clutter and results in wasted space in the file system.

Received mail is stored in the user's system mailbox. Anyone can read the messages while they are in the system mailbox. A personal mailbox, **mbox,** is available. However, the user must transfer the mail from the system mailbox to **mbox** manually.

Entering the mail Utility

When it is time to read the mail, the **mail** utility is invoked by simply typing **mail** and pressing the Return key. The command mode will be active. The screen will display a header line followed by a line indicating how many messages are currently in the user's system mailbox. Subsequent lines display the message headers which provide information about each message. Each line is similar to the following line:



```
3  sully  Tue Apr 3 13:25  11/195  "mail status"
```

Viewing the Messages

Either the message header, a portion of the message, or the entire message can be displayed. The **h** and **t** commands are useful for finding a particular message. The **p** command is useful for reading an entire message.

MESSAGE NUMBERS

Each message is assigned a number. Generally, the most recently received messages have the highest number. The commands that manipulate the messages, such as viewing, deleting, saving, and responding to mail, all require message numbers as arguments. These message numbers can be referenced in several different manners.

Generally, if no message number is specified, the message that was last acted upon will be affected. We call this default message number the current message. Message numbers can be specified with an actual number or range of numbers. Finally, message numbers can also be specified using + and - parameters. A + message number will cause the current message number to be changed so that it is closer to the first message. A - message number will cause the current message number to be changed so that it is closer to the last message number. Note that the first message number is always number one. The last message number is represented by a dollar sign. The = command causes the number of the current message to be displayed. The = command has the following format:

```
=
```

VIEWING THE MESSAGE HEADERS

The **h** command is used to view only the headers of the received messages. The **h** command has the following form:

h *number or range*

where *number or range* represents a message number or range of message numbers. The headers of the specified messages will be printed.

For example, the following command:

```
h 2
```

would cause the header for the second message to be displayed. Typing only an **h** and pressing the Return key causes all of the headers to be displayed.

DISPLAYING THE FIRST FIVE LINES

Just the first five lines of a file can be viewed. The **t** command allows the top of the message to be read. The **t** command has the following form:

t *number or range*

where *number or range* represents the message number or range of message numbers of the mail to be displayed.

For example, the following command:

```
t 5-$
```

would cause the first five lines of message numbers five through the last message to be displayed.

DISPLAYING THE ENTIRE MESSAGE

The entire message can be displayed using the **p** command. The **p** command has the following form:

p *number*

where *number* represents the message number of the file to be displayed.

For example, the following command:

```
p +3
```

would cause the message with a number three less than the current message to be displayed. If the current message number were five, then message number 2 would be displayed.

Discarding Unwanted Messages

Messages that have been read and are no longer needed should be deleted from the system. Deleting unneeded messages makes the task of handling the remaining mail easier and helps maintain free space within the XENIX file system.

DELETING

The **d** command is used to delete an unneeded message. The **d** command has the following form:

```
d number or range
```

where *number or range* represents a message number or range of message numbers to be deleted. Note that the messages are not renumbered after a delete command. So, there will be missing message numbers in the list after several delete commands. The messages will be renumbered the next time the **mail** utility is entered.

For example, the following command:

```
d + 2 - -1
```

would cause the following four messages to be deleted:

- The current message
- The two messages before the current message
- The message after the current message.

Note that we used a range of messages instead of specific message numbers. The +**2** specifies that two messages before the current message are to be acted upon by the command. The **-1** includes the message after the current message in the action. The first dash (-) is the delimiter in the range parameter. Even though the repetition of the dash in the range parameter may appear strange, it is a perfectly valid expression.

UNDELETING

Messages are not actually removed from the file system until the **mail** utility is quit. A file deleted earlier in the present **mail** session can still be accessed. The **u** command is used to undelete a file. An undeleted file will once again appear in the message list. The **u** command has the following form:

```
u number or range
```

where *number or range* represents a message number or a range of message numbers.

For example, the following command:

```
u 7
```

would cause message number seven to be undeleted. Note that only those messages that are undeleted when a quit from **mail** is executed are removed. Messages that are removed are irretrievable.

Saving Messages

Important messages or messages that are awaiting a reply should be saved. There are several places that can be used to store messages. The two standard sites are the system mailbox and the user's personal mailbox. Alternatively, the message can be saved in any XENIX file.

IN THE SYSTEM MAILBOX

The system mailbox is the site where all new messages are delivered. Mail is picked up at the system mailbox when the **mail** utility is invoked without specifying one or more users to send mail to. Each user has a system mailbox. Any messages in the system mailbox can be read by other users. The system mailbox should be used to store new messages and messages that are pending some action, such as a reply.

All messages that are not specifically deleted or moved to another location are left in the system mailbox when a quit is executed.

IN THE USER'S PERSONAL MAILBOX

The user's personal mailbox is a file in the user's home directory. This file is named **mbox.** Messages stored in the **mbox** are no longer accessible to all other users. The **mbox** should be used to store messages that require long term action or messages that must be saved for future reference.

The **mb** command is used to move messages from the system mailbox to the user's **mbox.** The **mb** command has the following form:

```
mb number or range
```

where *number or range* represents a message number or range of message numbers that are to be moved. Note that the actual move will not take place until the **mail** utility is quit.

For example, the following command:

```
mb 3-4
```

would cause message numbers three and four to be placed in the **mbox** file.

IN A SPECIFIC FILE

Some messages may need to be stored in a specific XENIX file. For example, a message pertaining to a current project should be stored with that project's files. The **s** or **w** commands are used to move messages to a specific file. The two commands are similar in the actions they perform. We will use the **s** command. However, a **w** command could be substituted for any of the **s** commands. Note that the **w** command causes only the body of the message to be saved. The **s** command has the following form:

s *number or range   file name*

where *number or range* represents the message number or range of message numbers that are to be moved and *file name* represents the name of the file to which the messages should be appended. Note that if the specified file does not exist, it will be created. If the specified file does exist, the message will be appended at the beginning of the file.

For example, suppose that we have a subdirectory named **projl** and a file in that directory named **messages.** The following command:

s 1 projl/messages

would cause message number one to be appended to the **messages** file in the **projl** subdirectory.

Responding to Received Mail

Some messages received over the **mail** system require some response. The response may be a reply to the sender or a forwarding of the message to another user. Responding to mail does not cause the message to be deleted from the user's system mailbox.

REPLYING TO THE AUTHOR

The author of a message can be answered using the **r** command. The **subject** field of the message being replied to is placed in the **subject** field of the reply. The compose mode is then activated. The body of the reply is then input in the same fashion as explained in the section on sending mail. The **r** command has the following form:

r *number*

where *number* represents the message number to be replied to.

For example, the following command:

```
r 3
```

would cause the **To** field to be set to the author of message three. The **subject** field would be set to the **subject** field of message three. The reply can now be composed just as it was when sending original mail. Note that the compose mode escape commands may all be used.

REPLYING TO THE AUTHOR AND ALL RECIPIENTS

The author and all recipients of a message can be replied to using the **R** command. The **R** command causes the header fields of the reply to be copied from the header fields of the message being responded to. Otherwise, the **R** command functions like the **r** command. Note that the people listed in the **Bcc** field of the original message do not get copies of the reply.

FORWARDING A MESSAGE

A message can be forwarded to other users. The body of a forwarded message will contain the header fields and the body of the original message. The actual header fields will specify who forwarded the message and who received copies. The **f** command causes the original message to be indented in the forwarded message. The **F** command allows for no indent. Otherwise, both commands function in the same manner. We shall use the **f** command in our examples. The **F** command could replace any of the **f** commands. The **f** command has the following format:

f *number*   *user names*

where *number* represents the number of the message to be forwarded and *user names* represents a list of logon names of the users who are to receive copies of the forwarded message.

For example, the following command:

```
f 2 beth harry
```

would cause message two to be forwarded to the users **beth** and **harry.**

Leaving the mail Utility

The **mail** utility can be left by two methods. Quitting the **mail** utility causes the mail session to be terminated. All changes, such as deletes and moves, are made to the file system. The **q** command is used to quit the **mail** utility. The **q** command has the following form:

```
q
```

Exiting the **mail** utility causes the mail session to be terminated. None of the changes are made to the file system. The **x** command is used to exit the **mail** utility. The **x** command has the following form:

```
x
```

Generally, the **q** command is used to leave the **mail** utility. The **x** command is generally used when a major error was committed and the current **mail** session must be abandoned.

MISCELLANEOUS FEATURES

In this section we will discuss several additional capabilities of the **mail** utility that did not logically fit into the previous discussions.

Reminders Via mail

The **mail** utility allows the user to set up a reminder service. On the specified dates, a message will be sent to remind the user of an important date or event. The XENIX operating system scans the file named **calendar** in each user's home directory. Lines containing either the current or following day's date are sent via the mail service.

To use the reminder service, a user should create a file named **calendar.** The contents of the file might resemble the following:

```
9/7  First deadline Friday.
9/25  Mom's birthday.
1/18  Final deadline.
```

Note that the reminder service requires the **calendar** utility to be run once each day. Consult the system manager to see if this utility is presently included as part of the daily routine.*

Hardcopy of a Message

The **l** command is used to obtain a printed copy of a message. The **l** command has the following form:

l *number or range*

where *number or range* represents a message number or range of message numbers that are to be printed.

Mailing List

A mailing list can be created using the **a** command. The mailing list is referred to as an alias. An alias can be used to refer to several people who are to receive a message, with one name.

* *Note to the system manager: This utility can be run daily by either putting it in the* **/etc/rc** file or by using the **chron** utility.

Current aliases can be displayed by typing **a** and pressing the Return key. The command to create a new alias has the following form:

```
a alias name   user names
```

where *alias name* represents the name of the alias to be created and *user names* represents a list of user logon names who are to be included in the alias.

For example, the following command:

```
a proj1.persnl joe linda dennis anne
```

would cause the alias **proj1.persnl** to be created with members **joe, linda, dennis,** and **anne.** Now, placing **proj1.persnl** in the **To** field of a message will cause all four of the listed people to receive the message.

Using a Text Editor from mail

Both of the editors **ed** and **vi** are available from either the mail command or compose mode. From the command mode, the commands to invoke the editors are the following: **v** for **vi** and **e** for **ed.** From the compose mode the commands to invoke the editors are the following: **~v** for **vi** and **~e** for **ed.**

The dead.letters File

If a message cannot be delivered, it is placed in the **dead.letters** file in the home directory of the sender. Generally, undeliverable messages are caused by an incorrect specification of the receiver. The contents of the message can be salvaged from the **dead.letters** file using the compose mode escape command **~d.** The command has the following form:

```
~d
```

This command causes the contents of the **dead.letters** file to be read into the body of the message currently being composed.

Options

As with the **vi** and **ed** utility, many options are available that slightly modify the manner in which the **mail** utility functions. We feel that the **mail** utility functions quite well without modifying its operation and that a discussion of these options is beyond the scope of an introduction to the **mail** system. Refer to the system documentation for more information on the available options.

SUMMARY

The following tables provide a command summary for the mail system. We have included commands that we did not discuss for the sake of completeness. Refer to the system documentation for more information about these commands.

Command Mode

Table 11.1. Viewing messages

Command Function	**Command Form**	**Page**
View entire next (lower) message.	Press Return key	—
View nth next (lower) message.	+ *number*	—
View nth earlier (higher) message.	— *number*	—
View entire specified message.	**p** *number*	248
View first five lines of specified message.	**t** *number*	247-248
View headers of message only.	**h** *number*	246-247

Table 11.2. Manipulating and answering messages

Command Function	**Command Form**	**Page**
Delete a message.	**d** *number*	249
Undelete a message.	**u** *number*	250
Edit a message (**ed**).	**e** *number*	256
Edit a message (**vi**).	**v** *number*	256
Forward a message to users (indent).	**f** *number user names*	253-254
Forward a message to users (no indent).	**F** *number user names*	253
Hold in mailbox. Used with **autombox** option.	**ho** *number*	—
Print a copy of a message on the system printer.	**l** *number*	255
Move a message to the private mailbox (**mbox**).	**mb** *number*	251
Reply to author of a message.	**r** *number*	252-253
Reply to author and all recipients of a message.	**R** *number*	253
Append a message to a file.	**s** *number file name*	251-252

Table 11.3. Other command mode commands

Command Function	Command Form	Page
Print number of current message.	=	—
Run a XENIX shell command.	! *command*	—
Print global aliases.	**A**	—
Print individual user's aliases.	**a**	256
Set individual user's aliases.	**a** *alias name user names*	256
Change directories.	**c** *directory name*	—
Display command list.	**list**	—
Display command menu help screen.	**?**	236
Send mail to users.	**m** *user names*	239
Quit **mail** utility.	**q**	254
Exit **mail** utility.	**x**	254
Run a XENIX subshell.	**sh**	—
Read commands from a file.	**so** *file name*	—
Search for a string.	**st** *search string range*	—
Display options help screen.	**set?**	—
Set options.	**set** *options*	—
Unset options.	**unset** *options*	—

Compose Mode

Table 11.4. Specifying message header fields

Command Function	Command Form	Page
Set **subject** field.	~ **s** *text*	240
Add users to **To** list.	~ **t** *user names*	240
Add users to **Cc** list.	~ **c** *user names*	241
Add users to **Bcc** list.	~ **b** *user names*	241
Add users to **Return Receipt To** list.	~ **R** *user names*	242
Review and edit all header fields.	~ **h**	242-244

Table 11.5. Reading in and editing body of message.

Command Function	**Command Form**	**Page**
Read in a file.	~ **r** *file name*	245
Read in the **dead.letter** file.	~ **d**	257
Read in a message (indented).	~ **m** *number*	—
Read in a message (not indented).	~ **M** *number*	—
Invoke an editor (**ed**).	~ **e**	256
Invoke an editor (**vi**).	~ **v**	256
Pipe message through a XENIX command.	~ | *command*	—

Table 11.6. Other compose mode escape commands.

Command Function	**Command Form**	**Page**
Execute XENIX shell command.	~ ! *command*	—
Execute **mail** command mode command.	~ : *mail command*	—
Begin a line with a tilde.	~ ~	—
Print global aliases.	~ **A**	—
Print individual user's aliases.	~ **a**	—
Set individual user's aliases.	~ **a** *alias name user names*	—
Print contents of message so far.	~ **p**	—
Abort message.	~ **q**	—
Write message to a file.	~ **w** *file name*	—
Display compose mode escape command help screen.	~ ?	236

XENIX Chapter 12 — Some Additional XENIX Features

INTRODUCTION

In this chapter, we discuss several additional features of the XENIX system that were not discussed in the previous chapters. The main objective of this chapter is to give the reader a knowledge of the power of the XENIX shell and the diversity of the available utilities. We won't attempt to discuss all of the XENIX utilities. Such a discussion would warrant a separate book. We will try to mention the more useful ones. For a complete listing of available utilities, consult the system documentation.

USING THE SYSTEM LINE PRINTER

The system line printer can be used to obtain permanent copies of XENIX files. Since more than one person can request that the printer output a file, the XENIX system must provide facilities to share the line printer among different users.

The XENIX system uses a spooling routine to allocate the printer. Each file to be printed is placed at the end of a queue. That file is printed after the preceding files in the queue have been printed. If there are a large number of files in the printer queue, the waiting time for a file to be printed may be substantial. However, the user is free to undertake other tasks while waiting for the printout.

The **lpr** command is used to submit files to the printer. The **lpr** command has one especially useful option, **-m** or the mail option. With it, the **mail** system is used to deliver a notice when the files have been printed. The **lpr** command has the following form:

lpr -m *file name*

where *file name* represents a list of one or more files that are to be printed. Note that the **-m** is optional.

For example, the following command:

```
1pr -m inventory.list
```

would cause the file **inventory.list** to be printed. The user would be notified by mail when the printing was completed with the following message:

```
Your /usr/lib/lpd job for the file inventory.list is finished.
```

REDIRECTION

In the examples given so far in this book, commands have taken their input from the terminal's keyboard, via the command line, and displayed their output on the terminal's screen. The keyboard is the standard input. The screen is the standard output. The input to a command can be taken from a different source, such as a file. The output from a command can be sent to a different source, such as a file. Such actions are called redirecting the input or output. Redirection is a useful technique. Topics discussed later in this chapter will make use of the technique.

Redirecting Input

The less than symbol (<) is used to indicate redirection of input. This redirection technique has the following form:

command < *file name*

where *command* represents a XENIX command that usually receives input from the keyboard and *file name* represents the name of a file that contains the input for the command.

For example, the following command:

```
mail helen tom < memo
```

would cause the file **memo** to be mailed to both **helen** and **tom.**

Redirecting Output

Output that is redirected to a file can either be appended to the end of the file or used to overwrite the current contents of the file. The file will be created if it did not previously exist.

OVERWRITING

The greater than symbol (>) is used to indicate redirection of output that will overwrite on the existing file. This redirection technique has the following form:

command >*file name*

where *command* represents a XENIX command that usually produces output for display on the terminal's screen, and *file name* represents the name of the file to be overwritten by the command's output.

For example, the following command:

```
cat address body > letter
```

would cause the files **address** and **body** to be written in the file **letter**. The file **address** would appear first.

APPENDING

The greater than symbol (>) is used twice in succession to indicate redirection of output that will be appended to an existing file. This redirection technique has the following form:

command >> *file name*

where *command* represents a XENIX command that usually produces output for display on the screen and *file name* represents the name of the file to be added to by the command's output.

For example, the following command:

```
cat postscript >> letter
```

would cause the file **postscript** to be appended to the end of the file **letter.**

UTILITIES THAT COMPARE AND MANIPULATE FILES

In this section we will discuss some of the XENIX utilities that compare and manipulate files. These commands allow the user to quickly accomplish file handling tasks.

Sorting Files

The **sort** utility is used to sort and merge files. The **sort** utility can sort according to several different methods. The sort command has the following form:

sort *file names*

where *file names* represents a list of one or more files. The sort command orders the file by the value of the ASCII codes of the characters at the beginning of each line. If more than one file is present, the files are merged into one ordered list. The result of the sort command is displayed on the screen.

OPTIONS

The following options are available to modify the method used to accomplish the sort:

b	Causes leading blanks in the comparison field to be ignored.
d	Causes only letters, digits, and blanks to be significant in the sort. This is called "dictionary order."
f	Causes the case to be ignored. That is, all upper case letters are treated as their lowercase counterparts.
n	Causes a numerical sort. This option is intended for use with numerical data.
r	Causes a reverse sort. That is, from "z" to "a" or from highest to lowest.

The **sort** command with options specified has the following form:

sort *-options* *file names*

where *options* represents a list of one or more of the options listed above.

For example, the following command:

```
sort -fr inventory
```

would cause the file **inventory** to be sorted in reverse alphabetical order with case ignored.

SPECIFYING A DIFFERENT SORT KEY

The **sort** utility usually uses the first entry on a line as the sort key. To use a different entry, the position parameters are used. The position parameters are used to mark the beginning and end of the sort key. Each position parameter has the following form:

fields.characters

where *fields* represents the number of fields to skip from the beginning of the line, and *characters* represents a number of characters to skip from the beginning of the field. A field is a string of non-blank characters separated by a blank. The + position parameter defines the start of the sort key. The - position parameter defines the end of the sort key. The **sort** command with position parameters specified has the following form:

sort *-options +position parameter -position parameter file names*

where *options, position parameter* and *file names* have been previously defined.

For example, suppose the inventory file contains two fields. The first field contains the name of the item. Each item name contains no blanks. The second field contains the number of the items that are in stock. The following command:

```
sort -n +1 inventory
```

would cause the inventory file to be sorted by the amount of each item currently in stock. The result would be displayed on the screen. Note that the . *character* designation is not mandatory nor is the - *position parameter.*

OUTPUTTING TO A FILE

The **sort** command can be used to create another file instead of displaying the results of the sort on the screen. This version of the sort command has the following form:

sort -*options* +*position parameter* -*position parameter* **-o***output file file names*

where *output file* represents the name of the file that is to receive the results of the output.

For example, the following command:

```
soft -n +1 -oon.hand inventory
```

would cause the result of the last example to be stored in the file **on.hand** instead of being displayed on the screen.

Comparing Files

XENIX provides several utilities for comparing files. Generally, the comparison involves only two files. The result of comparing two files that are radically different is generally of little use.

COMPARING SIMILAR FILES

The **diff** utility is used to compare similar files and create a script file containing **ed** commands to generate one file from the other. This utility is useful when two files vary only slightly. Only one file need be kept in the file system. The other can be generated from the script file. Note that if this technique is used, the file containing the complete file should not be changed. It should be labelled with a special extension, such as **.ancestor.** All write permissions should be removed from such a file. All working copies of the file would be stored as editing scripts. A working file would need to be regenerated before it could be used. The script file would have to be updated with any changes by overwriting the old contents with another **diff** command. This technique will use execution time on the computer but can save a significant amount of space in the file system.

The **diff** command to create an editing script has the following form:

diff -e .ancestor *file name* *script name*

where *file name* represents the name of the file that is being compressed to an editing script and *script name* represents the name of the file that is to receive the editing script.

For example, the following command:

```
diff -e file.ancestor first > script1
```

would cause the file **script1** to receive the editing script that will create the file **first** from the file **file.ancestor.** Before the editing script can be used, the commands to write the appropriate file must be added to it.

COMPARING THREE FILES

The **diff3** utility allows three files to be compared. The **diff3** command will display the differing ranges in the text. The following indicators are used:

= = = =	**All three files are different.**
= = = **1**	**First file is different.**
= = = **2**	**Second file is different.**
= = = **3**	**Third file is different.**

The **diff3** command has the following form:

diff3 *first second third*

where *first, second,* and *third* represent the names of the files to be compared.

COMPARING LARGE FILES

The **bdiff** utility is used to compare files too large for the **diff** command. The **bdiff** command has the following form:

bdiff *file name1 file name2*

where *file name1* and *file name2* represent the names of the files to be compared. Note that **bdiff** splits the file into smaller segments and invokes **diff** on each segment.

FINDING COMMON OR UNIQUE LINES

The **comm** utility is used to find lines common to two files or unique in one of the files. The two files should be arranged in ASCII sequence before the comparison is made. This arrangement can be accomplished using the **sort** utility with no options or parameters except the files to be sorted.

The **comm** utility produces three columns of output. The first column contains the lines unique to the first file. The second column contains the lines unique to the second file. The third column contains lines common to both files.

The **comm** utility includes three options; 1, 2, and 3. The 1 option suppresses printing of the first column. The 2 and 3 options perform an identical function for the second and third columns. The **comm** command has the following form:

comm -*options first second*

where *first* and *second* represent the names of the files to be compared.

For example, the following command:

```
comm -12 inventory.march inventory.april
```

would cause the lines common to both the files inventory.march and inventory.april to be displayed on the screen. Note that it makes no sense to specify all three of the options.

Searching Files

XENIX provides several utilities to search a file for the occurrence of a pattern. Other search utilities include one that finds consecutive identical lines and one that allows for manual scanning of files too large for the XENIX editors to handle.

SIMPLE SEARCH

The **grep** utility is used to search a file for the occurrence of a particular string of characters. The **grep** command has the following form:

grep *-options search string file names*

where *options* represents a list of one or more of the options detailed below, *search string* represents the string being sought, and *file names* represents a list of one or more files to be searched. Some of the **grep** command's more useful options include:

v	Display all lines except those that match.
c	Display only the total number of matches.
n	Each line that matches is displayed with its line number in the file.
y	Causes case to be not significant in the search.
f *filename*	Causes the search string to be taken from the specified file instead of from the command line.

For example, the following command:

```
grep oil inventory
```

would cause every line that contains "oil" to be displayed on the screen. Note that a simple search pattern with only a few letters specified in the search string can result in more information than was really desired being accessed. In the previous example, entries such as coil and soil would also be found. A less ambiguous example follows:

```
grep  "motor oil"  inventory
```

This command would be more likely to access only the intended entry in the file. Note the use of quotation marks to set off the search string. It is necessary to set off the search string if it contains internal blanks.

SEARCH AND PROCESS

The **awk** utility is used to search a file for the occurrence of a pattern or patterns. A specific action can be associated with each different pattern. The **awk** command has the following form:

```
awk -f program file   file names
```

where *program file* is the name of a file that contains an **awk** program and *file names* is a list of one or more files that are to be searched. An **awk** program contains statements of the following form:

search string { *action* }

where *search string* represents the character string being sought, and *action* represents some action that should be taken as a result of the *search string* being found.

For example, suppose the file **inventory** contains two columns. The first column contains the name of the product. The second contains the number of that product currently on hand. Further, suppose that the file **low** contains the following:

```
$2 + 0 < 11 { print "low on ",$1 }
```

The following command:

```
awk -f low inventory
```

would cause all products with fewer than 11 items in stock to be listed on the screen. Note that the **$1** and **$2** were used to represent the first and second columns, respectively. The zero was added to the **$2** so that the characters in the second column would be treated as numbers.

FINDING IDENTICAL LINES

The **uniq** utility allows identical lines in a file to be found. The **uniq** utility locates only adjacent occurrences of identical lines. If all of the identical lines in a file need to be found, the file should first be ordered, using the **sort** utility. The **uniq** command has the following form:

uniq *-options file name*

where *file name* represents the name of the file to be scanned for identical adjacent lines, and *options* represents one or more of the options listed below.

The **uniq** command options are as follows:

u	Output lines that appear only once in the file.
d	Output lines that appear more than once.
c	Output lines that appear only once and one copy of lines that appear more than once. The **uniq** command will automatically operate in this function if no options are given.

For example, the following command:

```
uniq -d inventory
```

would cause lines that appear more than once in the sorted inventory file to be displayed.

SCANNING A LARGE FILE

A file too large to be handled by a text editor can be scanned manually. The manual scan is generally used to find where the file should be broken when it is to be split into smaller files. We will discuss splitting files in the next section. The **bfs** utility is used to accomplish such a task. The **bfs** command has the following form:

```
bfs file name
```

where *file name* is the name of the file to be scanned. Once the **bfs** command has been issued, the specified file can be viewed using **ed** address commands such as +, -, and line numbers.

For example, the following command:

```
bfs /etc/termcap
```

would cause **/etc/termcap** to be specified. A command such as **1,20p** would cause the first twenty lines of the **/etc/termcap** file to be displayed on the screen.

The **q** command is used to exit the **bfs** utility. Simply type **q** and press the Return key.

Splitting a File

Splitting a large file into two or more smaller files makes the file much easier to manage. The split should usually be made at the logical breaks in the file. For instance, in an alphabetical inventory file, a logical place to split the file would be between words beginning with "m" and "n".

SIMPLE SPLIT

The **split** utility is used to divide a large file into several, arbitrarily sized, smaller files. Each of the new files will be the same size except for the last file. The last file may vary depending on the portion of the original file remaining to be stored in it. The split command has the following form:

```
split -size original resulting
```

where *size* represents the number of lines that the resulting files will contain, *original* represents the name of the original file, and *resulting* represents the name of the newly created files containing the split. Note that each of these files will have a suffix appended to it. The first will have an **aa**, the second an **ab**, etc.

For example, the following command:

```
split -100 /etc/termcap splitcap
```

would cause the **/etc/termcap** file be split into the five files **splitcapaa, splitcapab, splitcapac, splitcapad,** and **splitcapae.** Each of these files would be 100 lines long except for **splitcapae** which would contain 40 lines. Note that the file is divided without regard to the file contents. The user has little control over where the breaks occur. Also, the split utility does not alter the original file.

CONTROLLED SPLIT

The **csplit** utility allows the user to split a file so that the breaks occur in selected spots. The **csplit** utility allows a large file to be split at logical breaks in the file's text. The **csplit** command has the following form:

```
csplit -f resulting   original   line numbers
```

where *original* represents the name of the file to be split, *line numbers* represents a list of one or more line numbers within the file *original* at which the splits will be made, and *resulting* represents the name of the resulting files. Note that each file will have a two digit suffix. The first will be **00,** the second **01,** and so forth.

For example, suppose we had a very long file named **inventory,** that contained our store inventory in alphabetical order. We could split this long file into smaller, more manageable files. The file must first be scanned using the **bfs** utility. The line numbers of those lines that are to be the first line in one of the smaller resulting files should be noted. For the purposes of the example, we will split the file into three smaller files. Suppose the first entry in the **inventory** file beginning with the letter "h" occurs at line 2192. Similarly, the first occurrence of the letter "p" occurs at line 4519. The following command:

```
csplit -f inventory. inventory 2192 4519
```

would cause the inventory file to be split into three smaller files. The name and contents of each resulting file follows:

inventory.00	All entries from the **inventory** file beginning with the letters "a" through "g".
inventory.01	All entries from the **inventory** file beginning with the letters "h" through "o".
inventory.02	All entries from the **inventory** file beginning with the letters "p" through "z".

Note that the **csplit** command does not alter the contents of the original file.

OBTAINING STATISTICS ON FILE SIZE

The **wc** utility is used to gather statistics regarding the size of files. The **wc** utility reports the file size in units of lines, words, and characters. The **wc** command has the following form:

```
wc file names
```

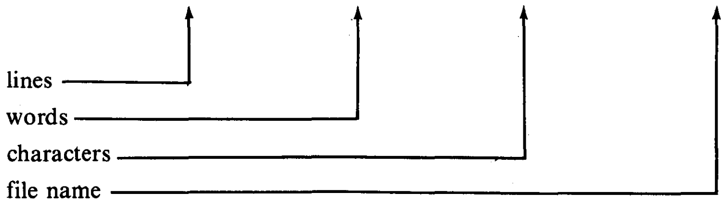
where *file names* represents a list of one or more files that are to be reported upon.

For example, the following command:

```
wc  inventory.00  inventory.01  inventory.02
```

would cause a display similar to the following:

```
2158     4316     47476     inventory.00
2782     5564     61204     inventory.01
2288     4576     50336     inventory.02
7228     14456    159016    total
```



PIPES

Pipes are used to designate the output from one utility as the input to another utility. Pipes can be used to quickly build a facility to accomplish a specific task. The utilities are used as the basic building blocks, and pipes are used to transfer the information between the utilities. Such a structure is often called a pipeline. The individual utilities used in a pipeline are sometimes called filters.

The vertical bar (|) is used to represent a pipeline. On the IBM PC XT the broken bar (¦) is used to represent the vertical bar (|). The pipe symbol is placed after the utility that places information in to it but before the utility that takes information from the pipe. A pipe between two utilities has the following form:

```
producer | consumer
```

where *producer* represents the command that inputs data and *consumer* represents the command that receives data.

For example, a simple use of a pipe follows:

```
who | wc -l
```

This sequence of commands would cause a display similar to the following:

```
2
```

Since the **who** command displays the users currently logged onto the computer, one to a line, and the **wc-l** command counts the lines, in the input, the output tells us that two users are currently logged onto the system.

We will now give a more complicated example. The following sequence:

```
sort inventory | uniq | wc -l > result
```

would cause the number of unique lines in the **inventory** file to be stored in the file **result.**

BACKGROUND TASK

A background task is a task that is executed by the computer at a lower priority than interactive jobs submitted by logged on users. A background task can be started and left to run to completion. The individual who initiated the background task can do other work on the system while the background task is executing.

Background tasks do not usually report on their progress via the display screen. Often, the last task a background task must undertake is to notify the originator by mail that the task is complete. The following list outlines the criteria that makes a task a good candidate for background processing:

1. Completion of the task takes a considerable length of time.
2. All the information to complete the task can be specified beforehand. Files are usually used to store the needed information.
3. The code that accomplishes the task should be thoroughly tested. Errors will be generated only in unusual circumstances.

A task is specified for background processing by the ampersand symbol (&). Background processing can be very effective when doing shell programming. We cover the topic of shell programming later in this chapter. For now, we will give only the following simple example:

```
sort st1.inv st2.inv st3.inv | uniq | wc -1 > result&
```

If the files **st1.inv, st2.inv,** and **st3.inv** were very long, this task could take an appreciable amount of time. The task carried out by this sequence fulfills the other criteria for a background process in that no further information is needed to complete the task and a similar sequence of command was tested in the last example in the section on pipes.

Background processing is a useful tool. However, a background task should only be submitted when necessary. On a busy system, adding many background tasks will cause system performance to become less efficient.

Information about the processes for which an individual is responsible is available through the **ps** command. When a background process is started, the screen will display a number. This number is the process' identification (PID) number. The **ps** command will cause information about all the user's active processes to be displayed. The PID number will be needed to identify the background process. The **ps** command has the following form:

ps

The only sure way to stop a background process is to use the **kill** command. The **kill** command has the following form:

kill *PID*

where *PID* represents a process identification number.

SHELL PROGRAMMING

The XENIX shell can be used as an upper-level programming language. Routines to accomplish specific tasks can be developed and stored in the file system. These files can then be invoked like a utility.

It should be noted that shell programs generally do not use the computer efficiently. However, this drawback is overcome by the ease with which new routines can be developed and modified. A shell program is an excellent approach for accomplishing small tasks and tasks that will be executed only a few times.

Storing a Shell Program

A shell program is stored in a regular text file. Before the program can be executed, this file must be designated as executable. The **chmod** utility can be used to accomplish this task. The command to make the file **my.prog** executable follows:

```
chmod u + x my.prog
```

If the user has several executable files, they should all be stored in the same place. By convention, a user's executable files are generally stored in the subdirectory of the home directory named **bin.** If this convention is followed, be sure that this subdirectory is included in the search path. The search path is defined by the variable **PATH** in the user's **.profile** file. By default, the **.profile** file defines **PATH** as follows:

```
PATH = .:$HOME/bin:/bin:/usr/bin:
```

The appearance of **$HOME/bin:** places the subdirectory **bin** in the search path. Note that this **bin** will be searched after the **.** directory but before the system's **/bin** directory. The search order is important. If the **$HOME/bin** and the **/bin** directories contain files with identical names, the first file that is found is the file that is used.

For example, suppose the user named an executable file in the **$HOME/bin** directory **who.** Then, whenever **who** was used, the search would return the file named **who** in the **$HOME/bin** directory. The file named **who** in the **/bin** directory would be ignored.

Composition of a Shell Program

A shell program can contain any of the following features:

- Any XENIX command
- Redirection of input and output
- Pipes and pipelines
- Control of flow commands

Control of flow commands include the following constructs:

- If, then, else
- Case of
- For loop
- While loop

Using these constructs along with the XENIX utilities allows a user to quickly assemble a program to accomplish complex tasks. The inclusion of the utilities is what makes the concept of shell programming so powerful. These utilities provide facilities to accomplish tasks that would require the writing of separate routines in most upper-level languages.

Creating a Simple Program

Suppose a particular user was often interested in how many users were currently logged onto the system. We already saw that this information can be displayed using the following pipeline:

```
who | wc -l
```

The user could write a simple shell program to accomplish this task. Assuming that the directory **bin** exists in the user's home directory, the following procedure will accomplish the creation of the program **whomany**. The **whomany** program will display the number of users currently using the system.

1. Make the **bin** directory the current working directory. From the home directory, type the following:

   ```
   cd bin
   ```

   and press the Return key.

2. Create the program file, named **whomany,** using a text editor. The file should contain the following:

```
#  This program outputs how many users are on the system
#  The usage is:
#            whomany
who | wc -l
```

   Note that the lines beginning with a pound sign (#) are comments.

3. Use the **chmod** command to make the **whomany** file executable. Type the following:

   ```
   chmod u + x whomany
   ```

   and press the Return key.

The user can now find out how many users are logged in by simply typing:

```
whomany
```

and pressing the Return key.

COMMUNICATIONS AMONG UNIX SYSTEMS

All communications discussed thus far have been between users of the same system. Communications with other UNIX systems is also possible.

Networks

Groups of several UNIX systems are referred to as networks. Networks are most commonly attached via phone lines. So, the machines do not need to be in close physical proximity to participate in a network.

A network will include facilities for sending mail to a user on a different machine and transferring files to other machines.

Networks range in accessibility from private networks, such as interoffice business networks that are not available to the general public, to national, nonrestrictive networks, such as The Source.

UNIX to UNIX Utilities

Communications between UNIX systems that are not part of a network is accomplished using a special set of utilities. For security reasons, access via these utilities to the remote machine is usually severely limited. A friendly user on the remote machine is quite helpful when trying to accomplish specific tasks on that machine.

The **cu** utility is used to contact another UNIX system. The connection is generally made over the telephone lines using a modem. Before the **cu** utility can be used, information about the remote machine must be known. This information includes the telephone number of the remote system as well as the baud rate and type of parity that will be used for the communication.

Commands starting with **uu** are used to accomplish communication tasks. The following commands are the most useful:

• **uucp**	UNIX to UNIX copy. Can be used to send files to the remote machine or retrieve files from the remote machine.
• **uux**	UNIX to UNIX execute. Allows commands to be executed on a remote machine.

In theory, these commands sound quite useful. In practice, most systems will put severe limitations on the commands and files an outside user can access. This action is taken, and rightly so, for security reasons. Many systems only allow outside users to access the **mail** utility and to send files to the public directory, generally named **uucppublic.**

TEXT FORMATTING

The text formatting utilities from UNIX are distributed in the XENIX Text Processing System. The utilities discussed in this section will not be available unless the Text Processing System has been added to the Operating System. Consult the system manager for information on which systems have been installed on the computer.

In this section we will briefly introduce the subject of text formatting. Primarily, we will discuss the use of the **nroff** utility. The **nroff** utility formats text for output devices such as printers.

Text formatting is the process of changing the appearance of the text. The content of the text is not changed when the text is formatted. Formatting is undertaken to increase the eye appeal of the text. A correctly formatted text gives the final copy a more professional appearance.

Input to nroff

Input to the **nroff** utility is generally one or more files. These files contain the text of the document to be formatted. Embedded within the document are formatting requests. These formatting requests are treated as commands by **nroff.** There are well over 60 such formatting commands. We discuss a few of the formatting requests later in this chapter.

Output from nroff

Output from **nroff** is formatted text. The output from **nroff** is sent directly to the display screen unless a different destination is defined.

Invoking nroff

The **nroff** utility should be invoked only after the necessary formatting requests have been inserted into the file. The command to invoke **nroff** has the following form:

```
nroff -options  file names
```

where *options* represent any of the options listed below and *file names* represents one or more files that are to be formatted. Note that options are not required. The dash (-) should be excluded if no options are specified. The following options are the more useful of these available:

o*list*	Only selected pages are output.
s	Stops before every page. Used to allow paper loading. Output resumes when the Return key is pressed.

The output from **nroff** can be redirected to another file using the > symbol. The output can be piped to the printer using the following command:

```
nroff file name | lpr
```

Selected Formatting Requests

In this section, we will discuss the more useful formatting requests. Many of the formatting requests that we do not touch upon relate to type font and size. These parameters are used by the **troff** utility, which generates output formatted for a phototypesetter. Such formatting requests do not work well with **nroff** or are simply ignored.

The general form of a formatting request is a period, followed by two characters. A number can usually be specified to change the scope of the command. Generally, if no number is specified, a one (1) is used. Note that all formatting requests are placed on a line that contains only the formatting request. The period must be in the first column. A period in the first column identifies the line as a formatting request to **nroff.**

Before discussing the formatting requests, it is important to note that the **nroff** utility will format text with no imbedded formatting requests. The formatting is executed so that the text fills a standard 8½ x 11 inch paper. The following is a list of the formatting features that are performed automatically:

- 66 lines per page
- 65 characters per line
- straight left margin
- straight right margin
- single-spaced

PAGE LENGTH

The length of a page of the document can be specified. The page length is always specified as a number of lines. The page length command has the following form:

.pl *number*

where *number* represents the number of lines on the page. If *number* is not present, the page length is set to 66 lines. The page length is usually set at the beginning of the file and only changed for special effects.

For example, the following command:

```
.pl 54
```

would cause the page length to be set to 54 lines.

LINE LENGTH

The length of a line of the document can be set. The line length is always specified as a number of characters. The line length is used to set the left and right margins. The line length command has the following form:

.ll *number*

where *number* represents the number of characters in each line. If *number* is not specified, the line length will be set to 65 characters per line. The line length is usually set at the beginning of the file and changed only for special effects.

For example, the following command:

.ll 50

would cause the line length to be set to 50 characters per line.

LINE SPACING

The **.ls** command is used to specify the line spacing of the text. The **.ls** command has the following form:

.ls *number*

where *number* represents the line spacing to be used. If *number* is not specified, the output will be single-spaced.

For example, the following command:

.ls 3

would cause the output to be triple-spaced. The line spacing is usually set at the beginning of the file and changed only for special effects.

FILLING

The **nroff** utility treats a text file as a very long string. It cuts this long string up to form lines. Spaces are added to lines so that the left and right margins are straight. This process is called filling. Note that the **nroff** utility removes the line breaks from the original file and inserts new ones. If the text must appear as it was in the original file, the fill function can be turned off with the **.nf** command. The **.fi** command causes the filling to be resumed. Defeating the fill function is useful when outputting tables.

CENTERING

Certain lines can be centered exactly as they appear in the original file. The **.ce** command is used to center text. The **.ce** command has the following form:

.ce *number*

where *number* represents how many lines should be centered. If *number* is not specified, only one line will be centered.

For example, the following command:

.ce 2

would cause the next two lines to be centered.

PARAGRAPHS

Paragraphs require special handling because they include several irregular features. The last line of a paragraph must not be filled and must be flush left. The first line of the next paragraph must be indented. Sometimes, extra white space must be added between the paragraphs.

The following commands can accomplish these tasks. The **.br** command is used to cause an unconditional break. This causes the line before the break to be not filled and flush left.

The **.sp** command is used to insert extra white space in to the output. The **.sp** command also causes a break. The **.sp** command has the following form:

.sp *number*

where *number* represents the number of blank lines to be inserted into the output.

The **.ti** command causes the next line to be indented. The **.ti** command has the following form:

.ti *number*

where *number* represents the number of spaces that the line should be indented.

To specify a paragraph with no extra spacing inserted, the following commands could be used:

```
.br
.ti 5
```

However, most text has many paragraphs. It would be much more convenient if we could specify a paragraph with a single command. We can define our own formatting request with the **.de** command. The **.de** command has the following form:

.de *name*
list of formatting commands

where *name* represents a two-letter name. Note that the name must be different from those already in use.

For example, the following command:

```
.de pa
.br
.fi 5
..
```

would cause the **.pa** command to be defined to accomplish paragraph breaks. The **.pa** command is usually set at the beginning of the file.

Besides saving keystrokes when entering the formatting commands, the definition of a **.pa** command to accomplish paragraph breaks allows the text to be restyled with a different form of paragraph very quickly. For example, the text could be reformed into block style paragraphs by redefining the **.pa** command in the following fashion:

```
.de pa
.sp 2
..
```

Example

We now present an example using the simple formatting commands previously discussed. A copy of the original (often called "raw") file follows:

```
.pl 54
.ll 40
.ls 2
.de pa
.br
.ti 5
..
.ce 2
Selected nroff Formatting Requests
Simple Example
.pa
In this example we will show how the
commands that we discussed work.  We
will use all of the commands that we
covered.
.pa
Note that once you get used to using
the text formatter, you can place the
editing commands right in the file as
you go along.  When you are familiar with
this basic set of commands or need
more complex
formatting, consult the system
documentation.
There are many different
commands available.  You
can
accomplish almost any
formatting task using
the XENIX text formatters.
Other formatters include ones for
equations and tables.
```

```
    .pa
    We will now present a short
    summary table of the
    formatting commands covered
    in this discussion.
    .sp 4
    .ls
    .nf
    Table 12.1. Formatting requests.
----------------------------------------------
  Function                 Command

----------------------------------------------
Set page length.            .pl #

----------------------------------------------
Set line length.            .ll #

----------------------------------------------
Set line spacing.           .ls #

---------------------------------------------
Turn off fill function. .nf

---------------------------------------------
Turn on fill function.      .fi

---------------------------------------------
Center lines.               .ce #

----------------------------------------------
Break a line.               .br

----------------------------------------------
Insert white space.         .sp #

----------------------------------------------
Indent the next line.       .ti #

----------------------------------------------
Define a new formatting .de name
request.                    commands
                            ..

----------------------------------------------
.fi
.sp 2
.ls 2
.pa
This is the end of our simple example.
Compare the raw file and the
formatted output.  Note the effect
of the various commands.
```

A copy of the output from nroff, printed by a dot matrix printer, follows:

```
     Selected nroff Formatting Requests

               Simple Example

          In this example we  will  show  how

     the commands that we discussed work.  We

     will use all of  the  commands  that  we

     covered.

          Note that  once  you  get  used  to

     using  the text formatter, you can place

     the editing commands right in  the  file

     as  you go along.  When you are familiar

     with this basic set of commands or  need

     more  complex  formatting,  consult  the

     system  documentation.  There  are  many

     different  commands  available.  You can

     accomplish almost  any  formatting  task

     using  the  XENIX text formatters. Other

     formatters include  ones  for  equations

     and tables.

          We will now present a short summary

     table of the formatting commands covered

     in this discussion.
```

Table 12.1. Formatting requests.

Function	Command
Set page length.	.pl #
Set line length.	.ll #
Set line spacing.	.ls #
Turn off fill function.	.nf
Turn on fill function.	.fi
Center lines.	.ce #
Break a line.	.br
Insert white space.	.sp #
Indent the next line.	.ti #
Define a new formatting request.	.de name commands ..

This is the end of our simple example. Compare the raw file and the formatted output. Note the effect of the various commands.

Appendix A

XENIX Utilities Discussed in the Text

This appendix contains a listing of the XENIX utilities discussed in the text. These utilities are grouped according to their function. For individual commands within the more complicated utilities such as **mail, vi,** and **ed,** refer to the tables in the text.

Privileged Instructions: Utilities Reserved for the Super-User

Utility Name	Function	Page
disable	Disable a special device file.	82
enable	Enable a special device file.	59
/etc/haltsys	Stop the system immediately.	53
/etc/install	Modify the XENIX installation.	46, 48
/etc/shutdown	Stop the system after a delay period.	54
fdisk	Hard disk partition control.	45
hdinit	Initialize the hard disk.	34
mknod	Enable a parallel line printer.	64
mkuser	Create a new user ID.	70-72
pwadmin	Password administration.	98-99
rmuser	Remove a user ID.	88-89
sysadmin	Create a system backup for maintenance purposes.	113
wall	Send a message to all users currently logged onto the system.	107

Utilities Designed to Compare and Manipulate File Contents

Utility Name	Function	Page
awk	Search and process a file.	272
bdiff	Compare two large files.	269
bfs	Scan a large file.	274
comm	Find common or unique lines in two files.	270
csplit	Controlled split of a file.	275-276
diff	Compare two files.	268
diff3	Compare three files.	269
grep	Search contents of a file.	271
sort	Sort and merge one or more files.	265
split	Arbitrary split of a file.	275
uniq	Find common or unique lines in one file.	273
wc	Obtain statistics on the size of a file.	277

Utilities for Handling Files

Utility Name	Function	Page
cat	Concatenate and display files.	153
chgrp	Change the group ownership of a file.	152
chmod	Change the access permission of a file.	150, 152
chown	Change the individual ownership of a file.	150-151
cp	Copy a file to another location.	153-154
find	Locate files with certain characteristics.	107, 110
ln	Create a new link to a file.	155
more	Display a long file. Allows the user to scroll through the file.	153-154
mv	Move a file to another location.	151-152
rm	Remove a file from the system.	148, 156
umask	Set or display the file creation mask.	157-159

Utilities Dealing with Directories

Utility Name	Function	Page
cd	Change directories.	145-146
l	Display contents of a directory.	143-146
mkdir	Create a directory.	147
pwd	Display the name of the current working directory.	146
rmdir	Remove a directory.	148

Miscellaneous Utilities

Utility Name	Function	Page
cu	Call another UNIX system.	284
df	Display free space in file system.	104-105
du	Report of file system usage.	109
ed	Line editor.	166-199
/etc/mkfs	Create a file system.	161-162
/etc/mount*	Mount a file system.	162-163
/etc/umount*	Unmount a file system.	163-164
format	Initialize a floppy diskette.	160-161
lpr	Obtain a copy of a file on the line printer.	261-262
mail	Electronic message service.	235-260
nroff	Text formatter for the line printer.	285-286
quot	Summary of file system ownership.	109-110
tset	Set the terminal type.	63-64
uucp	UNIX-to-UNIX copy.	284
uux	UNIX-to-UNIX execute.	284
vi	Full screen editor.	200-234

* Consult the system manager concerning availability fur for use.

Index

$17.95

XENIX™ User's Handbook

XENIX User's Handbook serves as a clear, concise and practical introduction to the XENIX operating system. This book benefits both the system manager and the general user. **XENIX User's Handbook** assists the system manager from the initial installation to system security and maintenance. The general user is guided from logon procedures to usage of the utilities that make XENIX such a powerful tool.

The topics selected for discussion include the tasks necessary to implement the XENIX system on your computer as well as the more popular and useful utilities. Example commands are included to clarify discussions. The following topics are covered in **XENIX User's Handbook:**

- XENIX system installation
- Adding terminals and printers
- Creating user accounts
- Relationship of additional RAM to system performance
- Printer usage
- XENIX file system
- Text editors **vi** and **ed**
- Electronic **mail** utility
- Text formatting with **nroff**
- File handling and manipulation
- Shell programming
- XENIX system security and maintenance

XENIX User's Handbook serves as a learning guide as well as a reference tool and is an invaluable aid for XENIX users.

ISBN: 0-938862-44-8

84-23705