intel®

XENIX* 286
SYSTEM ADMINISTRATOR'S
GUIDE

*XENIX is a trademark of Microsoft Corporation.

Intel Corporation, 3065 Bowers Avenue, Santa Clara, California 95051

Order Number: 174389-001

XENIX* 286
SYSTEM ADMINISTRATOR'S
GUIDE

Order Number: 174389-001

*XENIX is a trademark of Microsoft Corporation.

Intel Corporation, 3065 Bowers Avenue, Santa Clara, California 95051

The information in this document is subject to change without notice.

Intel Corporation makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Intel Corporation assumes no responsibility for any errors that may appear in this document. Intel Corporation makes no commitment to update or to keep current the information contained in this document.

Intel Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in an Intel product. No other circuit patent licenses are implied.

The following are trademarks of Intel Corporation and its affiliates and may be used only to identify Intel products:

BITBUS
COMMputer
CREDIT
Data Pipeline
Genius
$\overset{\Delta}{i}$
i
I^{2}ICE
ICE
iCS
iDBP
iDIS
iLBX
i$_m$
iMDDX
iMMX
Insite
int$_e$l
int$_e$lBOS
Intelevision
int$_e$ligent Identifier
int$_e$ligent Programming
Intellec
Intellink
iOSP
iPDS
iRMX
iSBC
iSBX
iSDM
iSXM
KEPROM
Library Manager
MCS
Megachassis
MICROMAINFRAME
MULTIBUS
MULTICHANNEL
MULTIMODULE
OpenNET
Plug-A-Bubble
PROMPT
Promware
QUEST
QueX
Ripplemode
RMX/80
RUPI
Seamless
SLD
SYSTEM 2000
UPI

MS-DOS and XENIX are trademarks of Microsoft Corporation.

REV.	REVISION HISTORY	DATE
-001	Original issue	11/84

TABLE OF CONTENTS

CONTENTS

CONTENTS **PAGE**

CONTENTS **PAGE**

CONTENTS **PAGE**

FIGURES

intel®

CHAPTER 1 INTRODUCTION TO SYSTEM ADMINISTRATION

Audience

This book is written for the person responsible for administering the computer system. It covers regular administrative duties, such as adding users, backing up files, and monitoring system use. For installation and configuration information, see the *XENIX 286 Installation and Configuration Guide.* For information about communications networks, see the *XENIX 286 Communications Guide.*

Manual Overview

This guide has these chapters and appendixes:

1. **Introduction to System Administration** -- a discussion of the role of the system administrator, suggestions for keeping records, and checklists.

2. **Starting and Stopping the System** -- instructions for starting the system, logging on, logging off, and stopping the system.

3. **Administering Users** -- instructions for adding and maintaining user accounts, removing a user, maintaining groups, administering passwords, and communicating with users.

4. **Working with Files** -- commands and shell scripts for locating files and directories, and suggestions for examining files.

5. **Working with File Systems** -- a brief discussion of file systems, with instructions for creating a new file system on a flexible disk and for mounting and unmounting a file system.

6. **Providing System Security** -- a presentation of security issues with specific suggestions for protecting your system.

7. **Backing Up** -- strategies and procedures for keeping copies of data.

8. **Monitoring System Use** -- procedures for keeping enough free space, accounting for time used, checking processes, and changing execution priorities.

9. **Solving System Problems** -- instructions for solving problems with terminals, printers, file systems, system crashes, and an inoperable system.

10. **Tailoring the Environment** -- suggestions for tailoring the XENIX system for your installation.

A. **Standard Devices** -- a list of standard devices, with sizes for file systems.

B. **Backup Logs** -- sample backup logs.

C. **The Root Directory** -- a list of files and directories in the root directory.

D. **Related Publications** -- a list of related XENIX publications.

Notational Conventions

As you read this manual, you should be aware of these conventions:

- Real XENIX command and file names are shown in boldface in the text.
- Artificial file names used for illustration are shown in quotation marks in the text.
- Sample dialogues show user entries in boldface and system prompts without boldface.
- When command syntax is given, any category names are shown in italics.
- Many of the commands described in this manual are in the **/etc** directory and path names are given instead of base command names (for example, **/etc/shutdown** instead of **shutdown**). If the **/etc** directory is part of the **root** search path, only the base command name is necessary.

Your Role as System Administrator

As the system administrator, you are responsible for maintaining the computer and its software. You will find that you have several roles--problem-solver, recordkeeper, trainer, monitor, and operator--and that all of these roles are important. If you do your job well, the system will be more efficient, organized, and secure.

First Steps

When you are starting as a system administrator, you should complete these steps:

1. Read the *Overview of the XENIX 286 Operating System* if you are not familiar with XENIX concepts and terminology.

2. Install the XENIX 286 software (if it has not been installed already), following instructions in the *XENIX 286 Installation and Configuration Guide.*

3. Create an ordinary user account for yourself, and go through the *XENIX 286 User's Guide*. Learn at least one text editor (**ed**, **vi**, or **ex**).

4. Read this manual.

5. Plan system backups. (See Chapter 7 and Appendix B.)

6. Tailor the environment (optional). (See Chapter 10.)

7. Add users. (See Chapter 3.)

Routine Tasks

These are tasks that should be performed on a regular basis:

1. Make backups. (See Chapter 7.)

2. Run **fsck** to check the integrity of the file system.

3. Check free space. (See Chapter 8.)

4. Check system security. (See Chapter 6.)

As-Needed Tasks

These are tasks that should be performed as needed:

1. Add and remove users.

2. Solve system problems.

3. Add or remove devices. See the Chapter 10 for information about terminals, and see the *XENIX 286 Installation and Configuration Guide* for information about other devices.

Privileged Functions

When you log on as **root**, you have unlimited power over the system because you automatically override all file protections. Since **root** is under no restraints, you should log on as **root** only when it is necessary to perform functions reserved for **root**. For example, you need to be **root** to edit any files, such as **/etc/passwd** and **/etc/group**, that belong to **root**. You also need to be logged on as **root** to execute the privileged commands listed in Figure 1-1. Most of the privileged files and commands are in the **/etc** directory.

asktime This command prompts for the time of day. Normally it is included in the **/etc/rc** file so you will be prompted for the time when you start the system.

chgrp Only **root** can change the GID (group ID) of someone else's files.

chmod Only **root** can change the permissions on someone else's files, and only **root** can set the sticky bit.

chown Only **root** can change the UID of someone else's files.

chroot Only **root** can change the root directory for a command.

config Only **root** can use **config** successfully because **root** owns the files affected by configuration.

copy Anyone can use this command to copy files, but only **root** can set the set UID and set GID permissions with it.

cpio Only **root** can copy special files.

cron Only **root** can execute commands at specified times with **cron**.

date Anyone can use this command to see the date, but only **root** can change the date.

disable Only **root** can disable terminal ports.

enable Only **root** can enable terminal ports.

fsck Only **root** can check and repair the file system.

haltsys Only **root** can stop the system.

Figure 1-1. Privileged Functions

instl Only **root** can install XENIX.

kill Only **root** can kill someone else's processes.

lpdrestart Only **root** can restart a line printer.

mkfs Only **root** can create a new file system.

mknod Only **root** can build special files for block and character devices.

mkuser Only **root** can add new users.

mount Only **root** can mount a file system.

netutil Only **root** can administer a mail network.

nice Anyone can use **nice** to reduce the priority of a process, but only **root** can use it to increase the priority of a process.

passwd Only **root** can change someone else's password, and **root** does not need to know the old password.

pwadmin Only **root** can administer password aging.

rmuser Only **root** can remove users.

setmnt Only **root** can establish the **/etc/mnttab** table of mounted file systems.

shutdown Only **root** can shut the system down.

su Only **root** can use **su** without a password to gain someone else's user ID, and thus, permissions.

sysadmin Only **root** can back up and restore files.

umount Only **root** can unmount a file system.

uuclean Only **root** can clean the **uucp** spool directory. The **uuclean** command is usually started by **cron.**

uusub Only **root** can monitor the **uucp** network. The **uusub** command is usually started by **cron.**

wall Only **root** can override protections against messages being displayed on the terminal.

Figure 1-1. Privileged Functions (Continued)

Recordkeeping

Keeping accurate records is an important part of your responsibility as a system administrator. Some records can be kept manually and other records can be kept automatically by the system.

Manual logs usually include backup logs (see Appendix B for samples) and a site log for all hardware changes, problems, error messages, and repairs. Many administrators also keep records of all times the system is down.

With XENIX, several important recordkeeping functions are automated and others can be. For example:

- The **dump** program records file system backups and levels in the **/etc/ddate** file. If this file is cleared, the next backup will be a complete backup of the file system.
- The system keeps track of all logins, logouts, and crashes in the **/usr/adm/wtmp** file. You cannot read the file with **cat** or an editor, but you can check its contents by using **who** with **/usr/adm/wtmp** as an argument. If you just want to check current logins, use **who** without any arguments.
- You can keep a log of all processes. (See Chapters 8 and 10.)
- You can keep a log, called **/usr/lib/cronlog**, of processes started automatically by **cron**. (See Chapter 10.)
- You probably should keep a log of all uses of the **su** command, both successful and unsuccessful. Watch for users who try to become **root** since they may be seeking powers they should not have. (See Chapter 10.)

CHAPTER 2
STARTING AND STOPPING THE SYSTEM

Starting the System

The recommended procedure is to start the system, following the steps listed below, then let it continue to operate until hardware servicing is necessary.

1. Turn on the power to the computer and hard disk. When * appears, wait for the system to start automatically or speed up the process by typing U. If you type U, a message appears with a prompt to "Enter <cr>:". Press RETURN and wait while the system is tested and a series of lines is displayed as the system is tested. When you see "SCT SUCCESSFUL ... Now Booting System", the system is ready to run. If a dot appears instead, type "b" and press RETURN.

2. Clean the file system if this appears:

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

   Type the letter "y" (for "yes") and press RETURN to clean the file system. During the cleaning, damaged files are repaired or deleted.

   You will have to reset your computer (by pressing RESET) and start the system again if any of these prompts appear:

   ```
   ***** BOOT XENIX *****
   **Normal System Shutdown**
   REBOOT XENIX (DON'T SYNC!)
   ```

3. Choose the mode of system operation. This prompt gives two choices:

   If you are going to do system maintenance, give the **root** password. This logs you on as **root** in single-user mode, and the **root** prompt (usually #) appears so you can begin to give commands. When you are finished, press CONTROL-D at the **root** prompt if you want to put the system in multiuser mode.

   If you do not need to do system maintenance, press CONTROL-D to go into multiuser mode. Commands in the **/etc/rc** file are executed, and you are prompted for the time. Enter it in the format shown on the screen. For example:

   ```
   Current System Time is Mon Sep 10 18:39:21 PDT 1984
   Enter new time ([yymmdd]hhmm): 8409110820
   Tue Sep 11 08:20 PDT 1984
   login:
   ```

   Other output may appear on the screen, depending on the instructions in **/etc/rc.**

Logging On

Logging on means giving your login name and password so you can go into a login directory. As the system administrator, you should use two different logins: **root** for privileged functions such as adding users, and a regular login for other functions. When you start the system and select the single-user mode of operation, you are logged on automatically. At other times, log on at the login prompt. If it is not displayed, bring it to the screen by pressing CONTROL-D if you have a Bourne shell or by giving the **logout** command if you have a C shell.

At the login prompt, type your login name and press RETURN. When you need to do privileged functions, log on as **root** by giving **root** as the login name and giving the **root** password when prompted for a password. The message of the day is displayed, if there is one, and the shell prompt for **root** (usually #) appears. This is a sample login:

```
login: root
Password:ex45ch
#
```

The password is shown above as an illustration, but it would never really be echoed on the screen. This security measure makes it harder for people to learn other users' passwords.

When you do not need to perform privileged functions, log on as a regular user. After you have logged on, the shell prompt appears on the screen. This is usually $ for the Bourne shell or % for the C shell.

Logging Off

When you want to log off, press CONTROL-D for a Bourne shell or give the **logout** command for a C shell. This takes you to your previous shell if you are not in your login shell and you need to repeat the step. You are logged off when the login prompt appears. Always log off before leaving your terminal. Remember, if you leave your terminal while logged on as **root**, anyone who comes to the terminal can have **root**'s power.

Changing Accounts

When you want to go to a different login account, log off, then log on again with the new login name.

Changing to root ID

If you are logged on as a regular user and need to perform a privileged function, you can use the **su** command to change **root**'s user ID. Just remember that any files you create will have **root**'s user ID and group ID. When you are ready to return to your regular user ID, press CONTROL-D.

In this sample dialogue, the system administrator uses **su** to change to **root**'s user ID (**root** is the default user ID for the **su** command):

```
$ su
password:ex45ch
#
```

Stopping the System

The system performs best if it runs continuously. When you need to bring it down, you must follow the correct procedure. If you do not, you will corrupt your file system and may lose files. **Never just turn the power off.**

Stopping Normally with shutdown

The recommended procedure for stopping the system is to use the **shutdown** command. When you use it, users are warned that the system will go down, the **sync** command is executed automatically so the file system is updated with information in the buffers, file systems are unmounted, and the **haltsys** command is executed automatically to actually stop the system. Once you have given the **shutdown** command, let it run to completion. Do not try to stop it. Follow these steps:

1. Go to the console, log on as **root,** give the **root** password, and stay in the root directory. You must do these things to run **shutdown** successfully.

2. Be sure everyone is logged off before you shut the system down. Use **wall** to warn users, then use **ps -e** to check logins. For information about **wall,** see "Sending a Message with **wall**" in Chapter 3.

3. Invoke the **shutdown** command with the number of minutes to elapse before the shutdown begins (0-15). For example, this allows seven minutes:

   ```
   # /etc/shutdown 7
   ```

   Note: This manual shows commands from the **/etc** directory with the path name. However, if **/etc** is part of your search path, you can use this command:

   ```
   # shutdown 7
   ```

   If you do not give the number of minutes, you are prompted for the information:

   ```
   Minutes till shutdown?(0-15)
   ```

   Enter a number between 0 and 15 or just press RETURN for a default of 5.

   The **shutdown** command kills processes, then unmounts file systems. If you are told a device is busy, it means a file system could not be unmounted, and you may have to clean the file system when you start the system again.

4. You may turn the computer off or press RESET when "**Normal System Shutdown**" appears.

Stopping Immediately with haltsys

The **haltsys** command stops the system without any warning. It is faster than **shutdown** but is a safe alternative only if you are the only person on the system. The procedure is

1. Log on as **root.**

2. Give this command:

   ```
   # /etc/haltsys
   ```

3. You may turn the computer off or press RESET when "**Normal System Shutdown**" appears.

Stopping Manually

In rare instances, the **shutdown** or **haltsys** program may be damaged and you cannot stop the system with them. The possibility is remote, but if it occurs you can follow this procedure to stop the system manually:

1. Log on as **root.**

2. Use **wall** (write to all users) to warn users that the system will be shut down.

3. When all users are off the system, use this command to terminate all multiuser processes and put the system in single-user mode:

   ```
   # kill -1 1
   ```

   This command kills all processes abruptly and should not be used to go into single-user mode unless the system is being stopped manually as described in this section. In other cases, use the **/etc/shutdown su** command to go into single-user mode.

4. Log on as **root** by giving the **root** password when this appears:

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

5. Use the **sync** command twice to force all input and output to be completed:

   ```
   # sync
   # sync
   ```

   After the # prompt returns, you can turn the computer off or press RESET. Since you did not use **shutdown** or **haltsys,** you will be told the system is unclean the next time you start it and you will have to let cleaning proceed.

Going into Single-User Mode

If you are in multiuser mode and want to go into single-user mode, it is not necessary to shut the system down completely and start it again. Instead, you can use the **shutdown** command with the **su** option. You can specify the number of minutes (0-15) that are to elapse before the shutdown occurs or take 5 minutes as the default. The following command brings the system into single-user mode in 10 minutes (or sooner, if everyone except the console logs off before 10 minutes have elapsed):

```
# /etc/shutdown 10 su
```

intel®

CHAPTER 3
ADMINISTERING USERS

Users and Groups

Everyone who has an account is considered a user. This includes both regular users, whom you add to the system as needed, and special system users, who are created automatically as owners of important system files.

Sometimes, several people need to share files, so it is convenient to define them as a group in the **/etc/group** file. The members of the group have group privileges that are controlled by the owner.

Regular Users and Groups

Regular users are the people who work on the system. You create a user account for each user, including yourself, with the **mkuser** command. When a user is added, the user's login name, encrypted password, user ID number (UID), group ID (GID), comment (optional), login directory, and login shell are recorded in the **/etc/passwd** file. The user's group membership is also recorded in the **/etc/group** file. These files and the **mkuser** command are described later in this chapter.

The group in the **/etc/passwd** file is the group the user belongs to at login, but the user can use the **newgrp** command at any time to move to another group he or she belongs to. A user can be a member of any number of groups in the **/etc/group** file but can work in only one group at a time. Any files the user creates belong to the group the owner is in at the time.

System Users and Groups

When the system is installed, the users and groups listed in Figure 3-1 are created automatically. These users own certain system files that are essential to system operation, so do not delete them or change information about them.

System users have user ID numbers under 200 and belong to system groups that have group ID numbers under 50. Notice that **root**'s UID is 0. This is what really gives **root** super-user powers, although the name **root** is also significant and is expected by some programs.

System User	User ID	Group	Group ID	Comment
root	0	**root**	0	The super-user
cron	1	**cron**	1	Daemon for periodic tasks
bin	3	**bin**	3	The owner of system binaries
uucp	4	**uucp**	4	Account for **uucp**
sys	5	**sys**	5	The owner of system source
asg	6	**asg**	6	The owner of assignable devices
sysinfo	10	**uucp**	10	Access to system information
network	12	**network**	12	Micnet account

Figure 3-1. System Users

Adding Accounts

You add a user to the system by using the **mkuser** command, which prompts you for the user's login name, group, password, shell, and a comment (optional). After you have entered this information, you are given a chance to change it, then all of the information except the password goes into the **/etc/passwd** file, along with the encrypted password and the login directory, which are inserted automatically.

As part of **mkuser**, a mailbox is created for the user in the **/usr/spool/mail** directory and important files are created in the user's login directory. For a user with a Bourne shell, a **.profile** file is created. This file has information that is used at login. For a user with a C shell, a **.login** file and a **.cshrc** file are created. The **.login** file is used at login, and the **.cshrc** file is used afterwards. These files placed in the login directory are taken from the **/etc/default/mkuser** directory and can be changed. (See Chapter 10.)

Steps

Follow these steps to add a user account:

1. Set up a terminal, as described in Chapter 10.

2. Log on as **root**.

3. Enable the user's terminal port, if disabled, with the **enable** command. You can use the **ps -e** command to check whether the port is enabled, since it lists enabled terminal ports only. Wait one full minute between each use of **enable** or **disable**, or the system may crash.

4. Use the **mkuser** command to add a user account and choose detailed or brief instructions. The detailed instructions walk you through the steps, explaining each entry, while brief instructions just prompt you for information. See Figure 3-3 and Figure 3-4 for examples.

5. If you haven't set the **TERM** variable in the default **.profile** file (see Chapter 10), use a text editor, such as **ed** or **vi**, to set it. If the user has a Bourne shell, add these lines to the user's **.profile** file:

   ```
   TERM = `tset - -r`
   export TERM
   ```

   Notice that back quotes (not regular quotes) are used. This is essential.

   You could define a particular terminal, such as "TERM=h8020e", but then the wrong terminal characteristics would be used if the user logged on at a different terminal. With this **tset** command, the user can log on at any terminal without changing characteristics because **tset** reads information from the **/etc/ttytype** file. Exporting a variable means sending it to any shell invoked by the login shell.

   If the user has a C shell, the **TERM** variable goes in the user's **.login** file and is defined with this command:

   ```
   setenv TERM `tset - -r`
   ```

   Notice that back quotes (not regular quotes) are used. This is essential.

6. If necessary, create other files for the user. For example, you may want to create a **.mailrc** file in the user's login directory to store instructions affecting electronic mail.

   It is common for a system administrator to add the **.mailrc** file and include instructions to store mail that has been read in a separate mailbox in the user's login directory so only unread mail will be kept in the mailbox in the **/usr/spool/mail** directory. The mailbox for mail that has been read is called **mbox** and is created when needed if the user's **.mailrc** file has this line:

   ```
   set autombox
   ```

   This line is also useful, since it causes mail to be displayed one screen at a time:

   ```
   set page = 22
   ```

   This line is often included so users can end their messages with a dot instead of CONTROL-D when they send mail:

   ```
   set dot
   ```

   The **.mailrc** file is also necessary if the user wants to use routing lists (called aliases) to send mail to specified users. See the *XENIX 286 User's Guide* for details.

Adding a User Account with mkuser

The **mkuser** command simplifies the process of adding users. It prompts for information about the user, gives detailed instructions if you need them, gives you a chance to change entries, updates **/etc/passwd** and **/etc/group**, creates a login (home) directory, creates a mailbox in the **/usr/spool/mail** directory and sends mail to the user, and creates essential files in the login directory (**.profile** for a user with a standard Bourne shell, or **.cshrc** and **.login** for a user with a C shell).

When you use **mkuser**, you may select detailed instructions that describe the information required or brief instructions that just prompt for information. The detailed instructions are self-explanatory, as you can see in Figure 3-3. When you are familiar with the information, you can select brief instructions, as described in Figure 3-4. As you add accounts, remember to press RETURN after each entry you type. At any field that allows "y" (for "yes"), "n" (for "no"), or "q" (for "quit") as a response, you may enter "q" to exit without adding the account.

Before adding an account, get this information:

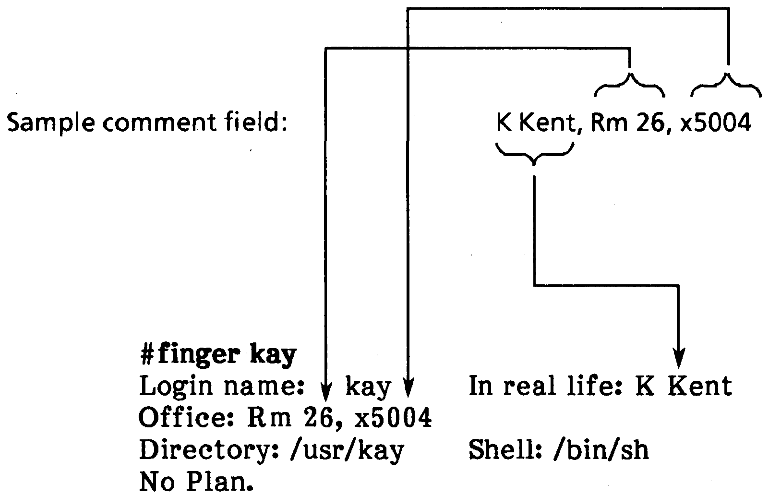
1. Login name. Give each user a unique name for logging on. The login name is usually the user's first name in lowercase letters.

2. Group ID. Assign each user to one login group. During **mkuser** you can select the default group (**group**, with GID 50), select an existing group from the list displayed, or define a new group. If you define a new group, you may assign a group ID or have the system assign one. The group name can be 1-8 characters.

3. Password. Assign each user a password because omitting passwords threatens system security. The minimum size of the password is controlled by the **PASSLENGTH** variable, which you can change. (See "Administering Passwords" later in this chapter.) The password can be more than eight characters, but the system uses only the first eight and produces an encrypted password of 13 characters.

4. Login shell. You can give the user a Bourne shell, a visual shell, or a C shell (if you have the Extended System). If you want to assign a restricted shell that limits the user's access to the system, select the Bourne shell with **mkuser**, then change the login shell in **/etc/passwd** to **/bin/rsh** and see **rsh** in the *XENIX 286 Reference Manual*.

5. Comment. You may want to include information about the user. The **finger** command expects this field to have a user's real name, office, phone extension, and home phone number. It is not necessary to include all of these pieces of information, but if you do, separate them with commas. Figure 3-2 shows how **finger** displays the login name, office, and phone extension from the comment field.



```
#finger kay
Login name:   kay      In real life: K Kent
Office: Rm 26, x5004
Directory: /usr/kay    Shell: /bin/sh
No Plan.
```

Figure 3-2. Use of Comment

```
# mkuser

                              Newuser

                       Add a user to the system

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

              -------- Add a new user to this system --------

This program will guide you through the steps required to add a new XENIX
user to the system.  It will ask you for the following information.  If you don't
know any of these items, press BREAK and come back later.

             login name,  comment,  group,  shell,  password

Generally, the program will tell you what it wants (such as the login name),
describe the acceptable responses, and prompt you for an answer.

Nothing is updated on the system until you confirm everything is correct. The
program will ask you this after you have entered all the information.  At any
time until then you can press BREAK to exit without adding the user.  You can
also answer 'q' or 'quit' to any question to achieve the same effect.

Press RETURN to start:  RETURN

Please give the new user's login name.  This must be at
least 3 characters long, begin with a lower case letter,
and contain only lower case letters and digits.  It must
also not already exist in this system.

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

Figure 3-3. Detailed Instructions for **mkuser**

```
Users are usually assigned to the default user group.
This is group name "group", number 50.  If you want to
assign the new user to some other group, you can either
specify an existing group, or a new group to be created.


Do you want to use the default group? (y/n/q): n
Existing groups are:

        Group "group" (50):
        Group "publications" (51)

Do you want to use one of these groups? (y/n/q): n

Please enter name for new group: sales


Please enter number for new group
Or press ENTER for default number: RETURN


Please give the initial password for this user.


For maximum security a minimum number of characters is required.
Up to eight characters are significant.
You may use any characters except RETURN and LINE FEED.


Note that the password entry will NOT echo on the screen.
This is just to provide an extra degree of security.

If you don't want a password for this user, just type RETURN.

Please enter at least 5 characters for the password
Enter password: xyz12
Re-enter for check: xyz12


Please specify the type of shell (command interpreter)
this user requires.  You can type 1, 2, or 3 as follows:

        1     Standard (Bourne) Shell.
        2     Visual Shell.
        3     C Shell.  (Requires Software Development System)


ENTER Shell type (1, 2, or 3) and press ENTER: 3
```

Figure 3-3. Detailed Instructions for **mkuser** (Continued)

```
There is an optional field in the password file
where you can put a comment, such as the user's
full name, phone #, room #, etc.  If you wish to
leave this blank, just type RETURN.  The comment
should be short (up to 20 characters).


Please Enter Comment:    >------------------------------
                         >J West,Rm 255,x5005


These are the details you have entered for the new user.
Please check they are correct:



User name is "jane", user id is 203.
Group name is "sales", group number is 52.
Comment Field is: J West,Rm 255,x5005
Shell is "/bin/csh"

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


You can change any of the following items:

                        username
                        group
                        password
                        comment
                        shell

Which of these do you want to change? comment


There is an optional field in the password file
where you can put a comment, such as the user's
full name, phone #, room #, etc.  If you wish to
leave this blank, just type RETURN.  The comment
should be short (up to 20 characters).


Please Enter Comment:    >----------------------------
                         >J West,Rm 25,x5005
```

Figure 3-3. Detailed Instructions for **mkuser** (Continued)

```
User name is "jane", user id is 203.
Group name is "sales", group number is 52.
Comment Field is: J West,Rm 25,x5005
Shell is "/bin/csh"

Do you want to change anything? n
Password file updated
Group file updated
Home directory /usr/jane created
/usr/jane/.cshrc created
/usr/jane/.login created
Test mail sent to user: jane
User jane added to this system

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

Figure 3-3. Detailed Instructions for **mkuser** (Continued)

When you are familiar with the information required by **mkuser,** you can select brief instructions. See Figure 3-4 for a sample **mkuser** dialogue with brief instructions.

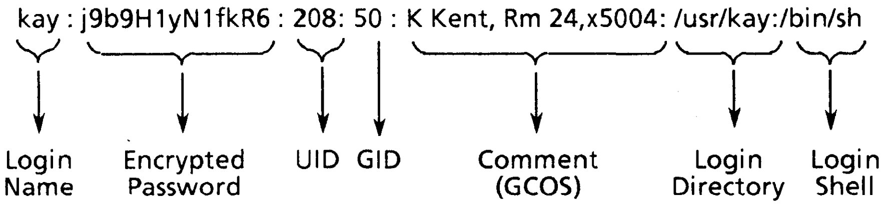
```
# mkuser
                               Newuser

                         Add a user to the system



Do you require detailed instructions?(y/n/q): n
Enter new user's login name: kay
Do you want to use the default group? (y/n/q): y
Please enter at least 5 characters for the password
Enter password: xx=yy
Re-enter for check: xx=yy
ENTER Shell type (1, 2, or 3) and press ENTER: 1
Please Enter Comment:    >----------------------------
                         >K Kent,Rm 24,X5004


User name is "kay", user id is 204.
Group name is "group", group number is 50.
Comment Field is K Kent,Rm 24,x5004
Shell is "/bin/sh"

Do you want to change anything? (y/n/q): n
Password file updated
Group file updated
Home directory /usr/kay created
/usr/kay/.profile created
Test mail sent to user: kay
User kay added to this system


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

Figure 3-4. Brief Instructions for **mkuser**

Maintaining Accounts

After you have added a user account, you may log on as **root** and use a text editor, such as **ed** or **vi,** to change information in **/etc/passwd.** You may change the group ID, comment, and login shell. If you do change the login shell, remember that a user with a Bourne shell needs a **.profile** and a user with a C shell needs a **.login** and a **.cshrc.** The **/usr/lib/mkuser/mkuser.prof** file has the contents of the original **.profile** used by **mkuser,** so you can copy it into the user's login directory and name it **.profile.** Likewise, you can copy **/usr/lib/mkuser/mkuser.login** and name it **.login** and copy **/usr/lib/mkuser/mkuser.cshrc** and name it **.cshrc.**

The **/etc/passwd** file is important, so be very careful if you edit it. For safety, you may want to copy the file, edit the original, check the edited file with **pwcheck,** then remove the copy when you know the edited file is right. Figure 3-5 identifies the fields in **/etc/passwd,** Figure 3-6 has a sample **/etc/passwd** with two regular users ("kay" and "jane") in addition to system users, Figure 3-7 gives an example of changing the comment field in **/etc/passwd,** Figure 3-8 gives an example of copying the file before making the change, and Figure 3-9 has sample error messages from **pwcheck.** If you get a message, examine the displayed line carefully and edit the file to correct the error. If you are told the login directory is not found, the problem is that the directory identified in the line does not exist. If you are told the login directory is null, the problem is that the line itself does not include the login directory.



F-0331

Figure 3-5. Fields in the **/etc/passwd** File

```
# cd /etc
# more passwd
root:lizyO4aERunxs:0:0:The Super User:/:/bin/sh
cron:NOLOGIN:1:1:Daemon for periodic tasks:/:
bin:NOLOGIN:3:3:The owner of system binaries:/:
uucp::4:4:Account for uucp:/usr/spool/uucp:/usr/lib/uucp/uucico
sys:NOLOGIN:5:5:The owner of system source:/:
asg:NOLOGIN:6:6:The owner of assignable devices:/:
sysinfo:NOLOGIN:10:10:Access to system information:/:
network:NOLOGIN:12:12:Mail account:/usr/spool/micnet:
kay:j9b9H1yN1fkR6:208:50:K Kent,Rm 24,x5004:/usr/kay:/bin/sh
jane:j9ABtMuOLAL8I:209:52,J West,Rm 25,x5005:/usr/jane:/bin/csh
```

Figure 3-6. Sample **/etc/passwd** File

```
# cd /etc
# ed passwd
988
*/kay
kay:j9b9H1yN1fkR6:208:50:K  Kent,Rm  24,x5004:/usr/kay:/bin/sh
*s/Rm 24/Rm 26/p
kay:j9b9H1yN1fkR6:208:50:K  Kent,Rm  26,x5004:/usr/kay:/bin/sh
*w
988
*q
# pwcheck
#
```

Figure 3-7. Editing **/etc/passwd** with **ed**

```
# cd /etc
# cp passwd passwdcopy
# ed passwd
988
*/kay
kay:j9b9H1yN1fkR6:208:50:K  Kent,Rm  24,x5004:/usr/kay:/bin/sh
*s/Rm 24/Rm 26/p
kay:j9b9H1yN1fkR6:208:50:K  Kent,Rm  26,x5004:/usr/kay:/bin/sh
*w
988
*q
# pwcheck
# rm passwdcopy
#
```

Figure 3-8. Copying and Editing **/etc/passwd**

```
# pwcheck


learn::11:11:Learn account:/usr/lib/learn:/usr/bin/learn
       Login directory not found
       Optional shell file not found

demo::200:50:Demonstration  account:/usr/demo:/bin/sh
       Login directory not found
```

Figure 3-9. Sample **pwcheck** Messages

Removing a User

When you need to remove a user from the system, save any of the user's important files, remove the user's mailbox, remove the files in the user's home directory, and use **rmuser** to remove the user's login directory (including subdirectories) and entries in **/etc/passwd** and **/etc/group.** You can remove only regular users with **rmuser.** You cannot remove a system user (**root, cron, bin, uucp, sys, asg, sysinfo,** or **network**) or any other user whose user ID is less than 200.

The steps for removing a user are listed below, and sample dialogues are shown in Figure 3-10 and Figure 3-11.

1. Log on as **root.**

2. Use **tar** to save files that the user will need again (after a leave of absence, for example), or transfer files to directories of other users.

3. Remove the user's mailbox in the **/usr/spool/mail** directory.

4. Go to the user's login directory.

5. Remove all files in the user's login directory except . and .. with this command:

   ```
   # rm -fr .
   ```

6. Go to a directory outside the user's directory structure, because the user's login directory cannot be removed if you are in it.

7. Use **rmuser** to remove the user account. This command prompts you for the user's login name (you have to give the name, not the UID) and asks you to confirm that you want to remove the user. The account is removed only if

   - The login name is not a system name and the user ID is 200 or greater.

   - The user's mailbox in **/usr/spool/mail** is empty or gone.

   - The user's login directory has no files other than ., .., **.profile, .login,** or **.cshrc** (and of these, only . and .. have to be present).

   The screen tells you whether the user has been removed and gives the reason if the user is not removed. If you are in the user's login directory when you use **rmuser,** the user will be removed from **/etc/passwd,** but the login directory will not be removed.

8. If no one will be using the terminal, use **disable** to disable it, then turn off the power to the terminal and remove it (optional). For example, this command disables the terminal named **/dev/ttyc1:**

   ```
   # disable /dev/ttyc1
   ```

   Wait one full minute between each use of **disable** or **enable,** or the system may crash. If the terminal is disabled, you may disconnect the terminal line from the back of the terminal or turn off the power to the terminal.

Figure 3-10 illustrates how a user and all of the user's files are removed from the system. Figure 3-11 illustrates how a user's files with "test" as part of the file name are reassigned to a new owner ("sam", who is a member of the same group) and moved to a new directory before the user ("fred") and his remaining files are removed from the system. The -**f** option of **rm** means to remove a file even if it has write protection. The -**r** option means to go through the entire directory subtree.

Note that when the screen prompts you to press ENTER, you may press ENTER or RETURN.

```
# rm /usr/spool/mail/harry
# cd /usr/harry
# rm -fr .
# ls -a
.
..
# cd /etc
# rmuser
                        RMUSER - remove a user from the system


This program allows you to remove users from the system.
It will ask you for the username, and then delete the corresponding
password file entry, the user's mail box, home directory, and
.profile file.

Before removing a user, you should check that the corresponding
mail box is empty, and all the files owned by the user have either
been deleted or transferred to some other user.  The program will
check this for you, and refuse to remove the user if the check
fails.


Press ENTER when you are ready:RETURN


Enter name of id to be removed: harry
Removing user harry from the system.  CONFIRM ? (y/n/q): y
User harry removed from the system
Do you want to remove another user? (y/n): n
#
```

Figure 3-10. Removing a User and All the User's Files

The first line of Figure 3-11 instructs the system to start at the "/usr/fred" directory and find all files in the subtree that include the word "test". The * before and after means that any other characters can precede and follow the word "test". For example, if these files existed, they would be found: "test1", "amtest", "test.4". The **-exec** option means to execute the following command, which is **chown.** The argument is "sam", which means to make "sam" the new owner. The brackets mean to substitute the path name of the file found. The backslash and semicolon (\;) are necessary to end the **-exec** option. The second line instructs the system to start at the "/usr/fred" directory, find all files belonging to "sam", and use the **-exec** option to move them to the "/usr/sam" directory. If by chance files in the two directories had the same name, those originally belonging to "fred" would replace those belonging to "sam".

```
# find /usr/fred -name '*test*' -exec chown sam {} \;
# find /usr/fred -user sam -exec mv {} /usr/sam \;
# rm /usr/spool/mail/fred
# cd /usr/fred
# rm -fr .
# ls -a
.
..
# cd /etc
# rmuser

                      RMUSER - remove a user from the system


This program allows you to remove users from the system.
It will ask you for the username, and then delete the corresponding
password file entry, the user's mail box, home directory, and
.profile file.

Before removing a user, you should check that the corresponding
mail box is empty, and all the files owned by the user have either
been deleted or transferred to some other user. The program will
check this for you, and refuse to remove the user if the check
fails.


Press ENTER when you are ready:RETURN


Enter name of id to be removed: fred
Removing user fred from the system.  CONFIRM ? (y/n): y
User fred removed from the system
Do you want to remove another user? (y/n): n
#
```

Figure 3-11. Moving Files and Removing a User

Maintaining Groups

Information about groups is stored in the **/etc/group** file, which has the fields shown in Figure 3-12. When the system is installed, system groups are created automatically, and when you add a user with **mkuser**, group membership is recorded in this file. The group itself is added if it did not exist before.

The **/etc/group** file is owned by **root** and can be modified with a text editor such as **ed** or **vi**. If you are logged on as **root**, you may add a group, add a member to an existing group, remove a member from a group, or remove a group. The file can have any number of groups, and a user can belong to any number of groups. These pieces of information, separated by colons, are stored in the file:

- Group name. Give each group a different name, using any combination of eight letters and numbers.

- Group password. Although space is allowed for a password for historical reasons, there is no easy way to use one.

- Group ID number (GID). Use a different number between 0 and 65535 for each group. Group ID 50 is reserved as the default for all users, and group IDs less than 50 are reserved for system groups.

- Login names of group members. List all group members by their login names, and separate names with a comma. A group may have any number of members and a user may be listed in any number of groups.

Figures 3-13 and 3-14 illustrate how to edit **/etc/group**. Figure 3-13 shows how to check group names and numbers with **more** (to prevent selecting an existing number), add a new group to **/etc/group** with **ed**, and check the edited file with **grpcheck**. Figure 3-14 shows how to look at **/etc/group** with **more**, copy **/etc/group** for safety, add users to an existing group with **ed**, check **etc/group** with **grpcheck**, and remove the copy of the file. Figure 3-15 illustrates some sample messages from **grpcheck**.

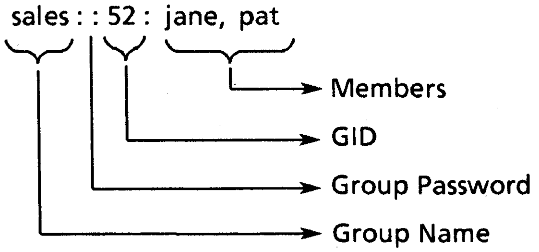


Figure 3-12. The Fields of the **/etc/group** File

```
# cd /etc
# more group
root:x:0:root
cron:x:1:cron
bin:x:3:bin
uucp:x:4:uucp
asg:x:6:asg
sysinfo:x:10:uucp
network:x:12:network
sys:x:5:sys
group::50:root,kay
pubs::51:jack
sales::52:jane
# ed group
219
*a
admin::53:fred,kay
.
*1,$p
root:x:0:root
cron:x:1:cron
bin:x:3:bin
uucp:x:4:uucp
asg:x:6:asg
sysinfo:x:10:uucp
network:x:12:network
sys:x:5:sys
group::50:root,kay
pubs::51:jack
sales::52:jane
admin::53:fred,kay,sam
*w
247
*q
# grpcheck
#
```

Figure 3-13. Adding a Group

```
# cd /etc
# more group
root:x:0:root
cron:x:1:cron
bin:x:3:bin
uucp:x:4:uucp
asg:x:6:asg
sysinfo:x:10:uucp
network:x:12:network
sys:x:5:sys
group::50:root,kay
pubs::51:jack
sales::52:jane
# cp group groupcopy
# ed group
247
*/admin
admin::53:fred,kay
*s/kay/kay,sam/p
admin::53:fred,kay,sam
*w
251
*q
# grpcheck
# rm groupcopy
#
```

Figure 3-14. Adding Users to a Group

```
# grpcheck


admin::53:fred,kay,sam,mike
      mike - Logname not found in password file


sales:52:jane,kay
      Too many/few fields
```

Figure 3-15. Sample **grpcheck** Messages

Administering Passwords

Since much of system security depends on passwords, it is important for you to require passwords and force users to change them often.

Protecting the root Password

Your **root** password gives you super-user powers, including access to all files in the system and exemption from all file protections. Guard this password with care. Give it to others only if you trust them and only when absolutely necessary. Change the password often, and remember it. **If you forget the root password, you have to install XENIX again.**

Insisting on User Passwords

Always assign a new user a password when you use **mkuser,** always assign a password. If you do not, others can easily find the login name in **/etc/passwd** and go into the user's login directory.

Users can change their own passwords at any time with the **passwd** command. Encourage them to select difficult passwords that are at least five characters long and have letters, digits, and punctuation marks.

Enforcing Password Changes

Users often keep the same password until forced to change it, so you, as **root,** should use **pwadmin** to make them change passwords at regular intervals. Users may consider the changes inconvenient, but their data will be safer because of them.

You can force a user to change passwords immediately by using the **-f** option of **pwadmin.** For example, this forces a user whose login name is "kay" to change passwords at the next login:

```
# pwadmin -f kay
```

You can also prevent a user from changing a password by using the **-c** option. For example, this prevents a user whose login name is "jane" from changing her password:

```
# pwadmin -c jane
```

The recommended procedure is to force users to change passwords at regular intervals by setting defaults for password aging, then starting aging for individual users. Set defaults by defining **MINWEEKS** and **MAXWEEKS** in **/etc/default/passwd. MINWEEKS** is the minimum number of weeks that must elapse before a user can change a password and **MAXWEEKS** is the maximum number of weeks that can elapse before a change. Start password aging for individual users by invoking **pwadmin** with the **-min** and **-max** options to define the minimum and maximum number of weeks. This example makes "sam" wait one week before changing his password, but forces him to change it by the end of the second week:

```
# pwadmin -min 1 -max 2 sam
```

Only one login name can be used as an argument with **pwadmin**, so repeat the command for each user to be affected by password aging. The minimum and maximum you set with **pwadmin** apply only until a user's first password change. After that, the minimum and maximum in **/etc/default/passwd** are used automatically.

To check the password aging interval, use **pwadmin -d** option and the user's login name as an argument. For example:

```
# pwadmin -d sam
Minimum weeks: 1.
Maximum weeks: 2.
```

Password aging information for a user is stored as a series of characters following the encrypted password in **/etc/passwd.** For example, before password aging, the entry in the file might be

```
sam:nOecL6Sp7sY9g:210:53:S Dole,Rm 20,x5000:/usr/sam:/bin/sh
```

After password aging, the entry might be

```
sam:nOecL6Sp7sY9g,0/7A:210:53:S Dole,Rm 20,x5000:/usr/sam:/bin/sh
```

The password aging information is encoded in "0/7A" following the comma in the encrypted password field. Since the aging information is encoded, you cannot tell what it means by looking at the **/etc/passwd** file. Instead, you use the **-d** option of the **pwadmin** command, as shown above.

Changing the Minimum Password Length

As **root,** you may use a text editor such as **ed** or **vi** to change the minimum **PASSLENGTH** variable in **/etc/default/passwd.** A minimum of five characters is suggested.

Replacing a Password

A user who forgets a password cannot log on, so you must supply a new password. To do this, log on as **root** and use the **passwd** command with the user's login name as an argument. When you are prompted for the new password, type it and press RETURN. The sample dialogue in Figure 3-16 shows the password, but it would not really be echoed on the screen.

```
# passwd jane
Enter new password (minimum of five characters)
Please use a combination of upper and lowercase letters and numbers.
New password:Qq12j
Re-enter new password:Qq12j
#
```

Figure 3-16. Replacing a Password

Communicating with Users

XENIX gives you four ways to communicate with users via their terminals: edit the message of the day, send a message to all terminals immediately with **wall,** send a message to a particular terminal immediately with **write,** or send mail to specified users' mailboxes with **mail.**

The Message of the Day

The **motd** (message of the day) file in the **/etc** directory is displayed each time a user logs on. This is a regular text file that **root** can edit with a text editor such as **ed** or **vi.** Just remember to use spaces instead of tabs in the file and make the message as short as possible. If a message is long, it will be very slow on some terminals, and if it fills more than one screen, it will scroll so quickly users will not see the beginning lines.

System administrators often use this file to welcome users to the system and remind them when the system is scheduled to be down.

Sending a Message with wall

If you want to send a message to all users immediately, use the the **wall** command. For example, if you wanted to warn people that the system would be down for two hours, you could send a message with **wall:**

```
# wall
The system will be down for two hours. Please log off.
CONTROL-D
```

After you pressed CONTROL-D, this message would appear at each terminal (including yours) immediately:

```
Broadcast Message from root
The system will be down for two hours. Please log off.
```

Sending a Message with write

If you want to send a message to a user's terminal immediately, use **write** with the user's login name as an argument. The user's terminal announces that you are sending a message. The two of you can take turns sending lines back and forth (a line is sent when you press RETURN), but you need some kind of signal to mark the end of the message (some use "o" for "over") and another symbol to mark the end of the conversation (some use "oo" for "over and out"). When the conversation is over, both of you press CONTROL-D to return to the shell.

If the person is not logged on, you are notified and the **write** command ends. If a process on the receiving terminal is not to be interrupted, you are warned that permission to that terminal is denied and the **write** command ends. Permission may be denied because the user has used the **-n** option with the **mesg** command to prevent incoming messages, or because a process such as **nroff** does not allow terminal interruptions.

For example, suppose you are logged on as "sarah" and want to remind Kay to see you before 2:00. You could give this command:

```
$ write kay
```

This would cause her terminal to display this message identifying you and your terminal:

```
Message from sarah ttyc1
```

She could respond:

```
$ write sarah
Hello. What's up?
o
```

You could then send your message:

```
Please see me by 2:00.
o
```

She could end the conversation:

```
Okay.
oo
```

Both press CONTROL-D to return to the shell.

Sending a Message with mail

If it is not necessary for users user to see a message immediately, you should use **mail.** You may send the same message to one or more people, and the arrival of mail is announced at login or at the completion of the command in progress. In this example, mail is sent to mailboxes belonging to Jack, Jane, and Kay:

```
$ mail jack jane kay
Subject: Signup sheet
Please check the new signup sheet.
CONTROL-D
(end of message)
$
```

Use CONTROL-D to end the message, as shown above. Messages that cannot be sent (because of misspelled login names, for example) are saved in the **dead.letter** file, if possible. If the file exists, the message is added to it, otherwise the file is created. It could not be saved, however, if you lacked write permission on your login directory. In this case, you would be warned that the file could not be opened.

If the message is saved, you can redirect the message to the correct login. Before you do this, you may want to edit the **dead.letter** file or copy it and edit it, because otherwise all messages in **dead.letter** will be sent, not just the most recent one. In this example, the **dead.letter** file is redirected, then deleted:

```
$ mail sam < dead.letter
$ rm dead.letter
```

intel®

CHAPTER 4 WORKING WITH FILES

Examining Files

You may use several different commands to check file contents, giving the file name as an argument in each case. The commands are

file Use **file** when you want to check the type of contents in a file. For example, before you use **cat, ed, vi,** or **more,** you should use **file** to be sure the file you are interested in is an ASCII file (has only characters that can be displayed on the screen). If it is not, the nonprintable characters can make the terminal unresponsive. (See Chapter 9 if this happens.)

cat Use **cat** to display the contents of a short file (one that has fewer than 24 lines) on the screen.

ed, vi, ex Use a text editor when you want to change a file.

more Use **more** when you want to see a file one screen at a time. Once the file is on the screen, you can press RETURN to see one more line, type "d" to scroll up one-half screen, press the space bar to see the next screen, or type "q" to return to the shell.

Changing File Ownership

When a file is created, it is owned by its creator. Occasionally it is necessary to transfer ownership to someone else. For example, people may be reassigned to different projects, someone may leave, or you may create a file when logged on as **root** and need to reassign it to your regular account.

If the file should be transferred to another group, use **chgrp** to change the group. File owners can use the command with their own files, and **root** can use it to transfer any file to any group. This example transfers the "list.customer" file to the "admin" group:

```
# chgrp admin list.customer
```

Use the **chown** command to change file ownership. File owners can use the command with their own files, and **root** can use it to assign any file to a new owner. This example makes "sam" the new owner of the "list.customer" file:

```
# chown sam list.customer
```

Using Permissions Effectively

Much of the security of your system depends on careful use of file permissions and many user problems arise from confusion over permissions. As the main overseer of security and provider of assistance, you should have a thorough understanding of permissions.

Access Permissions for Ordinary Files

Access to each file is governed by read, write, and execute permission for the owner, other members of the owner's group, and all others. These permissions are defined as follows:

Read — Reading a file means looking at its contents. Displaying a file on the terminal, printing it, compiling it, and copying it are all examples of reading a file.

Write — Writing a file means changing it in some way. Adding and changing information are examples of writing a file.

Execute — Executing a file means running it as a program. Most executable files are compiled programs, and execute permission is required to run them. Some executable programs are shell scripts (programs using XENIX commands and the shell programming language) and require read permission. For example, someone who has read permission on a shell script called "check" can execute it with this command:

```
$ sh check
```

Someone who has execute permission as well as read permission can execute a shell script by invoking the file name. For example:

```
$ check
```

Set UID/GID — Each user has a user ID number (UID) and a group ID number (GID), and these numbers are checked against permissions whenever someone tries to access a file.

Occasionally a user needs someone else's UID (usually **root**'s). For example, a user needs **root**'s UID to change a password, because only **root** can change the **/etc/passwd** file. XENIX allows for this with set UID permission on the **passwd** command. Set UID permission gives a user the effective UID of the owner while the command is being executed. As soon as the command is completed, the owner's real UID is in effect.

Any executable file, except a shell script, can have set UID or GID permission so anyone who executes the file has the effective ID of the owner or the group.

Access Permissions for Directories

Directories can be read, written to, or searched.

Read

Reading a directory means looking at the contents of the directory file itself. Since a directory contains only a list of file names and their inode numbers, reading it means using the **ls** (list) command to look at the list. Figure 4-1 shows what kind of information is available to someone with read permission on a directory. The **-i** option of the **ls** command shows inode numbers and file names.

```
$ ls -i  /usr/mary
450 letters
460 memos
475 programs
$
```

Figure 4-1. Reading a Directory

Read permission on a directory gives access to the names of files in the directory, not to the files themselves.

Someone who has read permission on a file and knows its name can see its contents provided he or she has search permission on its directory. Read permission on its directory is not necessary.

Write

Writing to a directory means creating a new file (including a subdirectory) in it or deleting a file from it. It may help to picture the actual contents of the directory file. Writing to a directory means adding a name to the list of files or removing a name from the list, and these things can be done only with the appropriate commands. You cannot edit a directory file with a text editor.

Deleting a file requires write permission on its directory, but not on the file. If write permission on the file is missing, the mode is shown and pressing RETURN causes the file to be saved, while typing "y" and RETURN causes it to be deleted.

Changing the contents of a file requires write permission on the file but not on the directory.

Search

Directories have search permission instead of execute permission since it is meaningless to execute a file that is not a program. Searching a directory means going to it with the **cd** (change directory) command or searching through its list of files when a file name is given. Using a file name successfully requires search permission for every directory in the path.

Access Permissions for Special Files

System owners, such as **root, bin,** and **asg,** own all of the special files. Ordinary users generally have write permission for terminals and printers and no permission for other devices. Special files for terminals are usually owned by **root** when they are not being used. A user becomes the owner temporarily at login and can set the access permissions on the terminal. When the user logs off, ownership reverts to **root.**

Read Reading a special file means looking at its contents. For example, if you deny others read permission for your terminal, they cannot read what you are typing. Read permission for a printer is meaningless.

Write Writing a special file means sending data to it. For example, if you give others write permission on your terminal, they can send messages to your screen. People usually give others write permission on their terminals by using the **mesg** command to permit or prevent messages from reaching the terminal. The **mesg y** command permits others to send messages, and the **mesg n** command prevents others from sending messages.

Execute Trying to execute a special file is meaningless.

Representing Permissions

Permissions can be represented in two ways. One way is to show them with characters: r for read permission, w for write permission, x for execute permission, s for set UID or GID permission, and a dash (-) for permission denied. These permissions are shown for the owner, other members of the group, and all others. For example, read, write, and execute permission for the owner, other members of the group, and all others, are represented in Figure 4-2. Permissions are often called the file mode, protection bits, or permission bits. Examples are shown in Figure 4-3.

The other way to represent permissions is with these octal numbers (numbers in a number system with 8 as the base):

4 = read
2 = write
1 = execute
0 = deny permission

You add the numbers for permissions. For example, read, write, and execute permission is represented by 7 (4+2+1), and read and execute permission is represented by 5 (4+1). The owner, others in the group, and all others have separate totals. For example, 777 means full permissions for the owner, others in the group, and all others. If set UID or set GID permission is in effect, one of these digits precedes the series:

4 = set UID permission
2 = set GID permission
6 = set UID and GID permission

For example, 6711 gives UID and GID permission to anyone executing the file, gives full permission to the owner, and denies permission to others in the group and all others.

Figure 4-3 shows how the two different methods represent the same permissions.

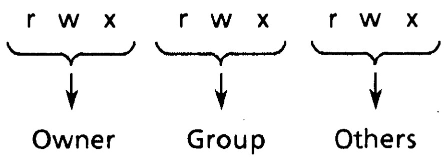


Figure 4-2. Representing Permissions with Characters

Characters	Numbers	Meaning
rwxr-xr-x	755	read, write, and execute permission for the owner read and execute permission for the group read and execute permission for others
rwxr-x---	750	read, write, and execute permission for the owner read and execute permission for the group no permission for others
r-x--x--x	511	read and execute permission for the owner execute permission for the group execute permission for others
rw-rw-rw-	666	read and write permission for the owner read and write permission for the group read and write permission for others
rwsr-x---	4750	read, write, and execute permission for the owner owner's permission for anyone executing the file read and execute permission for the group no permission for others

Figure 4-3. Representing Permissions

Seeing Permissions

The long directory listing (**ls -l**, **l**, or **lc -l**) shows permissions as characters. Preceding the rwx codes is another code that identifies the file type. It is - for an ordinary file, d for a directory, b for a block special file, c for a character special file, p for a named pipe, s for a semaphore, or m for shared data (memory).

Default Permissions

When users create ordinary files in the **/usr** directory, certain permissions are assigned automatically, so they are called default permissions. Initially, the defaults are

```
rwxr-xr-x
```

These permissions give read, write, and execute permission to the owner, read and execute permission to the group, and read and execute permission to all others, and they are created by this command in **/etc/profile:**

```
umask 022
```

The default is created by using **umask** to define the permissions to be removed from a base of 777 (octal for full permission for everyone). When 022 is removed from 777, the result is 755, which removes write permission for the group and others.

It is often desirable to change the defaults. For example, the preferred default may be full permission for the owner, read and execute permission for others in the group, and no permission for others. The octal representation is 750, which requires subtracting 027 from 777. The 750 permissions become the default if **root** edits **/etc/profile** to include this command:

```
umask 027
```

Users can set their own **umask** value in their **profile** files. If they do, it is used to form the default instead of the **umask** value in **/etc/profile.**

After creating files, users can invoke **chmod** (change mode) to change permissions. For example, when Mary creates her "a.jones" letter, it has the default permissions. If she wants to include write permission for others in her group so they can make revisions on the original draft, she can use this command:

```
$ chmod g + w a.jones
```

The "+w" means add write permission, and the "g" means for the group. See the *XENIX 286 User's Guide* and the *XENIX 286 Reference Manual* for more information about **chmod.**

Summary of Permissions for Common Functions

Figure 4-4 illustrates the permissions needed for common functions like copying files. If a user has problems with these functions, it is often because of a misunderstanding of file or directory permissions.

Command	Permissions		Messages/Comments
	Directory	File	
cat	--x	r--	"cat: cannot open *file*" -- no read permission on the file.
cd	--x	---	You can go to a directory only if you have search permission.
copy, cp	-wx	r--	"copy [cp]: cannot open *file*" -- no read permission on the file or search permission on the source directory. "copy: cannot create *file*" -- no write permission on the destination directory. The destination file takes the permissions of the source, minus the current umask. If you want the same permissions, use **umask 0.** For example: **(umask 0; copy *source destination*).** Because of parentheses, only the current command is affected.
ed	-wx	rw-	"cannot create output file" -- no write permission on the directory. "cannot open input file" -- no search permission on the directory or read permission on the file. "cannot open *file*" -- no write permission on the file.
find	r-x	---	"find: cannot open */directory*" -- no read permission on a directory. If you lack search permission on a directory, its files are ignored.
head	--x	r--	If you know a file name, you can read it, even if you lack read permission on the directory. If you lack read permission on the file or search permission on the directory, you get a "Permission denied" message.
l, lc, ls	r-x	---	". unreadable" -- no read permission on the directory. If you lack search permission on the directory and try to use an option that gives a long listing (such as ls -l), a "*file* not found" message is given for each file.
mail		rw-	If you lack write permission on your mailbox, you can't delete messages, but others can send messages because of set UID permission on **mail.**

Figure 4-4. Permissions Required for Common Functions

Command	Permissions		Messages/Comments
	Directory	File	
mkdir	-wx	---	"mkdir: cannot access" -- no write or search permission on the directory.
more	--x	r--	"*file*: No such file or directory" -- no search permission on the directory when the path name was used. "*file*: Permission denied" -- no read permission on the file.
mv	-wx	---	"mv: cannot create *file*" -- no write permission on the directory you want to move the file to. "mv: cannot unlink *file*" -- no write permission on the source directory when you try to rename a file.
pr	--x	r--	"pr: can't open *file*" -- no read permission on the file.
rm	-wx	---	"rm: *file* not removed" -- no write permission on the directory. With write permission on the directory but not the file, you are shown the file permissions (in octal representation). If you just press RETURN, the file is saved. If you type "y" and press RETURN, the file is removed. "rm: *file* non-existent" -- no search permission on the directory.
rmdir	-w-	---	Write permission on the parent directory is required.
sort	--x	r--	"sort: can't open file" -- no read permission on the file.
tail	--x	r--	"tail: cannot open input" -- no search permission on the directory or read permission on the file.
vi	-wx	rw-	"Permission denied" -- no search permission on the directory, no read permission on the file, or new file with no write permission on directory. If you have read but not write permission on the file, "read only" is noted.
wc	--x	r--	"wc: can't open file" -- no search permission on the directory or read permission on the file.

Figure 4-4. Permissions Required for Common Functions (Continued)

Locating Files and Directories

This section explains how to locate particular files and directories in the file system.

Using ls and lc

You may use **ls** (list) or **lc** (list in columns) to check the contents of a specific directory. Use the **-a** option if you want to see file names beginning with a dot, since they are not included otherwise.

Locating Subdirectories in Root Directory

You may use this command to list all of the subdirectories in the **root** directory:

```
# ls -l  /  |grep  '^d'
```

The -l option of the **ls** command identifies directories with the "d" symbol as the first character, and '^d' means lines beginning with "d", so this **grep** command selects all directory listings.

Searching through Subtrees

You can use the **lc** -**R** command to check the subdirectories within a directory, and you can use the **find** command to print a list of files by owner, type, or other criteria. The output of the command is simply a list of file names, and it is printed only if you use the **-print** option. Some of the more common options for **find** are described in this manual, and a complete list of options is given in the *XENIX 286 Reference Manual.*

When you use the **find** command, you give the command name, the directory where the search is to start, and options. The search should begin at the lowest possible level because it is time-consuming to search through directories unnecessarily. For example, you should begin the search at the root (/) directory only if you really need to search through the entire directory structure.

Several options take numbers (such as a group ID, user ID, or number of days) as arguments. In these cases, you may give an explicit number, use "+" to mean more than, or use "-" to mean less than. For example, if the option refers to days, +7 means more than 7 days, -7 means fewer than 7 days, and 7 means 7 days.

You can use shell metacharacters such as ? and * but you should enclose the entire file name in quotes if you do. Some people enclose file names in quotes even if metacharacters are not used. See Figure 4-5.

You usually specify what you want to find, but you may also reverse the meaning of your criteria by including an exclamation mark (!). For example, you may want to list all files except ordinary files.

See Figure 4-5 for sample uses of **find**, and see Figure 4-6 for sample output. In the samples in this section, the current directory (.) is the starting point for the search. Notice that the **-print** option is used in most of the examples. No output is printed unless it is included.

Finding Files

```
    # find . -print                    List all files and directories.
```

Finding Files by Type

```
    # find . -type b -print            List block special files.
    # find . -type c -print            List character special files.
    # find . -type d -print            List directory files.
    # find . -type f -print            List ordinary files.
    # find . -type p -print            List named pipes.
```

Finding Files by Name

```
    # find . -name "chap1" -print      List files named "chap1".
    # find . -name "chap*" -print      List files whose names begin with "chap".
```

Finding Files by Owner

```
    # find . -user mary -print         List files owned by "mary".
    # find . -user 205 -print          List files of a user with UID 205.
```

Finding Files by Group

```
    # find . -group admin -print       List files of the "admin" group.
    # find . -group 52 -print          List files of the group with GID 52.
```

Finding Files by Last Access Time

```
    # find . -atime -10 -print         List files accessed less than 10 days ago.
    # find . -atime +10 -print         List files accessed more than 10 days ago.
    # find . -atime 10 -print          List files accessed 10 days ago.
```

Finding Files by Number of Links

```
    # find . -links 3 -print           List files with 3 links.
```

Finding Files That Do Not Match Criteria

```
    # find . ! -type f -print          List files that are not ordinary files.
```

Figure 4-5. Sample **find** Commands

```
# cd /usr
# find . -group pubs -print | more
./spool/mail/jack
./jack
./jack/.profile
./jack/letters
./jack/memos
./jack/newsletters
./jack/newsletters/employee
./jack/newsletters/employee/apr
./jack/newsletters/employee/may
./jack/newsletters/employee/june
./jack/newsletters/customer
./jack/newsletters/customer/apr
./jack/newsletters/customer/may
./jack/newsletters/customer/june
# find . -group pubs -type f -print | more
./spool/mail/jack
./jack/.profile
./jack/newsletters/employee
./jack/newsletters/employee/apr
./jack/newsletters/employee/may
./jack/newsletters/employee/june
./jack/newsletters/customer
./jack/newsletters/customer/apr
./jack/newsletters/customer/may
./jack/newsletters/customer/june
```

Figure 4-6. Sample **find** Output

Finding Files and Executing Commands

The examples in Figure 4-7 show how to use options of the **find** command to locate files and directories. Sometimes you may want to act on the files and directories that are located. Two different options, **-ok** and **-exec**, make this possible.

You can use the **-ok** option when you want to decide how to treat each file. For example, if you are not certain which directory a file is in but know you do not need to see certain directories, you may use this command to find all of the subdirectories in the current directory and decide interactively whether to look at the list produced by the **ls** command:

```
# find -type d -ok ls { } \;
```

When you use the **-ok** option, the { } symbols are replaced by the files identified by **find,** and the expanded command line is displayed with a question mark. The command is executed only if you answer "y". Notice that the "\;" symbols are used to close the command line. Closing this way is necessary when you use the **-ok** option.

You can use the **-exec** option when you want to execute some command with the files that are located by **find.** For example, this removes all access rights other users have to the files in your directories:

```
# find . -exec chmod o-rwx {} \;
```

It is necessary to close the line with the "\;" symbols when you use the **-exec** option.

Working with Non-XENIX Files

If you need to work with non-XENIX files, such as MS-DOS files, see **dd** and **dos** in the *XENIX 286 Reference Manual.*

intel®

CHAPTER 5
WORKING WITH FILE SYSTEMS

File Systems

A file system is a physical partition of a disk that has been initialized by the **mkfs** command. It has a super-block followed by cylinder groups made up of a cylinder group block, an inode list, and data, as described in the *Overview of the XENIX 286 Operating System*. The primary file system is the root file system, which is created automatically during installation and resides on a disk. The root file system contains the root directory (**/**), which is the base of the hierarchical structure of directories that XENIX uses to keep track of the files in a file system. For a description of the root directory, see Appendix C.

A XENIX system may have other file systems as well. Disks greater than 20 megabytes have a separate user file system, created during installation, for user files and for application software. Additional file systems can be created on hard or flexible disks with the **mkfs** command, which is privileged.

You can access files in a file system only if the file system is attached to the root directory hierarchy. During system startup, the root file system is permanently mounted, and the user file system (if one exists) is usually mounted automatically by a **mount** command in **/etc/rc**. Traditionally, the user file system is attached to the **/usr** directory. If you have additional file systems, you may mount them as needed with the **mount** command, attaching them to the **/mnt** directory or to any empty directory (except your working directory) in the root directory structure. If the directory you attach to is not empty, its contents are invisible and inaccessible until you unmount the file system.

Records of mounted file systems are kept in **/etc/mnttab**. The system checks this table of mounted file systems for several commands, but you cannot read it directly because it is a data file. If you want to know what file systems are mounted, use the **mount** command without arguments, as illustrated in Figure 5-3.

Creating a New File System on a Flexible Disk

This section explains how to create a new file system on a flexible disk. If you are adding a new hard disk, see the *XENIX 286 Installation and Configuration Guide* for instructions for configuring it and creating a new file system on it.

It may be appropriate to create a file system on a flexible disk if it will be used for files that are rarely needed or files that most users should not even be aware of. For example, if employee resumes are only used occasionally, you may want to keep resumes on a flexible disk and mount the file system only when the resumes are needed.

Follow these steps to create a new file system on a flexible disk:

1. Log on as **root**. Creating a file system is a privileged function.

2. Format the disk. See "Formatting Disks" in Chapter 7.

3. For a 5¼-inch flexible disk in drive 0, give this command:

   ```
   # /etc/mkfs /dev/dvf0 360 358
   ```

   For an 8-inch flexible disk in drive 0, give this command:

   ```
   # /etc/mkfs /dev/df0 1224 1222
   ```

   The command line includes the command name (**/etc/mkfs**), the device name, the size of the file system in 1,024-byte blocks, and the cylinder group size. With flexible disks, the cylinder group size is always two blocks less than the file system size (ignoring any fractions). (See Appendix A for a list of device names and disk sizes.) Two additional values can be specified (see **mkfs** in the *XENIX 286 Reference Manual*), but you should use the defaults for a flexible disk.

   If the device has data, you are warned with this prompt:

   ```
   mkfs: device contains data. Overwrite?(y/n)
   ```

   If the data is unnecessary, answer "y" to continue. The data will be erased. If you want to save the data, answer "n" to stop without creating a file system.

Mounting and Unmounting a File System

When you want to use the files on a flexible disk, insert the disk and use the **mount** command to mount the secondary file system. When you no longer need the files, use the **umount** (yes, that is the correct spelling) command to unmount them. You have to mount a file system before you can use it and unmount it when you are finished. Mounting does not move any files. It just connects the directories on the secondary file system to a directory already mounted on the root directory hierarchy.

Some mounting and unmounting is done automatically. The root file system is mounted during startup, and you cannot unmount it. File systems that are used regularly are usually mounted with a **mount** command in the **/etc/rc** file. All mounted file systems, except the root file system, are unmounted automatically if you use the **haltsys** or **shutdown** command.

Remember this important rule: **Always unmount a file system before removing the flexible disk it resides on.** Removing a disk whose file system is still mounted is like turning the system off without shutting it down properly. It results in an unclean file system, which you will have to clean the next time you try to mount the disk.

Mounting a File System

Mounting a file means attaching it to a directory in the root hierarchy. The directory you attach to should be an empty one. Follow these steps to mount a file system:

1. If the file system is on a flexible disk, insert the disk in the appropriate drive.

2. Log on as **root.**

3. Use the **mount** command with the name of the device and the name of the empty directory the system is to be attached to. You may use the **/mnt** directory, which is predefined for mounting file systems, or some other directory. When you give the command, specify the block device, then the directory. For example, this attaches the file system from the 5¼-inch flexible disk in drive 0 onto the **/mnt** directory:

   ```
   # /etc/mount /dev/dvf0 /mnt
   ```

   Note: If the disk is write-protected, include the **-r** option. For example:

   ```
   # /etc/mount -r /dev/dvf0 /mnt
   ```

   If the structure needs cleaning, this warning appears:

   ```
   mount: Structure needs cleaning
   ```

   Use **fsck**, then try to mount it again.

   If the file system has already been mounted (it cannot be mounted twice) or any user is in the directory, this prompt appears:

   ```
   mount: Device busy
   ```

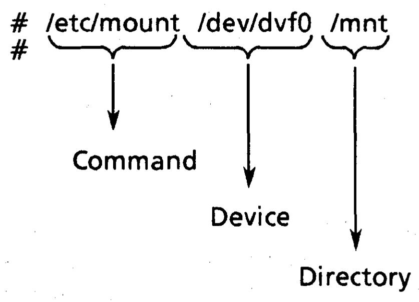
   When the user leaves the directory, you may try the **mount** command again.

4. Use the **cd** command to change to the directory with the newly mounted file system, then use the **l** command to list the contents of the directory.

   When you have finished working in the directory, use the **cd** command to move to another directory so you can unmount the disk.

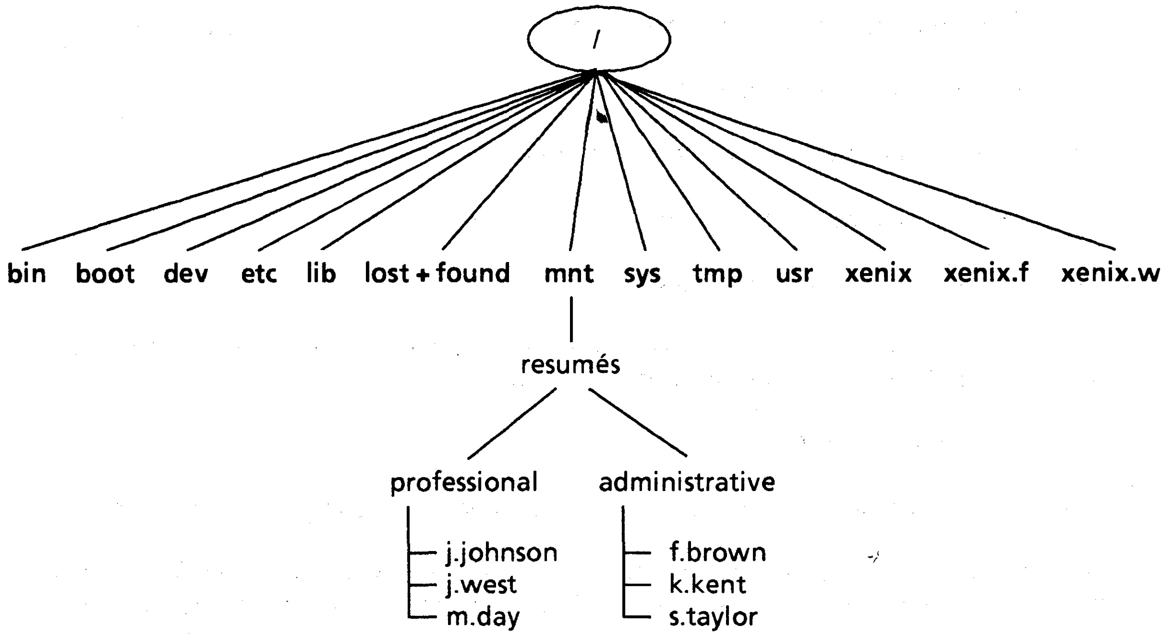
   Note: **Leave the disk in the drive until you have unmounted the file system.**

Figure 5-1 gives a sample dialogue for mounting a file system on a flexible disk, Figure 5-2 illustrates how the file system has been attached to the **/mnt** directory, and Figure 5-3 shows how to check mounted file systems.



F-0336

Figure 5-1. Sample **mount** Command



F-0323

Figure 5-2. Mounted File System

```
# /etc/mount
/dev/root on / read/write on Tue Sep 4 15:43:00 1984
/dev/usr on /usr read/write on Tue Sep 4 15:43:39 1984
/dev/dvf0 on /mnt read/write on Tue Set 4 15:43:50
```

Figure 5-3. Checking Mounted File Systems

Unmounting a File System

Be sure to unmount a secondary file system before you remove the flexible disk from the disk drive. Unmounting a secondary file system removes it from the directory in the root directory hierarchy. No files are moved or destroyed. The directory remains part of the root directory hierarchy, but it has no files.

To unmount a file system, follow these steps:

1. Leave the disk in the drive.

2. Log on as **root.**

3. Give the **umount** command with the name of the device. For example, you would use this command to unmount the 5¼-inch flexible disk in drive 0:

   ```
   # /etc/umount  /dev/dvf0
   ```

   If files or directories are being used, you will be told the device is busy. You cannot unmount a file system that is in use.

4. Remove the disk.

intel®

CHAPTER 6
PROVIDING SYSTEM SECURITY

Security Issues

As the system administrator, you are responsible for the computer hardware, the software, and all of the data on your system. You should take all necessary precautions to protect the system from physical damage, loss of data, and unauthorized access to data. This chapter explains what problems you should be aware of and gives suggestions for solving them. Chapter 4 has additional information about protecting files.

Protecting Devices

Damaged disk drives and disks can corrupt a file system and cause users to lose data. You can protect the physical components of the computer by taking these precautions:

- Restrict the computer area, if necessary. Computer systems are sometimes kept in locked rooms.

- Allow only essential personnel in the computer area.

- Keep backup disks or tapes organized and locked up. They should not be stored with the computer itself. Some administrators keep backups in safes, and some keep offsite backups as insurance against fire, theft, or other onsite problems.

- Keep data storage devices away from magnetism, direct sunlight, and severe changes in temperature.

- After you have attached a label to a flexible disk, do not use a pencil or a ball point pen to write on the label.

Using Super-User Powers Carefully

When you log on as **root**, you have full power over all files in the system. This power makes it possible for you to solve problems, but it also poses a potential security risk, because you can destroy data if you are not careful. Moreover, anyone who gets the **root** password has the same power. Following these guidelines should help you prevent abuse of super-user powers:

- Give others the **root** password only when absolutely necessary.

- Change the **root** password often, but do not write it down, and do not forget it. If you forget the **root** password, you have to install the system again.

- Log on as **root** only when absolutely necessary. Chapter 1 explains which functions are privileged.

- Always log off after finishing privileged functions. If you leave your terminal when you are still logged on, anyone who comes to the terminal can act as **root.**

- Be sure that you make the working directory (.) the last element in **root**'s search path, if you include it at all. The reason for this is that the system goes through your search path in sequence when you give a command. The first program with the command name is the command that is executed. A clever user could put a special program in his or her working directory and give it a name like **ls,** assuming that at some point you would be in the directory and try to execute the real **ls** (which is in the **/bin** directory). If the working directory came before **/bin** in the search path, the user's **ls** program would be executed instead of **/bin/ls.** You can imagine the danger, if, for example, the user's program had instructions for removing system files.

- Do not change access permissions on the devices in the **/dev** directory. These permissions are set so the owner (**root, sysinfo,** or **asg**) has read and write access, while all others are denied access. If users had access to these devices, they could go into any files in the system.

- When you are logged on as **root,** be very careful when you use metacharacters such as * and be very careful with command syntax or you may destroy files you did not intend to destroy.

- Insist that all users have passwords. You can control this because only **root** can give a user a blank password. As **root** you can also set the minimum length of a password with the **PASSLENGTH** variable in the **/etc/default/passwd** file.

- Force users to change passwords at regular intervals. You may do this with the **pwadmin** command. (See Chapter 3.)

- Do not change permissions on files in the directories created during installation unless you have a very good reason for doing so and fully understand the consequences.

- Be particularly careful with set UID and set GID permissions on a command, since they give anyone who invokes it the powers of the owner or group. The permissions are set correctly initially, and you should not change them. If you remove the permissions, some commands may fail to work. If you add the permissions, someone may gain unauthorized access to files and commands.

- Check set UID permissions. At regular intervals, print a hard copy of files that have set UID permission. Compare the list to the previous list to see whether someone has set UID permissions that allow security violations. You may use this command to print a hard copy of special files and files with set UID permission:

  ```
  # ncheck -s | lpr
  ```

  By default, the command includes files on all mounted file systems. You may use this command to print a hard copy of all files with set UID or set GID permission in all mounted file systems:

  ```
  # find / \( -perm -02000 0 -perm 04000 \) -exec ls -l { } \; | lpr
  ```

  If you put this command in a shell script, check the shell script itself to be sure that it has not been changed.

- Pay close attention to access issues when you set up a computer network.
- Monitor the use of the **su** command. (See Chapter 10.)
- Assign users a restricted shell if you want to keep them in one directory and limit their use of XENIX commands. (See **rsh** in the *XENIX 286 Reference Manual.*)

Advising Users

The system is safe only if users can protect their data from those who should not see it, use it, or change it. They may not be aware of potential security hazards, so it helps to give them guidelines. For example:

- Remind users to log out before leaving their terminals.
- Encourage users to select difficult passwords that include letters, digits, and punctuation marks.
- Help users understand permissions. New users may understand protections on files but not on directories. Since a combination of file and directory permissions controls file access, users need to be aware of both. See Chapter 4 for a summary of permissions required for common functions.
- Suggest that users make the current directory the last item in their search paths, or not include it at all.
- Recommend that users encrypt sensitive files with the **crypt** command. The user who gives this command defines some key to the encryption and no one, not even **root**, can decode encrypted files without the key except by cryptanalysis.

Backing Up Data

One of your most important responsibilities is to keep duplicate copies of the files on your system. These duplicate copies are called backups, and they are so important that Chapter 7 is dedicated to backing up. For additional security, keep offsite backups to safeguard data if onsite data is destroyed.

Protecting File Systems

Data is useful only if the system's control information about files and directories is accurate so the data can be accessed. You can take these measures:

- Shut your system down properly with **shutdown** so the file system is updated with information in the buffers.
- Use the **fsck** command on a regular basis to check the integrity of the file system. Many file system errors can be corrected if **fsck** finds them soon enough.
- Clean the file system if prompted that it is unclean. (To do this, answer "y", for "yes", when asked whether to proceed with cleaning.) If errors cannot be corrected, corrupted files are deleted, which is an important reason for keeping backups. (See Chapter 9 for information about file system problems.)

intel®

CHAPTER 7
BACKING UP

Commands for Copying Files

Copying files is a basic system function, and XENIX offers several commands that are useful under different circumstances. You may copy files because you need to save one version and revise another, or transmit files to another system, or maintain duplicates as a form of insurance. Maintaining duplicate files is usually called backing up, and it is one of your most important functions. If a serious hardware failure or operator error destroys important files, you need to be able to recover them from current backups. This chapter is essentially about backing up, but the other commands for copying are summarized for your convenience, and you may check the *XENIX 286 Reference Manual* for more details. The commands for copying files are

- **cp.** Any user who has read permission on a file can invoke **cp** to copy it. The user can also copy it into another directory as long as he or she has write permission on that directory. For example, imagine that a user makes monthly reports that have similar format and content. The most efficient procedure may be to copy the old report and edit the copy for the new month. This sample command makes a copy of "may.report" in the working directory and calls it "june.report":

  ```
  $ cp may.report   june.report
  ```

  This example places a copy of "may.report" in the "sales" directory and names the copy "may.report":

  ```
  $ cp may.report   sales
  ```

- **copy.** Any user with the correct permissions can copy files or directories.

- **cpio.** The **cpio** command is not intended for backing up files, but it is useful as a filter. For example, this command copies the contents of a directory into an archive (the **-o** option means copy out):

  ```
  $ ls | cpio -o > /dev/dvf0
  ```

  The **cpio** can be used with **find.** For example, this dialogue duplicates the "sales" directory and calls the copy "sales.nw":

  ```
  $ cd sales
  $ find . -print | cpio -pd sales.nw
  ```

  The normal operation is to copy to or from an archive, and the **-p** option is necessary if you are not using an archive. The **d** option creates directories as needed.

- **tar.** The term **tar** is a shortened form of "tape archiver", but you may use **tar** to copy any number of files onto a formatted disk or tape or into a file. With **tar,** you can move files onto a device without creating a file system on the device. The **tar** command itself keeps the information it needs about files and directories, but only **tar** can interpret the information. This means that if you use **tar** to store files on a disk, you can read the contents of the disk only with **tar.**

  The **tar** command is useful because you can use it to copy data into a special file, such as a disk, then take the disk to another machine. It is also good for copying entire disks or directories.

  For backing up files on a system with multiple file systems, **sysadmin** (see below) is the preferred method, but **tar** can be used and is appropriate for installations that have only one or two users who work on a few files. The users can keep backups of those files without backing up the entire file system on a regular basis. Using **tar** is the recommended method for systems that have only the root file system.

- **sysadmin.** The **sysadmin** command presents a menu of options related to backing up and is the recommended method of backing up file systems in most installations. With **sysadmin,** you make daily and full (periodic) backups. You should use **sysadmin** if several users work on many different files. Strategies and procedures are discussed later in this chapter. You may also use **sysadmin** to print a list of files that were backed up and to restore individual files.

- **dump.** You may use **dump** to copy an entire file system onto disk or tape and use **restor** when you want to restore the file system or individual files. The **dump/restor** method of backing up is appropriate if you need daily, weekly, and monthly backups. Since **sysadmin** is a shell script based on **dump** and **restor,** a file system can be restored with **restor** even if it has been backed up with **sysadmin.**

Strategies for Backups

The purpose of backing up is to keep copies of files so you can go back to them if files are destroyed. You can assume that at some time you will need backups, and you will be very glad to have them. When you restore files from backups, users have to redo work done since the backup. This forms the basis for your backup strategy. Back up often enough to keep the amount of work that has to be redone to a minimum.

If your system has few users and they work on a limited number of files, you may prefer to back up files as necessary with **tar.** If several users work on many files, backing up the file system on a regular basis and at a scheduled time is essential. Plan the schedule well in advance, and edit the message of the day in **/etc/motd** to remind users the day before and the day of the full backup.

A typical schedule includes a daily backup and a full backup once a week. The daily backup copies only the files modified since a more comprehensive backup. The full backup copies all files in the file system. It is advisable to do a full backup once a week, and it is essential to do a full backup at least once a month. If you do full backups each week, daily backups should not take long because few files should have changed since the last full backup.

Put the system in single-user mode for any backup because files in use are not copied and the archive set can become corrupted if there is activity during a backup.

System activity is usually concentrated in the user file system, so it needs to be backed up often, usually daily and weekly. The root file system does not need to be backed up frequently unless files in it are being changed.

When you do a backup, record it in a log such as one of those in Appendix B. Label disks or tapes with the date and the files and directories copied. System administrators often keep daily backups for two weeks and periodic backups indefinitely. After a backup has expired, you may use the disk or tape for new backups.

Formatting Disks

Either flexible disks or tape can be used as archive devices for backups. Disks must be formatted before their first use, so choose disks that are blank or have no useful data, and be sure the disks are write-enabled. The 8-inch disks are write-enabled if they have write-enable tabs, and the 5¼-inch disks are write-enabled if they do not have tabs. Formatting erases any data on the disk.

The recommended practice is to format disks as needed during the **sysadmin** or **dump** backup, but disks can be formatted in advance with the **format** command. A disk always has to be treated as a raw device for formatting, so the device name has to begin with "r" (for "raw"). The first sample command could be used to format a 5¼-inch flexible disk in drive 0, and the second sample command could be used to format an 8-inch disk:

```
$ /etc/format  /dev/rdvf0
$ /etc/format  /dev/rdf0
```

If the disk is not inserted correctly, an error message explains that the device cannot be opened, and the disk is not formatted. Insert the next disk and give the command again. Repeat until you have formatted a full set of disks. The number depends on the number and size of files. A 40-megabyte disk that was 85 percent full would require 95 5¼-inch disks, and a 35-megabyte disk that was 85 percent full would require 25 8-inch disks.

Verifying Disks

The **format** program does not check for bad blocks, so some administrators verify disks before using them for backups. After you have formatted a 5¼-inch disk, you may give this command to check whether the disk is useful:

```
dd if=/dev/rdvf0 of=/dev/null bs=20b
```

After you have formatted an 8-inch disk, you may check the disk with this command:

```
dd if=/dev/rdf0 of=/dev/null bs=20b
```

Expect to see the number of disks read and written to. An I/O error message means the disk should be discarded.

Backing Up with tar

You can use **tar** to copy files, read the contents of a disk with copied files (and you can only read the contents with **tar**), and restore files. The options used most often are described below:

c The **c** option causes writing to begin at the beginning of the archive device instead of after the last file. Metacharacters can be used.

u The **u** option adds the named files to the archive if they are not already there or if they have been modified since last written on that archive. Metacharacters can be used.

v The **v** option causes the name of each copied file to be printed to the screen.

k The **k** option is used to define the size of the archive volume when it is a disk. The size must be a multiple of one-half the blocking factor. For example, if the blocking factor is 20, the value of **k** has to be a multiple of 10. The **k** option is necessary for any backup that exceeds one archive volume.

f The **f** option is included so the archive device can be named.

b If you specify a raw device, you must use the **b** option and give the blocking factor. The recommended factor is 20 for disks and 60 for tapes.

t The **t** option causes the names of the specified files to be listed each time they occur on the archive.

x The **x** option causes the named files to be extracted from the archive. If no files are named, all files on the archive are extracted. If several files on the archive have the same name, the last one overwrites all earlier ones. Metacharacters cannot be used.

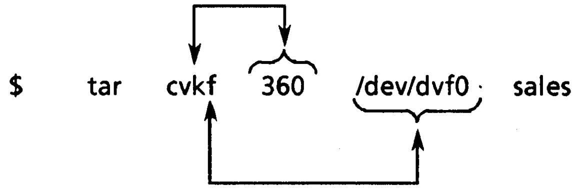
n The **n** option is useful if the archive device is not a tape.

When you use **tar** to copy files, include the **cvf** options, identify the device, and list the files. If the device is a disk, include the **k** option and give the disk size.

You can read the contents of the archive device with the **tvf** options. If the archive device is not a tape, the listing will be faster if the **n** option is included.

You can restore all of the files on the archive device with the **xvf** options.

The syntax for **tar** requires you to name the options, then specify appropriate arguments in the same order as the options. See Figure 7-1 and Figure 7-2.



F-0338

Figure 7-1. Sample **tar** Command Syntax

$ tar cvkf 360 /dev/dvf0 sales

This command copies the "sales" directory and its subdirectories onto the 5¼-inch flexible disk in drive 0 (assuming the disk is formatted, which it must be). The disk size is 360 kilobytes.

$ tar cvkf 1220 /dev/df0 sales

This command copies the "sales" directory and its subdirectories onto the 8-inch flexible disk in drive 0 (assuming the disk is formatted, which it must be). The disk size is 1220 kilobytes.

$ tar cvbf 60 /dev/rmt0 sales

This command copies the "sales" directory and its subdirectories onto tape. The blocking factor is 20.

$ tar tvnf /dev/dvf0

This command lists the contents of the archive device, which is a 5¼-inch flexible disk. The n option works with disks only.

$ tar tvf /dev/dvf0

This command reads the entire archive to verify the accuracy of the copy onto a 5¼-inch flexible disk.

$ tar tvf /dev/rmt0

This command gives the contents of the archive device, which is tape.

$ tar xvf /dev/dvf0

This command copies the contents of the archive device, which is a 5¼-inch flexible disk, into the file system. The files are restored with the names they had when put on the disk.

Figure 7-2. Sample **tar** Commands

Defaults

Before using **dump** or **sysadmin**, you should be aware of the variables in **/etc/default/dump.** These variables are used as defaults by **dump** and **sysadmin**, and you can use a text editor such as **ed** or **vi** to change them:

archive The device or file that receives the backup. For a list of supported devices, see Appendix A. If disks are the archive media, choose the raw form of the name (with "r" at the beginning) so the disk can be formatted, if necessary, during a backup. For example, if **archive** is **/dev/rdvf0**, you can format a 5¼-inch flexible disk during a **dump** or **sysadmin** backup. If the device is **/dev/dvf0**, you cannot format during a backup.

filesystem The file system to be backed up. This should not be a raw device.

kbytes For block devices, the number of 1,024-byte blocks the backup media will hold. For tape backups, this is 0.

tapesize The length of the tape, in feet. For backups onto disks, this is 0.

density The density of the tape, in bytes per inch (BPI). For backups onto disks, this is 0.

level The level (0-9) of the backup. See an explanation of levels later in this chapter.

record Whether the date and level of the backup are to be recorded automatically in the **/etc/ddate** file. The value is 0 for no or 1 for yes. If the date and level are not recorded, the next backup will automatically be a full backup. Setting this variable to 1 is equivalent to using the **u** option with **dump.** (The **dump** command and its options are described later in this chapter.)

If you plan to use **sysadmin**, go into **/etc/default/dump** and be sure you have the right variable settings because you cannot override any defaults (except **level**) during the **sysadmin** program.

Figure 7-3 has sample variables for a system that uses 5¼-inch flexible disks for backups, and Figure 7-4 has sample variables for a system that uses tape. The lines beginning with # are comments. A system with 8-inch flexible disks would use the defaults in Figure 7-3, except that the archive would be **/dev/rdf0** and the number of K bytes would be 1220.

```
# cat /etc/default/dump
# the backup media, or file
archive = /dev/rdvf0

# the file system to be backed up.  Shouldn't be a raw device.
filesystem = /dev/usr

# K bytes the backup media will hold (for block devices)
kbytes = 360

# The tape length in feet.
tapesize = 0

# Density of the tape in Bytes Per Inch
density = 0

# Incremental level of the dump
level = 9

# Whether to record the dump or not (0 = no, 1 = yes)
record = 1
```

Figure 7-3. Sample Defaults for 5¼-Inch Disk Backup

```
# cat /etc/default/dump
# the backup media, or file
archive = /dev/rmt0

# the file system to be backed up.  Shouldn't be a raw device.
filesystem = /dev/usr

# K bytes the backup media will hold (for block devices)
kbytes = 0

# The tape length in feet.
tapesize = 2300

# Density of the tape in Bytes Per Inch
density = 16000

# Incremental level of the dump
level = 9

# Whether to record the dump or not (0 = no, 1 = yes)
record = 1
```

Figure 7-4. Sample Defaults for Tape Backup

Using sysadmin

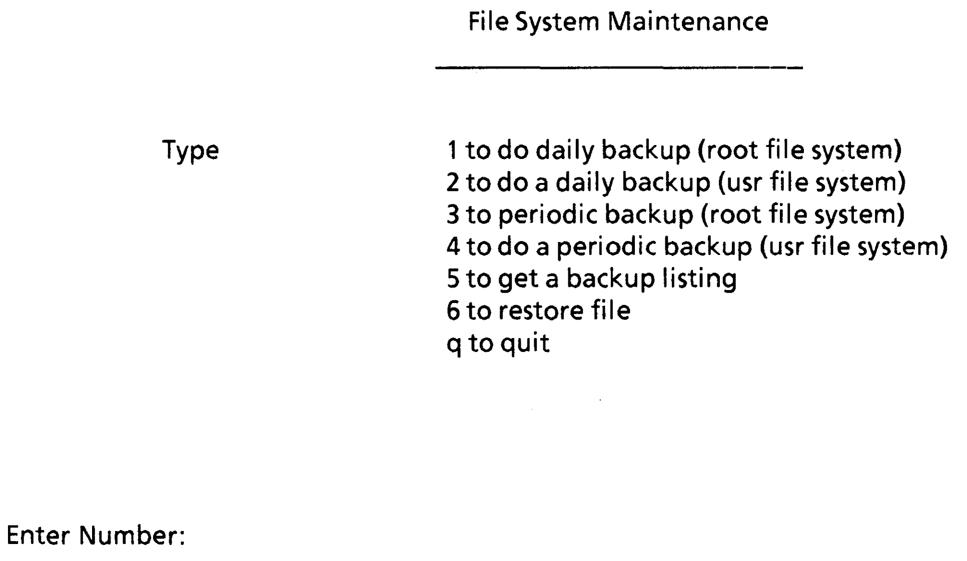
The **sysadmin** backup is recommended for most installations because it is easy to use and offers both daily and full backups. It is a menu-driven backup based on the **dump** program. Technically, a daily backup is a level 9 backup and a periodic backup is a level 0 backup. (See "Backup Levels" later in this chapter.)

Before you try to use **sysadmin**, read the previous sections in this chapter. When it is time to do a backup, log on as **root**, use **wall** (write to all users) to warn users, then use this command to go into single-user mode (giving the number of minutes, 0-15, or omitting the number for a default of 5):

```
# shutdown 7 su
```

This shuts the system down in 7 minutes (or sooner, if all users except the console log off before 7 minutes have elapsed) and leaves you as **root** in single-user mode.

After you are in single-user mode, use the **sysadmin** command to bring up the menu of options shown in Figure 7-5 and described in the following sections. You may be in any directory when you invoke **sysadmin.** The first backup should be a periodic backup, which backs up the entire file system.



```
                       File System Maintenance
                       -----------------------

        Type           1 to do daily backup (root file system)
                       2 to do a daily backup (usr file system)
                       3 to periodic backup (root file system)
                       4 to do a periodic backup (usr file system)
                       5 to get a backup listing
                       6 to restore file
                       q to quit



Enter Number:
```

Figure 7-5. Menu of **sysadmin** Options

Daily Backup

A daily backup copies each file that has changed since the last periodic backup. Each day at a specified time, do a daily backup of the user file system and record the backup in a log. (See logs in Appendix B.)

Go into single-user mode by giving this command (changing the number of minutes, if desired, from 0-15):

```
# shutdown 7 su
```

The screen warns users of the shutdown and soon after everyone (except the console) has logged off, the shutdown begins, even if the number of minutes has not elapsed. Several messages are displayed as the shutdown occurs. When the file system is identified with the date, press CONTROL-D to go into single-user mode. At the **root** prompt (usually #), give this command:

```
# sysadmin
```

Select the daily backup of the file system from the menu. Label the first disk or tape "Volume 1" and insert it in the drive when prompted. Since you are doing a daily backup, the current date and the date of the last full backup are shown. If you are doing a full (periodic) backup, the date is shown as "the epoch". Two conditions could cause you to have a full backup even though you selected a daily backup: the file system has never been backed up before, or the backup date in **/etc/ddate** is blank.

Files are copied onto the archive device until it is full, and if another device is needed, you are prompted to change volumes. When this happens, remove the disk or tape. Label the next one as "Volume 2" and insert it. Repeat, increasing the volume number by one each time, until the screen displays "DONE".

When you are finished, you can press CONTROL-D to go into multiuser mode.

Periodic Backup

Each week (or at least each month), do a periodic backup to copy an entire file system. Go into single-user mode as shown above, then select the appropriate periodic backup from the menu. Since you are doing a periodic backup, the date is called "the epoch".

Label the first disk or tape "Volume 1", and insert it in the drive when prompted. The date is "the epoch", which means the backup will be a full backup of the file system. Files are copied onto the archive device until it is full, and if another device is needed, you are prompted to change volumes. When this happens, remove the disk or tape. Label the next one as "Volume 2" and insert it. Repeat, increasing the volume number by one each time, until the screen displays "DONE".

When you are finished, you can press CONTROL-D to go into multiuser mode.

Backup Listing

After doing a backup, you may print a list of the copied files. When you use this selection, you are prompted to insert the first volume and press ENTER (you can press ENTER or RETURN). As the header information in Volume 1 is read, a list of files and any error messages are placed in the **/tmp/backup.list** file, which is created automatically. If a message says "Tape is not a dump tape", it means the disk has the wrong format. (For example, if you copy files with **tar**, you cannot use **sysadmin** to get a backup listing.) If you want a hard copy of the list, give this command:

```
# lpr /tmp/backup.list
```

Just remove the list, when it is no longer needed, with this command:

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

If you do not remove it before the system comes up in multiuser mode again, it will probably be removed automatically, since the **/etc/rc** file usually has instructions for removing the files in the **/tmp** directory.

Restoring a File

Occasionally, an administrator needs to go to a backup to recover a file because a user has deleted it by mistake or some other serious problem has occurred. If few files need to be restored, you can use **sysadmin,** select the restore file option from the menu, and respond to the prompts on the screen. Notice that you need to give the full path name of the file. You can only restore one file at a time in this way.

If you need to restore an entire file system, use the **restor** command described later in this chapter.

Using dump

For many installations, daily and weekly backups are sufficient and can be made easily with **sysadmin**. However, if daily, weekly, and monthly backups are required, **dump** is the appropriate backup method.

Backup Levels

With **dump**, the backup levels are numbered 0-9, and a level 0 backup is a full backup that copies all files in the file system. The other levels are used with the last backup date to define which files are copied. The program copies files that have changed since the last backup with a lower level number. For example, imagine this sequence of events:

- The file system is backed up for the first time on October 20 and the level is 0. All files are copied.
- The file system is backed up on October 23 and the level is 5. All files that have changed since the level 0 backup are copied.
- The file system is backed up on October 24 and the level is 6. All files that have changed since the level 5 backup are copied.
- The file system is backed up on October 25 and the level is 7. All files that have changed since the level 6 backup are copied.
- The file system is backed up on October 26 and the level is 8. All files that have changed since the level 7 backup are copied.
- The file system is backed up on October 27 and the level is 1. All files that have changed since the level 0 backup are copied.
- The file system is backed up on October 28 and the level is 5. All files that have changed since the level 1 backup are copied.

Figure 7-6 illustrates two months of daily, weekly, and monthly backups. The daily backups have levels 5-8, the weekly backups have levels 1-3, and the monthly backup has level 0. This sample is based on a schedule that always has a level 0 backup on a weekend following a level 3 weekly backup and omits a backup on the Monday following the level 0 backup. The full backup is done on a weekend because it usually takes a significant amount of time and the administrator does not want to keep users off the system during the week. In the next month's schedule, the first weekly backup is level 2 because the previous weekly backup was level 1.

The schedule in Figure 7-6 is just a sample that is included to help you understand backup levels. You will want to create a schedule based on the level of activity, the amount of time to be spent backing up, and the number of backup sets required to restore a file system.

Sun	Mon	Tues	Wed	Thurs	Fri	Sat
	1 weekly level 1	2 daily level 5	3 daily level 6	4 daily level 7	5 daily level 8	6
7	8 weekly level 2	9 daily level 5	10 daily level 6	11 daily level 7	12 daily level 8	13
14	15 weekly level 3	16 daily level 5	17 daily level6	18 daily level 7	19 daily level 8	20 monthly level 0
21	22 skip	23 daily level 5	24 daily level 6	25 daily level 7	26 daily level 8	27
28	29 weekly level 1	30 daily level 5	31 daily level 6	1 daily level 7	2 daily level 8	3
4	5 weekly level 2	6 daily level 5	7 daily level 6	8 daily level 7	9 daily level 8	10
11	12 weekly level 3	13 daily level 5	14 daily level 6	15 daily level 7	16 daily level 8	17 monthly level 0
18	19 skip	20 daily level 5	21 daily level 6	22 daily level 7	23 daily level 8	24
25	26 weekly level 1	27 daily level 5	28 daily level 6	29 daily level 7	30 daily level 8	

F-0324

Figure 7-6. Sample Schedule of Backups

Using dump with Defaults

If you use the **dump** command without arguments, the default values in **/etc/default/dump** are used. The screen displays the current date and the dump date, which is the date the file system was last backed up. If the level was 0, the dump date is called the epoch. The screen prompts you for volumes as needed and gives you three options: continue, format, or quit. If the archive device has never been formatted, select the **f** option for format (the device name has to include "r" for raw), then continue. As the backup progresses, its four stages are identified on the screen. If the volume fills, you are prompted for another one and the process is repeated. Figure 7-7 illustrates a sample dialogue for a periodic (level 0) backup that involves 7 backup disks. For brevity, only the first four volumes are mentioned. The backup is based on the defaults shown in Figure 7-3.

```
# dump 0
      date = Thu Sep 6 08:51:18 1984
dump date = the epoch
Level 0 dump from /dev/usr to /dev/rdvf0
Archive device holds 360K
Dump will be recorded in /etc/ddate
Insert volume 1, then press <cr> to continue, f to format, or q to quit: f
Formatting volume... Formatted 80 tracks: 0 thru 79, interleave 4.
Insert volume 1, then press <cr> to continue, f to format, or q to quit:
RETURN
I mapping regular files
II mapping directories
estimated 7244 K on 21 volume(s)
III dumping directories
IV dumping regular files
Insert volume 2, then press <cr> to continue, f to format, or q to quit: f
Formatting volume... Formatted 80 tracks: 0 thru 79, interleave 4.
Insert volume 3, then press <cr> to continue, f to format, or q to quit: Formatting volume...
Formatted 80 tracks: 0 thru 79, interleave 4.
Insert volume 4, then press <cr> to continue, f to format, or q to quit:
RETURN
Formatting volume... Formatted 80 tracks: 0 thru 79, interleave 4.
 .
 .
 .
level 0 dump on Thu Sep 6 08:51:18 1984
DONE
7296K on 21 volume(s)

#
```

Figure 7-7. Sample **dump** Dialogue

Using dump with Options and Arguments

The recommended procedure is to set up defaults for **dump** in **/etc/default/dump,** but you can specify the appropriate information in the command line, which is made up of the command, options, and arguments. The options are described in Figure 7-8. The syntax for **dump** requires you to name the appropriate options, then specify appropriate arguments in the same order as the options. See Figure 7-9 for an example.

0-9 Use the appropriate number for the backup level.

f Use **f** if you want to back up onto a file or device other than the default.

u Use **u** if you want the date and level of the backup to be recorded in **/etc/ddate**.

s Use **s** for tape size if you are backing up onto tape.

d Use **d** for density if you are backing up onto tape.

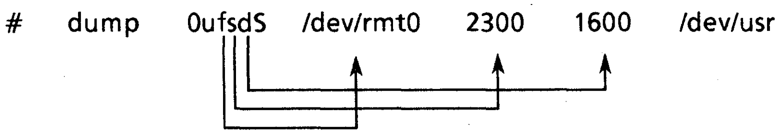
k Use **k** for disk size in 1,024-byte blocks if you are backing up onto a block device such as a flexible disk.

p Use **p** if you want to verify the archive media before you use it. This option checks for bad blocks.

w Use **w** to check backup levels and dates.

S Use **S** if the backup media is a streamer tape.

Figure 7-8. Options for **dump**



```
#    dump    0ufsdS   /dev/rmt0    2300    1600    /dev/usr
```

Figure 7-9. Sample **dump** Command

Checking the Validity of a Backup

After you have done a backup with **sysadmin** or **dump,** you should use the **restor c** command to compare the file system and the backup. If you do it soon after the backup, the two should be almost the same. Figure 7-10 explains some of the messages that may appear.

---

clear on dump media, not clear on filesystem

The file is in the file system but is not included in the backup.

clear on filesystem, not clear on dump media

After the backup, the file was moved or removed from the file system.

inode and data changed during or since dump

Since the backup, the file has changed. It may have changed during or after the backup.

inode (but not data) changed during dump

Someone may have modified the time but not changed any data.

data (but not inode) changed during dump

Someone may have changed information such as the mode, links, or user ID.

Figure 7-10. Messages from **restor c**

Checking Archives

To check whether a disk has been written by **dump** or **sysadmin,** use this command:

```
# dump p
```

To check whether a 5¼-inch disk has been written by **tar,** use this command:

```
# tar tvf /dev/dvf0
```

To check whether an 8-inch disk has been written by **tar,** use this command:

```
# tar tvf /dev/df0
```

Checking Backup Levels and Dates

You may use three different commands to check backup dates and levels as recorded in **/etc/ddate.** They are **sysadmin** (includes a menu selection for a backup listing), **dump w,** and **sddate.**

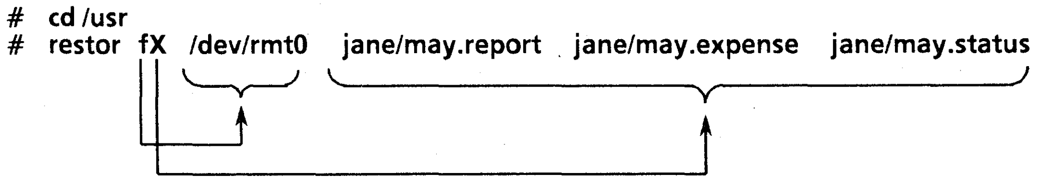
If you want a list of the files that were backed up, you can use the **dumpdir** command. It lists names and inode numbers of all files and directories copied during a dump. If it does not give you this information, the header record in Volume 1 has probably been damaged and you should do the backup again.

Restoring Files with restor

If you need to restore an individual file, you can use the **sysadmin** command and select the option for restoring files. However, if you need to restore directories or an entire user file system backed up with **dump** or **sysadmin,** use **restor.** You can also restore files and directories into the root file system, but you cannot restore the entire root file system.

You can set the default for the archive device in **/etc/default/restor,** but you give the remaining information on the command line. The syntax for **restor** is like the syntax for **dump.** You give the command name, then the appropriate options, then the arguments that correspond to the options.

You must use one, and only one, of these options: **r, R, t, T, x, X, c.** If desired, you may use the **f** option in combination with one of the other options. The syntax for **restor** requires you to name the appropriate options, then specify appropriate arguments in the same order as the options. See Figure 7-11 for an example of a command that restores three particular files with their original names from the tape archive, and see Figure 7-12 for the options for **restor.**



```
# cd /usr
# restor fX /dev/rmt0 jane/may.report jane/may.expense jane/may.status
```

F-0326

Figure 7-11. Sample **restor** Command

f Use **f** if you want to define the archive file or device instead of using the default.

r Use r if you want to restore a file system. If you use this option, specify the file system as an argument. You cannot restore the root file system.

R Use **R** if you want to restore a file system and specify the starting volume of a multivolume set. You will be prompted for a starting volume number. If you use this option, specify the file system as an argument. Since you can specify the starting volume number, you do not have to do the complete restoration without interruption. You can start it, stop without finishing, run **fsck** (this is necessary), and start again with the next volume number.

t Use t to print the date of the backup and the date of the last backup of that level. When prompted, insert Volume 1 and press RETURN. The two dates are displayed on the screen.

T Use **T** to show the date of the backup, the date of the last backup of that level, and all files dumped, with names and inode numbers.

x Use **x** to restore particular files. If you use this option, identify the file names as arguments, but do not include the first part of the path name. For example, if you want to restore **/usr/bin/lpr**, give **/bin/lpr** as the argument. When you use **x**, the restored file is listed in the directory under its inode number. For example, the **/bin/lpr file** would be listed by number rather than by name.

X Use **X** instead of **x** if you want the restored file to have its original name in the directory. Use **X**, too, if you want to restore an entire directory and all of its subdirectories.

c Use **c** to compare the restored file system and the archive device and report any inconsistencies.

Figure 7-12. Options for **restor**

When you want to restore files, follow this procedure:

1. Log on as **root.**
2. Insert Volume 1 of the backup archives. It has header information.
3. Give the **restor** command with the appropriate options and arguments.
4. Follow the directions on the screen.

intel®

CHAPTER 8
MONITORING SYSTEM USE

Keeping Enough Free Space

XENIX requires a certain amount of free space--15 percent at a minimum--to function efficiently. As the amount of free space approaches the minimum, system operation usually becomes sluggish. This chapter explains how to maintain enough free space.

If space shortage is a chronic problem, you may want to keep secondary file systems on flexible disks and mount them when you need them (see Chapter 5) or upgrade your system with a bigger hard disk.

Checking Free Space

Space is measured in blocks, and each block has 1,024 bytes of information. The less space you have, the more closely you should monitor it. The commands discussed below give you information about the number of blocks that are free or used.

Checking Blocks Free with df -t

Use the **df -t** command to find the number of free blocks on one or more file systems. This is a non-privileged command, and can be used with an argument to check free blocks on a particular file system or without an argument to check free space on all mounted file systems. For examples, see Figure 8-1. The first line for the file system shows the number of free blocks and inodes, and the second shows total blocks and inode blocks. To get the number of megabytes free, divide by 1,024. To calculate the percentage of free space, divide the number of free blocks by the number of blocks in the file system. For example, if **/dev/usr** has 19,864 blocks and 16,081 are free, approximately 80 percent of the file system is free (16,081/19,864 = .80).

```
$ df -t
/      (/dev/root):       4113 blocks       3613 i-nodes
       ( 7954 total data blocks, 250 total i-node blocks)
/usr   (/dev/usr ):       16081 blocks      9534 i-nodes
       ( 19864 total data blocks, 623 total i-node blocks)

$ df -t /dev/usr
/usr   (/dev/usr ):       16081 blocks      9534 i-nodes
       ( 19864 total data blocks, 623 total i-node blocks)
```

Figure 8-1. Checking Free Blocks with **df -t**

Checking Blocks Used with du

The **du** command reports the number of 1,024-byte blocks used. If you are checking usage for the current directory, no argument is necessary, but if you are checking other directories, give their names as arguments.

You may use three different options with **du.** The **-s** option gives you a sum total for each directory you name, the **-a** option gives a total for each file, and the **-r** option gives you messages if problems occur. Typical problems are that directories cannot be read and files cannot be opened.

Figure 8-2 illustrates how **du** can be used to check blocks used by a particular user.

```
# cd /usr/jack
# du
3     ./letters
2     ./memos
4     ./newsletters/employee
4     ./newsletters/customer
9     ./newsletters
16    .
# du -s
16    .
# du -a
1     ./.profile
1     ./letters/a.taylor
1     ./letters/r.reed
1     ./memos/k.blum
2     ./memos
1     ./newsletters/employee/apr
1     ./newsletters/employee/may
1     ./newsletters/employee/june
4     ./newsletters/employee
1     ./newsletters/customer/apr
1     ./newsletters/customer/may
1     ./newlsetters/customer/june
4     ./newsletters/customer
9     ./newsletters
16    .
```

Figure 8-2. Checking Blocks Used with **du**

Checking Use by Owners

The **quot** command summarizes block use by owner on a file system. If you give a file system as an argument, information is limited to that file system. Otherwise, information is given for all mounted file systems.

You may use several options with **quot**. The **-f** option gives the number of files (in the second column) as well as blocks used. For other options, see **quot** in the *XENIX 286 Reference Manual.*

For simplicity, Figure 8-3 illustrates the use of **quot** to check use by owners of files on the root file system. In practice, you will probably be more concerned with use on the user file system (**/dev/usr**), if you have separate file systems.

```
# quot /dev/root
/dev/root:
  3215      bin
   588      root
    67      sysinfo
# quot -f /dev/root
/dev/root:
  3215      292     bin
   588       80     root
    67        9     sysinfo
```

Figure 8-3. Checking Use by Owners with **quot**

Freeing Up Space

For efficiency and for clarity, it is advisable to keep only necessary files on the system.

Keeping Only Necessary Files

When space is getting low, you should remind users to delete unnecessary files. System administrators often put a reminder in **/etc/motd** to remind users of the situation when they log on.

Removing Core Files and Temporary Files

Program errors can cause extra files to be created or left in the file system. Normally these files are not necessary and should be removed.

When a program causes or encounters an error it cannot recover from, the program is terminated, and a file called **core** is created with a copy of the data used by the program.

You could use this command to locate all **core** files in the **/usr** directory that have not been used for a week:

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

You could use this command to locate and remove all of the core files in the **/usr** directory that have not been used for a week:

```
# find /usr -name core -atime +7 -exec rm {} \;
```

Another type of file is a temporary file that is created by a program to accomplish some intermediate task, then is not removed because the program has an error or is interrupted. Its name depends on the program that created it, and it is usually placed in the **/tmp** or **/usr/tmp** directory. The **/etc/rc** file normally has a command to clear these directories each time the system starts up, but if the system runs continuously, remove old files from the **/tmp** and **/usr/tmp** directories occasionally.

Clearing Log Files

Records of system activity are kept in several files that are referred to as log files. Some of these files can become very large, so it is wise to check them and clear them at reasonable intervals. These are the log files that should be monitored:

/usr/adm/pacct Accounting data is recorded in the file used as an argument with **accton.** By convention, this file is **/usr/adm/pacct.** It grows rapidly and should be cleared regularly.

/usr/adm/messages Error messages generated by the system are recorded in **/usr/adm/messages** (which is created automatically if needed) since **/etc/dmesg** is in **/usr/lib/crontab** and is executed regularly by **cron.**

/usr/adm/wtmp User logins and logouts, startups, and shutdowns are recorded in **/usr/adm/wtmp.**

/usr/adm/sulog Each use of the **su** command is recorded in **/usr/adm/sulog** if **/etc/default/su** has the option set.

/usr/lib/cronlog If the **CRONLOG** variable in **/etc/default/cron** is set to YES, **cron**'s actions are recorded in **/usr/lib/cronlog.**

/usr/spool/at/past Each use of **at** is recorded in **/usr/spool/at/past.**

/usr/spool/micnet/*/LOG The transmissions between machines in a Micnet network are recorded in **/usr/spool/micnet/*/LOG.** The "*" will be the name of a remote machine.

/usr/spool/uucp/LOG The transmissions between machines in a **uucp** network are recorded in **/usr/spool/uucp/LOG.**

/usr/spool/uucp/SYSLOG The information in **/usr/spool/uucp/LOG** is condensed and added to **/usr/spool/uucp/SYSLOG** when you execute **uulog.**

To clear a log file, redirect the **/dev/null** file to it. Since **dev/null** is empty, this overwrites the log file with an empty file. For example, you would use this command to clear the **/usr/adm/pacct** file:

```
# cp /dev/null /usr/adm/pacct
```

Log files can be cleared automatically, if the proper commands are placed in the **/usr/lib/crontab** file. See instructions for running processes automatically in Chapter 10.

Moving Files to Another File System

If users have large files that they rarely use, you may want to move them to a separate file system on a flexible disk and mount the file system only when it is needed.

Packing Large Files

Text files can be reduced to 60-75 percent of their original size with the **pack** command. When they are needed, they can be expanded to their original size with the **unpack** command. Users should pack and unpack their own files, but it may be necessary for you to remind them.

Accounting for Time Used

Some system administrators like to keep track of computer use for billing users or evaluating the system. This tracking is called process accounting, and is entirely optional. You may start and stop it at any time. If you start process accounting, a record for each process is stored in the file that you name as an argument to the **accton** command. By convention, this is **/usr/adm/pacct**. After process accounting has started, use **acctcom** to print reports covering completed processes. This section explains how to start and stop process accounting and how to display accounting data. If you want to make process accounting part of system startup (the typical case), see Chapter 10.

Starting and Stopping Process Accounting

A system administrator usually starts process accounting during system startup (see Chapter 10), but it can be started any time with the **accton** command and a file name. Traditionally, **/usr/adm/pacct** is used, as follows:

```
# /etc/accton /usr/adm/pacct
```

If **/usr/adm/pacct** does not exist, it is created by the command. If it does exist, accounting is turned on so records can be added. The **/usr/adm/pacct** file grows rapidly as it accumulates process accounting details, so check it regularly and clear it often as instructed in "Clearing Log Files" earlier in this chapter.

To turn process accounting off, give the **accton** command without a file name. This keeps new records from being added to **/usr/adm/pacct**, but it does not delete any existing records.

Displaying Accounting Information

The **acctcom** command produces reports of time spent, with each line representing the execution of one process. If the command is invoked without options or arguments, all processes in the **/usr/adm/pacct** are listed in chronological order beginning with the earliest. The command is not privileged, and any user may execute it:

```
# acctcom
```

If you want accounting information from an old file with another name, use the name as an argument. For example, this command would give accounting information from an old file named "/usr/adm/opacct":

```
# acctcom /usr/adm/opacct
```

If you have saved accounting records in several files, you may give more than one file name or use shell metacharacters. For example, if you have "/usr/adm/opacct1", "/usr/adm/opacct2", and "/usr/adm/opacct3", you may use this command to include all of them in the report:

```
# acctcom /usr/adm/opacct*
```

The information given by **acctcom** is described below and illustrated in Figure 8-4.

COMMAND NAME The name of the command.

USER The login name of the user who executed the command.

TTYNAME The terminal associated with the process. A "?" means that the process was not associated with a particular terminal. For example, in Figure 8-4, **cron, atrun,** and **sh** are not associated with a particular terminal because they are executed automatically.

START TIME The time the process began.

END TIME The time the process was completed.

REAL (SECS) The elapsed time, in seconds, from execution to completion.

CPU (SECS) Total time used by the CPU (central processing unit), in seconds. Total CPU time equals system time (time spent on system calls) plus user time (all other time spent on a process by the CPU).

MEAN SIZE(K) The average amount of memory used, in kilobytes, over the duration of the process.

```
# acctcom
COMMAND                     START     END       REAL     CPU      MEAN
NAME      USER  TTYNAME     TIME      TIME      (SECS)   (SECS)   SIZE(K)
#accton   root  ttyc2       14:44:57  14:44:57  0.20     0.14     16.50
cron      root  ?           14:45:00  14.45:00  0.02     0.02      4.00
atrun     root  ?           14:45:00  14:45:00  0.46     0.20     15.50
sh        root  ?           14:45:00  14:45:01  0.82     0.14     13.50
ls        root  ttyc2       14:45:00  14:45:00  0.40     0.20     21.50
```

Figure 8-4. Sample Process Accounting Output

Normally system administrators want to check usage by user, group, date, time, command, or some other factor, such as the hog factor, which is calculated by this formula:

(total CPU time)/(elapsed time) = hog factor

The hog factor may be significant if some processes seem to slow down everyone's work. You can identify processes with high hog factors and give them lower priorities (with **nice**), if necessary. (See Chapter 10.)

Figure 8-5 describes the options for **acctcom**, and Figure 8-6 illustrates some sample commands.

-b Read backwards, showing the most recent processes first.

-f Include the fork/exec flag and system exit status.

-h Show the hog factor instead of mean memory size. Processes with a high hog factor use a large percentage of system resources and are scheduled after processes with lower hog factors.

-i Include the input/output counts.

-k Show total kilocore minutes instead of mean memory size. A kilocore minute is 1,024 bytes of memory used for 60 seconds. This information is important if memory use is the basis for billing.

-m Show the mean memory size (the default).

-r Include the CPU factor. The formula is

user time/(system time + user time) = CPU factor

-t Show system CPU time and user CPU time separately.

-v Exclude column headings.

-l ***terminal*** List only processes associated with a particular terminal. For example:

```
acctcom -l /dev/ttyc2
```

-u ***user*** List processes belonging to a particular user. You may give the user's login name or ID. For example:

```
acctcom -u kay
acctcom -u 204
```

If you want to see processes for root, you may give # as the user. If you want to see processes associated with unknown users, give ? as the user.

-g ***group*** List the processes for a particular group. You may give the group name or ID. For example:

```
acctcom -g sales
acctcom -g 52
```

Figure 8-5. Process Accounting Options

-d *mm/dd*

List processes for a particular day. Give the month and the day. For example, this command lists processes for June 12:

```
acctcom -d 06/12
```

-s *time*

List processes executed after a particular time. The hour (in military time) is required but minutes and seconds are optional. The first example below lists processes executed after 9:00 a.m., and the second lists processes executed after 9:15 a.m.

```
acctcom -s 09
acctcom -s 09:15
```

-e *time*

List processes executed before a particular time. The hour (in military time) is required but minutes and seconds are optional. For example, this lists processes executed before 2:00 p.m.:

```
acctcom -e 14
```

-n *command*

List processes for a particular command. For example, this lists processes for the units command:

```
acctcom -n units
```

-H *factor*

List processes whose hog factor exceeds the specified factor. For example, this lists processes whose hog factor exceeds .50:

```
acctcom -H .50
```

The list does not give the hog factor itself.

-O *secs*

List processes whose system CPU time exceeds the specified number of seconds. For example, this lists processes whose total CPU time exceeds 2.00 seconds:

```
acctcom -O 2.00
```

-C *secs*

List processes whose total CPU time (system time plus user time) exceeds the specified number of seconds of CPU use. For example, this lists processes whose total CPU time exceeds 20 seconds:

```
acctcom -C 20.00
```

Figure 8-5. Process Accounting Options (Continued)

You may combine options on one command line, as illustrated in Figure 8-6.

```
# acctcom -u kay -s 8:41 -d 9/11
```

List the processes that login "kay" executed after 8:41 a.m. on September 11.

```
# acctcom -u # -d 10/20 -b
```

List the processes for **root** on October 20, starting with the most recent processes.

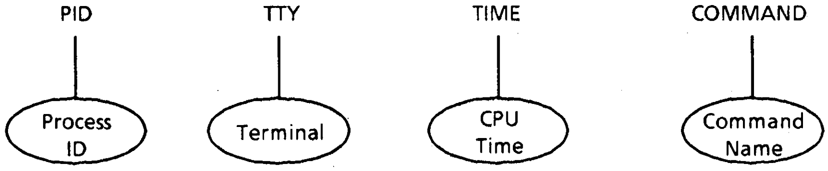
```
# acctcom -g 53 -n vi
```

List group 53's use of **vi**.

Figure 8-6. Sample Process Accounting Commands

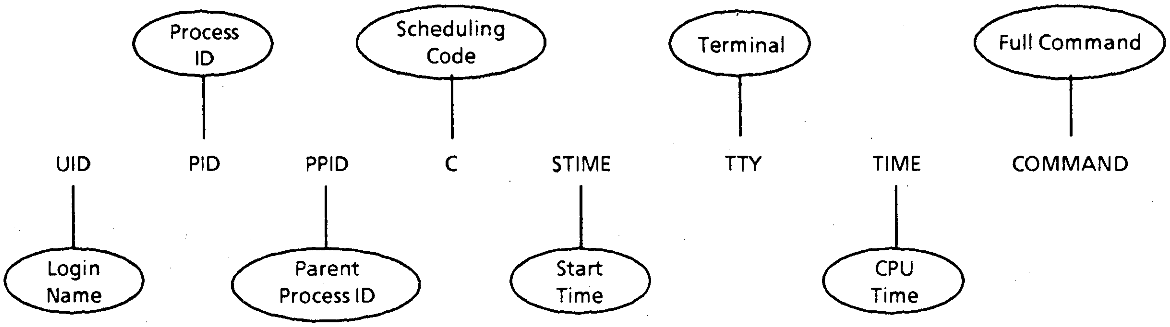
Checking Processes

The **ps** command lists the processes that are running, and can be invoked with options for brief form (see Figure 8-7), full form (see Figure 8-8), or long form (see Figure 8-9). Normally the **-e** (everyone) option is used so all processes are included. Without this option, only the processes of the terminal where you are logged on are included. The **-f** option lists users by their login names instead of their UIDs and is useful if you do not know users by their terminal names or UIDs. The **-f** option also gives full commands. For additional options, see **ps** in the *XENIX 286 Reference Manual.* For sample reports, see Figures 8-10, 8-11, and 8-12. Notice that you do not have to be **root** to check processes.



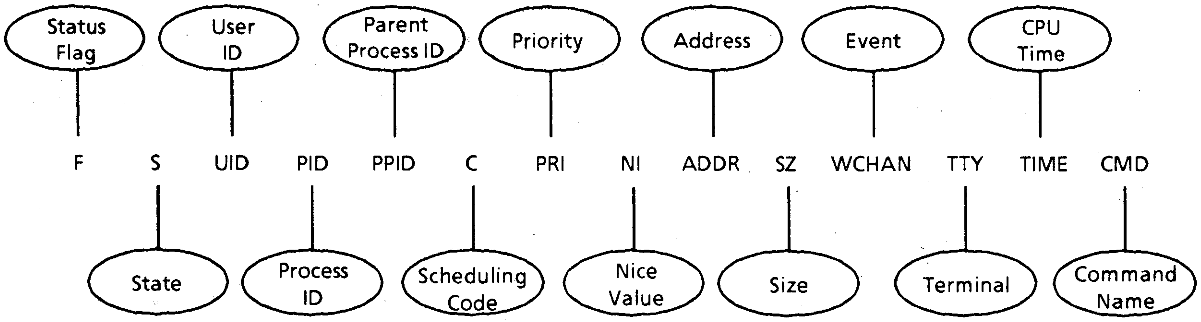
F-0327

Figure 8-7. Fields with the Brief Status Report (**ps**)



F-0328

Figure 8-8. Fields with the Full Status Report (**ps -ef**)



F-0329

Figure 8-9. Fields with the Long Status Report (**ps -el**)

The fields are defined as follows:

F

This represents the values of status flags associated with a process. The flags have these values, which are added to produce F:

01	in core
02	system process
04	locked in core (for physical input/output, for example)
10	being swapped
20	being traced by another process

S

This represents the state of the process:

0	nonexistent
S	sleeping
W	waiting
R	running
I	intermediate
Z	terminated
T	stopped

UID

The user ID of the owner of the process.

PID

The process ID. This number is used with the **kill** command to kill a process.

PPID

The process ID of the parent process.

C

Processor utilization for scheduling.

STIME

The starting time of the process. This is shown only with the full report (**ps -f**).

PRI

The current priority of the process. Higher numbers mean lower priority, and priorities change as new processes are initiated and old ones are completed.

NI

The initial **nice** value used in computing the priority. See the discussion of changing priorities later in this chapter.

ADDR

The memory address of the process (if resident) or the disk address.

SZ

The size in 1,024-byte blocks of the core image of the process.

WCHAN

The event for which the process is waiting or sleeping. If the column is blank, the process is running.

TTY

The terminal associated with the process.

TIME

The cumulative CPU time for the process.

CMD

The command name. If you print the full report (**ps -f**), the full command line, including options and arguments, is shown. For **getty**, the command name is sometimes left blank. The login shell is marked with a dash (**-sh** or **-csh**) in the full report only.

```
$ ps -e
   PID        TTY      TIME        COMMAND
     0          ?      0:02        swapper
     1          ?      0:06        init
   361          ?      0:00        getty
    47         f0      0:00        getty
    23          ?      0:19        update
    32          ?      1:36        cron
  2319         c0      0:00        getty
   506         c1      0:00        getty
  1289         c2      0:04        sh
    51         c3      0:00        getty
  2418         c2      0:04        ps
$
```

Figure 8-10. Brief Process Status Report

```
$ ps -ef
   UID    PID   PPID    C      STIME     TTY     TIME     COMMAMD
  root      0      0   16          ?       ?     0:02     swapper
  root      1      0    0          ?       ?     0:06     /etc/init
  root    361      1    0     Sep 11      co     0:00
  root     47      1    0     Sep 11      f0     0:00
  root     23      1    0     Sep 11       ?     0:19     /etc/update
  root     32      1    0     Sep 11       ?     1:36     /etc/cron
  root   2319      1    0   11:08:39      c0     0:00
  root    506      1    0     Sep 11      c1     0:00
  jack   1289      1    0   18:12:23      c2     0:04     -sh
  root     51      1    0     Sep 11      c3     0:00     [sh scrip]
 sarah   2418   1289   74   15:05:56      c2     0:04     ps
$
```

Figure 8-11. Full Process Status Report

```
$ ps -el
 F   S   UID    PID  PPID   C   PRI   NI   ADDR   SZ  WCHAN  TTY  TIME   CMD
 3   S     0      0     0  16     0   20    lad    0   67fd    ?  0:02   swapper
 1   S     0      1     0   0    30   20    204    4   dc6e    ?  0:06   init
 0   S     0    361     1   0    28   20     99    3   9fb9   co  0:00   getty
 0   S     0     47     1   0    28   20     ee    3   9f81   f0  0:00   getty
 1   S     0     23     1   0    40   20    20b    6           ?  0:19   update
 1   S     0     32     1   0    40   20    23f    4           ?  1:36   cron
 1   S     0   2319     1   0    28   20    1c3    3   67fe   c0  0:00   getty
 0   S     0    506     1   0    28   20     ae    3   6836   c1  0:00   getty
 1   S   202   1289     1   0    30   20    1ca   10   dd6A   C2  0:04   sh
 0   S     0     51     1   0    28   20    103    3   68a6   c3  0:00   getty
 1   R   200   2418  1289  74    87   20    212   20          c2  0:04   ps
$
```

Figure 8-12. Long Process Status Report

Changing Execution Priorities

Priorities are calculated by the system, but they can be influenced at execution time to some extent by **nice** values. These are values, from 0-39, that are set with the **nice** command. A **nice** value of 0 increases the priority as much as possible, and a **nice** value of 39 reduces the priority as much as possible.

Every process automatically begins with a **nice** value of 20. Any user can reduce the priority of a process by giving a positive **nice** value of 1-19, but only **root** can increase the priority by giving a negative **nice** value of -1 to -20. Figure 8-13 illustrates how priorities are reduced and increased with **nice**. Chapter 10 suggests how to tailor the environment so certain processes always have higher or lower priorities.

$ nice nroff Any user can invoke **nice** to reduce the priority of a command at execution time. If **nice** is given without a value, the default value of 10 is used, and the new **nice** value is 30. In this example, the user is changing the **nice** value to 30 so **nroff** will run with a lower priority.

$ nice -5 sort Any user can give a positive **nice** value of 1-19 to reduce the priority of a command. In this example, the user is changing the **nice** value to 25 (the automatic value plus 5) to reduce the priority of the **sort** command. The **-** is not used here as a minus sign. It just means the number is an option.

# nice --15 vi Only **root** can reduce the **nice** value, and thus increase the priority of a process at execution time. In this example, **root** is reducing the **nice** value by 15, which results in a **nice** value of 5 for the **vi** command.

Figure 8-13. Changing Execution Priorities

Notice that the long process status report in Figure 8-12 shows both the **nice** value and the priority of the process. The **nice** value is set at execution time and cannot be changed. The priority is frequently recalculated by the system, and the number given is the priority at one particular moment.

intel®

CHAPTER 9
SOLVING SYSTEM PROBLEMS

Solving Terminal Problems

Information about terminals is kept in several important files. The **/etc/ttys** file has an enabled/disabled code, a code that identifies terminal characteristics, and the terminal name. The **/etc/ttytype** file has the terminal type (such as tvi950 for the Televideo 950 model) and name. The **.profile** or **.login** file for each user should include the **TERM** variable. Many problems can be avoided by including the appropriate **tset** command in **.profile** and **.login.** (See Chapter 10.)

If a terminal does not function properly, the problem may be that the terminal line characteristics (tty settings) that the system maintains are incorrect or a program has halted because of an error. Typical symptoms are that the terminal ignores all keystrokes or strange or jumbled characters appear. Solutions for these problems are described in this section.

Restoring a Nonechoing Terminal

If a terminal does not display the characters typed on the keyboard, it is likely that a program has stopped because of an error. To correct the problem, go to the terminal and type the following, ignoring error messages that may appear when you press CONTROL-J:

```
CONTROL-J
tset
CONTROL-J
```

If this does not work, go to another terminal and kill the process running on the nonechoing terminal.

Restoring Columns

If the output on a screen seems to be jumbled, it may be because the terminal characteristics do not include tabs. Use this command at the terminal:

```
$ stty -tabs
```

Erasing Deleted Characters

If pressing BACKSPACE to erase a typing error makes the cursor back over the character but does not erase it, change the terminal characteristics by adding this command to the user's **.profile** (for Bourne shell) or **.login** (for C shell):

```
stty echoe
```

Fixing an Unresponsive Terminal

If all input to a terminal is ignored, try these solutions, in the order listed, until the problem goes away:

- Make sure the terminal is plugged in.

- Release the SCROLL (or NO SCROLL) key, if it is on, since this key stops and starts all movement on the screen. On some terminals, CONTROL-Q and CONTROL-S are used instead of SCROLL/NO SCROLL.

- Reset the terminal by pressing its RESET switch or by turning the terminal off and on again.

- If the terminal still ignores all input, some error has probably caused a process to hang up, so log on at another terminal, use **ps -e** to find the PID of the process running on the unresponsive terminal, and use **kill** with the PID to kill the process. For example, you would use this command to kill a process with PID 140:

  ```
  # kill 140
  ```

  If the process does not stop in a few moments, use **kill -9** with the PID to kill it. The **-9** option causes the process to stop, but it may cause the terminal to stop echoing what is typed. If this happens, see "Restoring a Nonechoing Terminal" earlier in this chapter. The **-9** option may also leave temporary files in the directory, so check the directory and remove any unnecessary temporary files.

Solving Printer Problems

Occasionally, users start the wrong print job by mistake, or a printer produces garbage characters or refuses to print. This section gives suggestions for responding.

Removing Jobs from the Print Queue

To remove a job from the print queue, first invoke **lpq** to find the job number, then invoke **lprm** with the job number as an argument.

Removing Garbage Characters

If a printer is printing garbage characters, reset the printer by pressing its RESET switch or by turning the printer off and on again. If garbage was in the buffers, this may clear it.

Restarting a Line Printer

After users have added files to the print queue, the line printer daemon (**lpd**) prints files, one by one, from the queue. If line printer errors occur--the printer runs out of paper, for instance--the queue may be locked so nothing is printed. Fix the printer problem, then log on as **root** and give the **/etc/lpdrestart** command. This command kills the process started by **lpd**, removes the lock from the print queue, and starts **lpd** again.

Solving File System Problems

When everything is going smoothly, the XENIX operating system maintains consistent records of the blocks, files, and inodes that make up directory hierarchies. Each block is accounted for, and each file has a path to the root directory. When these records are damaged, the file system is considered corrupt and may suffer from problems such as these:

- A file may not have a full path name to the root directory, so you cannot get to it. This kind of file is called an orphaned file.
- File names may reference nonexistent files.
- A block may be allocated to a file, but also be considered free. Since it appears free, it may be allocated to another file and the contents of the two files can become mixed.

Errors occur when a disk is damaged or when records are not updated with information in buffers because of a hardware failure, a power surge, or a system reset without the use of the **sync** command.

It is important for you to repair a damaged file system immediately. If people continue to work on a damaged file system, the problem gets worse.

Symptoms of Corrupt File Systems

A file system may be corrupted if it is unclean, and it is unclean whenever you do not shut the system down properly. You are warned at system startup if a system needs to be cleaned. Other symptoms of corruption are missing data, garbage characters in directories, programs that do not execute properly, and error messages such as "Cannot open ..." or "Read error in ..." when you use the **pwd** command.

Repairing File Systems

If file systems are corrupted, they must be cleaned with **fsck**. In the process, files are salvaged if possible, but some may be so badly corrupted that they have to be destroyed.

If the computer has stopped, restart it, and type "y" and press RETURN if prompted to clean the file system. This causes **fsck** to run automatically.

If the computer is running but you have reason to suspect that a file system is corrupt, unmount the mounted file system to be checked (do not try to unmount the root file system), then invoke **fsck**. During **fsck**, the screen displays information about the phases of checking and may prompt for actions. Unless you are a very knowledgeable XENIX user who wants to try something unusual, answer "y" to the yes-or-no questions. If any of these messages appears, press RESET to restart the system:

```
***** BOOT XENIX *****
**Normal System Shutdown**
REBOOT XENIX (DON'T SYNC!)
```

Responding to Root File System Out of Space

If the root file system has insufficient work space, an "out of space" message is displayed. You must delete one or more files to restore system operation. Delete only unnecessary files, such as old temporary files in the **/tmp** directory. Do not delete any of the files that were created during installation.

Recovering from a System Crash

A system crash is rare but memorable. You know that one has occurred if a message beginning with "panic" appears on the terminal console or all terminals ignore any keystroke (including the INTERRUPT key and QUIT key).

Most crashes occur because of hardware errors or damage to a file system that the operating system cannot ignore.

If an error message was displayed on the system console, try to find what caused the problem and fix it. The *XENIX 286 Reference Manual* has a list of error messages.

Reset the system by pushing RESET. Try this even if you were unable to pinpoint the cause of the crash.

If the system will not restart, or crashes each time it is restarted, call your field service representative.

If you cannot start the system with the boot disk in the distribution set for installation, the computer has a serious hardware malfunction. Call a hardware service representative for help.

Restoring an Inoperable System

On rare occasions, the system may fail to operate because critical XENIX system files have been modified, removed by mistake, or destroyed by a crash. The emphasis here is on prevention, because restoring a system is a time-consuming process. However, if the problem occurs, reinstall the XENIX system, then restore user programs and data files from backups. See installation instructions in the *XENIX 286 Installation and Configuration Guide*. See instructions for restoring from backups in Chapter 7 of this manual.

Improving Response Time

Several different circumstances can reduce the system's response time, including these:

- Too many processes with high hog factors. If users are running too many processes with high hog factors, see suggestions in Chapter 10.

- Incorrect terminal characteristics. If your text editors appear to be slow, use **stty** to check them. The correct formfeed (ff), newline (nl), and carriage return (cr) setting is usually 0. If it should be reset, add this command line to the user's **.profile** or **.login** file:

  ```
  stty ff0 nl0 cr0
  ```

- Too much distance between the system and terminals. If the cable between the system and a terminal is more than 50 feet long, response time for the entire system may be slow because so much time is spent sending false signals between the system and the terminal. RS-232 cables are not meant to be this long. You may be able to get around this by leaving the terminal on and connected, disabling the terminal port, or installing limited-distance modems or line boosters.

Responding to Panic Messages

Panic messages warn you of serious problems, and they are described in the *XENIX 286 Reference Manual.*

intel®

CHAPTER 10 TAILORING THE ENVIRONMENT

Changing System Startup

As you work with your system, you may want to modify system startup by editing the **/etc/rc** file that is executed whenever the system comes up in multiuser mode. For example, if you you create new file systems, you may want to include commands to mount them automatically. To check the contents of the **/etc/rc** file, use this command:

```
$ more /etc/rc
```

The file has both commands and comments that explain those commands. The comments are lines that begin with #.

If you want to change **/etc/rc**, log on as **root** and use a text editor. Examine the file and insert any new commands in the correct place, since commands will be executed in sequence. Also, read the new commands carefully after typing them because incorrect commands can cause the startup to fail.

Do not remove commands from the file unless you know what they do and are certain they are unnecessary.

Automating Process Accounting

Process accounting can be started and stopped manually, as explained in Chapter 8, or it can be automated by including the appropriate commands in **/etc/rc**. You may find that **/etc/rc** already includes these commands as comments (marked with #):

```
# mv /usr/adm/pacct /usr/adm/opacct
# cp /dev/null /usr/adm/pacct; chmod 644 /usr/adm/pacct
# /etc/accton /usr/adm/pacct
```

If you want to automate process accounting, use a text editor to remove the comment marks (#). The result is that each time the system starts, the existing process accounting file is renamed **/usr/adm/opacct**, a new **/usr/adm/pacct** is created and given the correct permissions, and process accounting is turned on.

Changing the Default .profile File

When you use **mkuser** to add a user with a Bourne shell, a **.profile** file is created in the user's login directory. The original contents come from **/usr/lib/mkuser/mkuser.prof,** which **root** can change with a text editor. For example, these lines should be added to **/usr/lib/mkuser/mkuser.prof** so a user can have correct terminal characteristics at any terminal:

```
TERM = `tset - -r`
export TERM
```

Changing the Default .login File

When you use **mkuser** to add a user with a C shell, a **.login** file is created in the user's login directory. The original contents come from **/usr/lib/mkuser/mkuser.login,** which **root** can change with a text editor. For example, this line should be added to **/usr/lib/mkuser/mkuser.login** so a user can have correct terminal characteristics at any terminal:

```
setenv TERM `tset - -r`
```

Changing the Default .cshrc File

When you use **mkuser** to add a user with a C shell, a **.cshrc** file is created in the user's login directory. The original contents come from **/usr/lib/mkuser/mkuser.cshrc,** which **root** can change with a text editor.

Changing the Default mkuser.mail File

When you use **mkuser** to add a user, a message is sent to the user's mailbox in the **/usr/spool/mail** directory. The message is taken from the **/usr/lib/mkuser/mkuser.mail** file, which **root** can change with a text editor.

Changing the /usr/lib/mail/mailrc File

The **/usr/lib/mail/mailrc** file is executed each time **mail** is invoked, and it is executed before the user's **.mailrc,** if that file exists. Administrators often include these commands in **/usr/lib/mail/mailrc** (or place them in users' **.mailrc** files as described in "Steps" in Chapter 3):

```
set autombox
set page = 22
set dot
```

Changing the /etc/profile File

The **/etc/profile** is executed before a user's **.profile** each time a user logs on. This file can be edited by **root,** and it usually includes these commands:

```
umask 022
stty echoe
```

See "Default Permissions" in Chapter 4 for an explanation of **umask.** Include **stty echoe** so pressing BACKSPACE over a character actually erases it from the screen.

Creating System Routing Lists for Mail

Often users send mail to a group of users and like to define routing lists, called aliases. (See the *XENIX 286 User's Guide.*) You can save them some time by defining standard lists for system-wide use. To do this follow these steps:

1. Go to the **/usr/lib/mail** directory.

2. Create the **aliases** file, if it does not already exist.

3. Add aliases to **/usr/lib/mail/aliases** in the format shown in Figure 10-1. Notice that the alias name is followed by a colon and the names of the people on the routing list, separated by commas. Just use a text editor such as **ed** or **vi.**

4. Execute **aliashash.** This command is in the **/usr/lib/mail** directory.

After the system-wide routing list has been created, a user can use the alias with **mail.** See Figure 10-2 for an example. Each user defined in the "sales" list in **/usr/lib/mail/aliases** will receive the message.

```
admin:fred,kay,sam
sales:jane,kay
all:sarah,mary,jack,jane,kay,sam,fred
```

Figure 10-1. Sample Routing Lists for Mail

```
$ mail sales
Subject: Monthly Reports
Monthly reports are due Friday.
(end of message)
$
```

Figure 10-2. Using Routing Lists for Mail

Creating Help Files for Users

Some administrators create text files of useful information for users--such as the backup schedule or the most common commands and options. If you create files for users, be sure the files give them read permission.

Reducing Priorities

Any user can reduce the priority of a process by executing it with the **nice** command. Sometimes, however, you may want to reduce the priority of certain processes whenever they are run. You can do this by creating a shell script. For example, if you wanted to add 5 to the **nice** value of **nroff,** you could give it a new name with the **mv** command, create a new **nroff** command as a shell script that reduces the **nice** value, and change the permissions so everyone can read and execute the file, as illustrated in Figure 10-3. Whenever anyone invoked **nroff,** the shell script with the new **nice** value for **nroff** would be executed.

```
# cd /bin
# mv nroff nnroff
# ed nroff
0
*a
nice -5 nnroff $*
.
*w
14
*q
# chmod a + rx nroff
```

Figure 10-3. Changing Command Priority

Keeping Programs in Swap Space

If your system has enough swap space, you can make a slight improvement in the response time for heavily-used programs by setting a bit, called the sticky bit, with the **chmod +t** command so the program is never removed from the swap area. For example, this command sets the sticky bit for **vi:**

```
# chmod  + t /bin/vi
```

You have to be logged on as **root** to use the **+t** option, and you should use it with caution, if at all. Keeping some programs constantly in the swap area is not something you should do casually because it can cause the swap area to become fragmented and cause the system to panic.

Running Processes Automatically

As you become familiar with the system, you may find you are doing some functions on a regular basis and decide to automate them. The **/etc/rc** file typically includes the **/etc/cron** command, which regularly checks the **/usr/lib/crontab** file and executes its commands at the stated time. You can edit **/usr/lib/crontab** to include more commands. The file has lines with six fields each, separated by spaces or tabs, to define

- The minute (0-59)
- The hour (0-23)
- The day of the month (1-31)
- The month (1-12)
- The day of the week (0-6, 0=Sunday)
- The command to be executed

For each of the first five fields, you may give

- A specific number
- An inclusive range (such as 1-7)
- A list of numbers separated by commas (such as 1,8,16)
- An asterisk (*) for all values

Figure 10-4 shows and explains sample entries for **/usr/lib/crontab.**

```
0, 5, 10 ,15 ,18, 20, 25, 30, 35, 40, 45, 50, 55  *  *  *  *    /usr/lib/atrun
```

Execute /usr/lib/atrun at the minutes specified at every hour, every day of the month, every month, every day of the week. This program runs processes requested by at.

```
20      1       *       *       *               usr/bin/calendar  -
```

Execute /usr/bin/calendar the twentieth minute of the first hour of every day, every month, every day of the week. The dash (-) means to send everyone their calendar by mail.

```
0, 10, 20, 30, 40, 50        *  *  *        *            /etc/dmesg  ->>/usr/adm/messages
```

Execute /etc/dmesg, which prints system messages, and add the output to /usr/adm/messages every 10 minutes of every hour, every day of the month, every month, every day of the week.

Figure 10-4. Sample **/usr/lib/crontab** Entries

Logging Processes Run by cron

If you would like to keep a log of all of the commands executed automatically by **cron,** use a text editor to set the **CRONLOG** variable in **/etc/default/cron** to yes, as illustrated below:

```
CRONLOG = YES
```

The processes will be recorded in **/usr/lib/cronlog.** If you keep this automatic log, clear it occasionally. (See Chapter 8.)

Logging Use of su

A person who is seeking unauthorized power over the system may use the **su** (switch user ID) command, so you should be aware of uses of the command, both successful and unsuccessful. If you set the **SULOG** and **CONSOLE** variables in the **/etc/default/su** file, you can be notified via your terminal screen each time the command is invoked successfully, and can keep a log of all attempted uses.

Set the **SULOG** variable to **/usr/adm/sulog,** which is the name of the log file, and set **CONSOLE** to the terminal where notification should be displayed. Figure 10-5 shows sample entries in **/etc/default/su,** and Figure 10-6 shows sample entries in **/usr/adm/sulog.** A minus sign (-) means an unsuccessful use of **su** and a plus sign (+) means a successful use. If the variables in **/etc/default/su** are stored with a #, they are comments, and you can change them to commands by removing # with a text editor such as **ed** or **vi.** If you keep the automatic log for **su,** clear it occasionally. (See Chapter 8.)

```
SULOG = /usr/adm/sulog
CONSOLE = /dev/console
```

Figure 10-5. Variables for Recording **su** Use

```
SU    10/29   15:34   -   kay-root     Kay seeks root's ID unsuccessfully.
SU    10/29   15:35   -   kay-root     Kay seeks root's ID unsuccessfully.
SU    10/29   15:36   -   kay-root     Kay seeks root's ID unsuccessfully.
SU    10/29   15:37   -   kay-sarah    Kay seeks sarah's ID unsuccessfully.
SU    10/29   16:10   +   sarah-root   Sarah seeks root's ID successfully.
```

Figure 10-6. Sample Log of **su** Use

Adding a Terminal

When you add a communications board to your system, you create nodes for all possible ports. (See the *XENIX 286 Installation and Configuration Guide.*) When you actually add terminals, follow these steps:

1. Run the cable from the port on the system.

2. Log on as **root.**

3. See if the particular terminal is listed in **/etc/termcap.** Most common terminals are listed in this file, but if yours is not, add information about it. See "Files" in the *XENIX 286 Reference Manual* for details.

4. Edit **/etc/ttytype,** if necessary. This file lists the terminal ports with default terminal types. If it lacks an entry for the type of terminal, add it. Check **/etc/termcap** for types, then use a text editor such as **ed** or **vi** to change the type. See Figure 10-7 for a sample **/etc/ttytype** file.

5. Edit **/etc/ttys,** if necessary. This file stores three pieces of information, which, unfortunately, are not separated. These fields are identified below and illustrated in Figure 10-8:

   - Enabled/disabled code. The first character is 0 for a disabled port, or 1 for an enabled port. You can change this code with a text editor or with the **enable** and **disable** commands, but leave the port disabled until the terminal is hooked up.

   - Baud rate code. The second character is a code for the baud rate. The most common codes are 3 for a rate that starts at 1200 and cycles down to 110 baud for modem support, or 6 for 9600 baud. For other codes, see Figure 10-9.

   - Port name. The rest of the line has the port name. The system administrator's console is identified as **console,** and the next port is identified as **ttyf0.** The names of the other ports begin with "tty" and end with a character and number. The character depends on the communications board and is "a" for an iSBC® 544 or 544A Intelligent Communications Controller Board, "b" for an iSBC 534 Four Channel Communications Expansion Board, and "c" or "d" for an iSBC 188/48 Advanced Communicating Computer. The number is the hexadecimal number of the port, 0-9 and a-f.

6. Connect the terminal, following instructions in your hardware documentation.

7. Use the **enable** command to enable the port. Enable the port only after you have made the correct entries in **/etc/ttys** and **/etc/ttytype** and connected the terminal. Wait one full minute between each use of **enable** or **disable,** or the system may crash.

```
# more  /etc/ttytype
h8020e      console
tvi970      ttyf0
tvi970      ttyc0
tvi970      ttyc1
h8020e      ttyc2
tvi970      ttyc3
tvi970      ttyc4
tvi970      ttyc5
tvi970      ttyc6
tvi970      ttyc7
tvi970      ttyc8
tvi970      ttyc9
tvi970      ttyca
tvi970      ttycb
```

Figure 10-7. Sample **/etc/ttytype** File

```
# more  /etc/ttys
16console
06ttyf0
16ttyc0
16ttyc1
16ttyc2
06ttyc3
06ttyc4
06ttyc5
06ttyc6
06ttyc7
06ttyc8
06ttyc9
06ttyca
06ttycb
```

Figure 10-8. Sample **/etc/ttys** File

-	110 baud; intended for an ASR-33 console
0	150 baud; intended for an ASR-37 console
1	Cycles through 300-150-110-1200 baud; useful for slow dial-up lines
2	300-baud console Decwriter
3	Cycles through 1200-300-150-110 baud; recommended for dial-up lines
4	2400 baud
5	4800 baud
6	9600 baud

Figure 10-9. Codes for Baud Rates

Removing a Terminal

When you need to remove a terminal from the system, follow these steps:

1. Log on as **root** at some other terminal.

2. Disable the terminal. For example, this command disables **/dev/ttyc2:**

   ```
   # disable /dev/ttyc2
   ```

   Wait a full minute between each use of **disable** or **enable,** or the system may crash.

3. Turn off the power to the terminal.

4. Disconnect the terminal from the system. The serial line is now free to accept another device.

Administering Assignable Devices

A user working with file systems mounted from flexible disks or other removable devices may want to deny other unprivileged users any access to the files on a device. A user can temporarily assign a device for exclusive use with **assign** if the device is listed in **/etc/atab** (a table of assignable devices) or **/etc/atab** does not exist but the device belongs to a system user named **asg.** When finished with the device, a user can invoke **deassign** to relinquish exclusive access.

The **/etc/atab** file is created, if it does not already exist, when **assign** is invoked. Any time you change the ownership of a device to or from **asg,** delete **/etc/atab** so it will be rebuilt with the correct information the next time someone uses **assign.** Figure 10-10 shows how to make a 5¼-inch flexible disk device assignable and Figure 10-11 shows how to keep a device from being assignable (by making the owner **root** or **sysinfo).**

```
# chown asg /dev/dvf0
# rm -f /etc/atab
```

Figure 10-10. Making a Device Assignable

```
# chown root /dev/dvf0
# rm -f /etc/atab
```

Figure 10-11. Making a Device Unassignable

APPENDIX A
STANDARD DEVICES

Standard Devices

Figure A-1 lists the standard devices with names and descriptions and Figure A-2 gives sizes for the root and user file systems on Winchester disks.

8-INCH SYSTEMS

f0	SS/DD 1024 bps, 8 spt, 1224 1K blocks
syf0	SS/SD 128 bps, 26 spt, 250.25 1K blocks
dxf0	DS/DD 256 bps, 26 spt, 994.5 1K blocks
df0	DS/DD 1024 bps, 8 spt, 608 1K blocks
dboot	DS/DD 256 bps, 26 spt, partial disk, boot disk
dram	DS/DD 256 bps, 26 spt, partial disk, boot disk

5$\frac{1}{4}$-INCH SYSTEMS

dnf0	DS/DD 512 bps, 8 spt, track 0 is SD (iRMX™ 86 format)
df0	DS/DD 1024 bps, 4 spt, 316 1K blocks
syf0	SS/SD 128 bps, 16 spt, 160 1K blocks
dzf0	DS/DD 512 bps, 8 spt, track 0 not SD (MS-DOS 1.0 format), 320 1K blocks
dvf0	DS/DD 512 bps, 9 spt, track 0 not SD (MS-DOS 2.0 format), 360 1K blocks
dboot	DS/DD 1024 bps, 4 spt, partial disk, boot disk
dram	DS/DD 1024 bps, 4 spt, partial disk, boot disk

ALL SYSTEMS - WINCHESTER DISKS

w0	Entire Winchester disk, 1024 bps
w0a	Root partition
w0b	Swap partition
w0c	User partition (does not exist on 310-17)

ALL SYSTEMS - SERIAL DEVICES

ttya[0-f]	iSBC 544 terminal ports
ttyb[0-f]	iSBC 534 terminal ports
tty[c,d][0-f]	iSBC 188/48 (and iSBX™ 354) terminal ports

ALL SYSTEMS - OTHER

lp	line printer parallel port on iSBC 286/10
rmt0	streamer tape drive on iSBX 217C

Key: SS = single-sided, DS = double-sided, SD = single density, DD = double density, bps = bytes per sector, spt = sectors per track, 1K = 1024 bytes

Figure A-1. Standard Devices

Disk Type	Root File System w0a	User File System w0c
10-megabyte Winchester	9522	--
15-megabyte Winchester	14886	--
32-megabyte Winchester (8-inch)	8208	18447
36-megabyte Winchester	8208	23670
62-megabyte Winchester (8-inch)	8208	45183

Figure A-2. File System Sizes for Winchester Disks

APPENDIX B
BACKUP LOGS

Backup Logs

The following pages have sample logs for recording backups.

LOG FOR DAILY
BACKUP WITH SYSADMIN

Date	Time	File System	Comments/Problems

F-0330

LOG FOR PERIODIC BACKUP WITH SYSADMIN

Date	Time	File System	Comments/Problems

F-0332

LOG FOR DAILY BACKUP WITH DUMP

Date	Time	File(s)	5	6	7	8	Comments/Problems

F-0333

LOG FOR WEEKLY
BACKUP WITH DUMP

Date	Time	File(s)	1	2	3	Comments/Problems

F-0334

LOG FOR LEVEL 0
BACKUP WITH DUMP

Date	Time	File System	Comments/Problems

F-0335

APPENDIX C
THE ROOT DIRECTORY

The Root Directory

The root directory (/) is the base of the hierarchical directory structure that XENIX uses to keep track of the files in a file system. The root directory includes these directories and files:

/bin This directory has the XENIX commands that users execute most often.

/boot This file has the code for a program that is needed to start the system.

/dev This directory contains special device files.

/etc This directory has commands that are usually reserved for the system administrator plus files that the system administrator uses.

/lib This directory has libraries of subroutines.

/lost+found This directory lists files and directories whose links to the file system have been destroyed because of some problem. The entries are placed in this directory automatically by the **fsck** command that the system administrator uses regularly to check the integrity of the file system.

/mnt This directory is normally used for file systems that are mounted on the root file system.

/sys This directory has the files necessary to rebuild the XENIX kernel.

/tmp This directory is used for temporary files that are created by programs. These files may be removed during normal operations, and they are usually removed each time the system is started.

/usr This directory is traditionally used for commands, libraries, and all login directories. It is the ancestor of all user files and directories.

/xenix This file has executable code for the XENIX kernel for a hard disk system.

/xenix.f This file has executable code for the XENIX kernel for the flexible disk system.

/xenix.w This file has executable code for the Winchester-based XENIX kernel for a boot disk.

intel®

APPENDIX D RELATED PUBLICATIONS

Related Intel Publications

Copies of the following publications can be ordered from

Literature Department
Intel Corporation
3065 Bowers Avenue
Santa Clara, CA 95051

Overview of the XENIX 286 Operating System, Order Number 174385 -- XENIX history, XENIX uses, basic XENIX concepts, and an overview of other XENIX manuals.

XENIX 286 Installation and Configuration Guide, Order Number 174386 -- how to install XENIX on your hardware and tailor the XENIX configuration to your needs.

XENIX 286 User's Guide, Order Number 174387 -- a tutorial on the most-used parts of XENIX, including terminal conventions, the file system, the screen editor, and the shell.

XENIX 286 Visual Shell User's Guide, Order Number 174388 -- a XENIX command interface ("shell") that replaces the standard command syntax with a menu-driven command interpreter.

XENIX 286 System Administrator's Guide, Order Number 174389 -- how to perform system administrator chores such as adding and removing users, backing up file systems, and troubleshooting system problems.

XENIX 286 Communications Guide, Order Number 174461 -- installing, using, and administering XENIX networking software.

XENIX 286 Reference Manual, Order Number 174390 -- all commands in the XENIX 286 Basic System.

XENIX 286 Programmer's Guide, Order Number 174391 -- XENIX 286 Extended System commands used for developing and maintaining programs.

XENIX 286 C Library Guide, Order Number 174542 -- standard subroutines used in programming with XENIX 286, including all system calls.

XENIX 286 Device Driver Guide, Order Number 174393 -- how to write device drivers for XENIX 286 and add them to your system.

XENIX 286 Text Formatting Guide, Order Number 174541 -- XENIX 286 Extended System commands used for text formatting.

Suggested Readings

The popularity of XENIX and other UNIX-like operating systems has caused many new books to appear in the bookstores. You may want to supplement the XENIX documentation with one or more of these books:

- Banahan, Mike, and Andy Rutter. *The UNIX Book*. New York: John Wiley & Sons, Inc., 1983.

- Bourne, S. R. *The UNIX System*. Reading, Mass.: Addison-Wesley Publishing Company, 1982.

- Christian, Kaare. *The UNIX Operating System*. New York: John Wiley & Sons, Inc. 1983.

- Groff, James R., and Paul N. Weinberg. *Understanding UNIX: A Conceptual Guide*. Indianapolis, Indiana: Que Corporation, 1983.

- Kernighan, Brian W., and Rob Pike. *The UNIX Programming Environment*. Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1984.

- Kernighan, Brian W., and Dennis M. Ritchie. *The C Programming Language*. Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1978.

- McGilton, Henry, and Rachel Morgan. *Introducing the UNIX System*. New York: McGraw-Hill Book Company, 1983.

- Sobell, Mark G. *A Practical Guide to the UNIX System*. Menlo Park, California: The Benjamin/Cummings Publishing Company, Inc., 1984.

- Thomas, Rebecca, and Jean Yates. *A User Guide to the UNIX System*. Berkeley, Calif.: OSBORNE/McGraw-Hill, 1982.

- Yates, Jean, and Sandra L. Emerson. *The Business Guide to the UNIX System*. Reading, Mass.: Addison-Wesley Publishing Company, 1984.

INDEX

intel®

XENIX 286
System Administrator's Guide
174389-001

REQUEST FOR READER'S COMMENTS

Intel's Technical Publications Departments attempt to provide publications that meet the needs of all Intel product users. This form lets you participate directly in the publication process. Your comments will help us correct and improve our publications. Please take a few minutes to respond.

Please restrict your comments to the usability, accuracy, organization, and completeness of this publication. If you have any comments on the product that this publication describes, please contact your Intel representative. If you wish to order publications, contact the Literature Department (see page ii of this manual).

1. Please describe any errors you found in this publication (include page number).

____

____

____

____

2. Does this publication cover the information you expected or required? Please make suggestions for improvement.

____

____

____

____

3. Is this the right type of publication for your needs? Is it at the right level? What other types of publications are needed?

____

____

____

____

4. Did you have any difficulty understanding descriptions or wording? Where?

____

____

____

5. Please rate this publication on a scale of 1 to 5 (5 being the best rating). ____________

NAME ______________________________ DATE ____________

TITLE ______________________________

COMPANY NAME/DEPARTMENT ______________________________

ADDRESS ______________________________

CITY ______________ STATE ______________ ZIP CODE ____________
(COUNTRY)

Please check here if you require a written reply □

WE'D LIKE YOUR COMMENTS ...

This document is one of a series describing Intel products. Your comments on the back of this form will help us produce better manuals. Each reply will be carefully reviewed by the responsible person. All comments and suggestions become the property of Intel Corporation.

NO POSTAGE
NECESSARY
IF MAILED
IN U.S.A.

Intel Corporation
5200 N.E. Elam Young Parkway.
Hillsboro, Oregon 97123

ISO-N TECHNICAL PUBLICATIONS

INTEL CORPORATION, 3065 Bowers Avenue, Santa Clara, California 95051 (408) **987-8080**

Printed in U.S.A.

SOFTWARE