UNIX® System V/386 Release 3.2

SYSTEM ADMINISTRATOR'S GUIDE

AT&T

UNIX® System V/386

Release 3.2

SYSTEM ADMINISTRATOR'S GUIDE

PRENTICE HALL

UNIX® System V/386 Release 3.2

System Administrator's Guide

Prentice Hall, Englewood Cliffs, New Jersey 07632

Library of Congress Catalog Card Number: 88-62531

Editorial/production supervision: Karen Skrable Fortgang
Manufacturing buyer: Mary Ann Gloriande

Published by Prentice-Hall, Inc.
A Division of Simon & Schuster
Englewood Cliffs, New Jersey 07632

NOTICE
The information in this document is subject to change without notice.
AT&T assumes no responsibility for any errors that may appear in
this document.

DIMENSION is a registered trademark of AT&T.
Develcon is a trademark of Develcon Electronics, Inc.
ETHERNET is a trademark of Xerox Corporation.
Intel is a registered trademark of Intel Corporation.
MS-DOS and XENIX are registered trademarks of Microsoft Corporation.
Smartmodem is a trademark of Hayes Microcomputer Products Inc.
UNIX is a registered trademark of AT&T.

The publisher offers discounts on this book when ordered
in bulk quantities. For more information, write or call:

Special Sales
Prentice-Hall, Inc.
College Technical and Reference Division
Englewood Cliffs, NJ 07632
(201) 592-2498

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN 0-13-944893-4

Prentice-Hall International (UK) Limited, *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Prentice-Hall Canada Inc., *Toronto*
Prentice-Hall Hispanoamericana, S.A., *Mexico*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Simon & Schuster Asia Pte. Ltd., *Singapore*
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*

Table of Contents

List of Figures

Chapter 1: Introduction

Purpose of This Guide

This *Guide* provides information needed to install and configure the UNIX System V/386, Release 3.2 operating system and to successfully administer the system. It provides specific instructions for

- installing the Base System
- installing the optional Foundation Set software packages
- installing software applications (including applications for the XENIX System)
- changing the date and time
- adding, changing, displaying, and deleting user logins
- changing user passwords
- reporting system information
- formatting floppies and copying floppy diskettes
- creating, mounting, and unmounting UNIX System file systems
- setting up the computer to support parallel and serial devices
- setting up the electronic mail interface
- shutting down the system prior to turning off the power or rebooting
- backing up data to floppies or cartridge tape
- restoring data from floppies or cartridge tape
- determining the status of printers and their queues
- setting up Basic Networking

Contents of This Guide

The material in this *Guide* is organized into the following chapters and appendixes:

- Chapter 1, *Introduction*, provides information necessary to understand the purpose and organization of this *Guide*.
- Chapter 2, *Software Installation*, tells you how to install the UNIX System V/386 Operating System, Release 3.2, Version 1 Foundation Set on your system. In addition, this chapter tells you how to display software packages that are installed and how to remove add-on packages (if desired).
- Chapter 3, *Using the UNIX System Shell*, describes the basics of the UNIX System shell, such as entering commands, correcting mistakes, printing a file, or running a program in the background, and gives a brief explanation of what manual pages are and how they are referenced.
- Chapter 4, *System Administration*, describes all the administrative tasks you can perform on your computer.
- Chapter 5, *Customizing Your Computer*, describes how you can tailor the environment and system security on your computer.
- Chapter 6, *All About File Systems*, describes what file systems are and describes procedures for creating, checking, repairing, and using file systems.
- Chapter 7, *LP Print Service Administration*, describes how to administer LP Services.
- Chapter 8, *Basic Networking Administration*, describes how to administer Basic Networking.
- Chapter 9, *Remote File Sharing Administration*, describes how to set up and administer the optional remote file sharing feature.
- Appendix A, *Changing Other User's Passwords*, contains a procedure to change passwords for other logins.
- Appendix B, *Adding Basic Networking*, contains the procedures to network your computer with other computers such as how to physically connect direct links and modems, set up modems, and initialize modems. Some recommended switch settings are also provided.

- Appendix C, *fsck Error Messages,* contains a description of the error messages given using **fsck** (the file system check command.)
- The Glossary defines the hardware and software terminology used in this *Guide.*
- The Index gives an alphabetical listing of topics, together with the page numbers on which they appear in this *Guide.*

Notational Conventions

Throughout this *Guide*, there are certain conventions used to illustrate responses and how to react to them. The following is a list of the conventions you will see:

- Commands and command options are in **bold** type.
- File names and references to other documents are in *italic* type.
- Literal command examples, screen prompts, and screen displays are in `monospace`. Only one frame will be shown in screen layout figures. Partial screens are used and may not exactly duplicate what appears on your terminal screen.
- Keyboard references are in initial capital letters (to match the computer keyboard layout) and boxed. In this document the return key is represented by [Enter]. Also, the notation [Ctrl] [d] means that you hold down the control key and strike the specified alphanumeric key, in this case, lowercase d.
- Screen-labeled keys are in all capital letters, with the key number inside parentheses [e.g., CANCEL, SAVE, CONT].

Foundation Set Software Packages

The Foundation Set is the fundamental UNIX System software product supplied with your system. The Foundation Set provides you with the UNIX Operating System kernel and a basic set of utilities. The Foundation Set consists of the following separately installable packages:

- Base System
- Editing
- Remote Terminal
- Security Administration
- 2 Kilobyte File System
- Network Support Utilities
- Remote File Sharing
- XENIX File System

The Base System package is the minimal required UNIX System. The other Foundation Set packages are optional, and you do not need to install them if you do not require the utilities they provide. Of course, you may install them later if you desire, and with the exception of the Remote Terminal Package, you may remove add-on packages if desired.

Base System

The Base System provides you with the UNIX Operating System kernel, standard device drivers, and approximately ninety of the most essential UNIX System utilities. The Base System allows you to

- Manage the hardware, software, and users on your computer system
- Perform Basic Networking (**uucp**) queued file transfers
- Share printers through the temporary storage of data and queuing of printer requests
- Tune system parameters to maximize performance under various load conditions, system configurations, and applications
- Modify the UNIX System to incorporate additional device drivers
- Run cooperating processes that share data and communicate with each other
- Perform mathematical calculations
- Check or change the executing environment of commands
- Schedule commands to be run at a later time
- Perform process accounting functions
- Install optional Foundation Set software packages and application software

Editing Package

The Editing Package provides three related editors based on a consistent set of text editor commands: two line editors (**ed** and **ex**) and a screen editor (**vi**). The **ed** editor is mainly for novice users. The **ex** editor is an advanced version of **ed** and is for experienced users. The **vi** editor is a screen editor that allows you to use your terminal as a window for viewing the text of a file. Within this window, you can add, delete, or change text in much the same way you would on a typewriter or with paper and pencil.

The Editing Package contains **spell** utilities that check for misspelled words in a file. It also contains other utilities that are helpful in developing shellscripts and examining files at the shell prompt.

Remote Terminal Package

The Remote Terminal Package consists of the *terminfo* data base that contains the descriptions and operating capabilities of over 150 popular terminal devices and terminal filters that allow a variety of terminals to print formatted output.

Security Administration Package

The Security Administration Package provides you with an encryption mechanism for protecting data that is being stored or transmitted. It gives your computer additional security protection beyond that obtained through login IDs, passwords, and permission modes. (This package has restricted distribution and is available only with systems intended for use within the United States.)

2 Kilobyte File System Utility Package

The 2 Kilobyte File System Utility Package provides an optional method of file system organization employing larger (2K) block sizes to improve disk input/output (I/O) performance.

Network Support Utilities Package

The Network Support Utilities Package supplements the Base System by extending system capabilities to support networking applications. This includes standard STREAMS protocol modules (for use by applications), a network utility that monitors network service requests (the Listener), and the version of the sharable Network Services Library required to compile application programs.

Remote File Sharing Package

The Remote File Sharing (RFS) Package allows you to share resources (directories containing files, subdirectories, devices, and named pipes) across a network with computers running UNIX System V, Release 3.0 or later releases. Administrators for computers on an RFS network can choose which directories on their systems they want to share and add them to a list of available resources on the network. From this list, they can choose resources from other computers on the network that they would like to use on their computers.

Each computer on an RFS network can be grouped with others in a "domain" or can operate as an independent domain. The domain can provide a central point for administering a group of computers. Unlike other distributed file systems used with the UNIX Operating System, RFS is built into the operating system itself to provide compatibility, security, and flexibility.

XENIX File System

The XENIX File System package provides support for mounting and using XENIX file systems just as you would under XENIX System V.

Installing Software

The UNIX System is delivered on diskettes organized in discrete packages, as described in the previous section. Before you can use your computer, you have to install the minimum Base System.

The first floppy diskette in the Base System contains the UNIX Operating System kernel and all commands necessary to install the rest of the Base System. After you have loaded the first diskettes, you then reboot the system from the hard disk.

After rebooting, you are ready to install the rest of the diskettes to complete the installation of the Base System.

With the Base System installed, you can then use the UNIX System automatic installation program to load the optional add-on packages that contain the rest of the UNIX System.

Getting Started

When you first turn on your computer (after all the appropriate software has been installed), it goes through a series of checks to make sure your computer is all right. The checks may take several minutes.

The following is a list of checks your computer makes when it is turned on:

- It runs a "resident diagnostic" program to check the hardware in your computer.
- It determines whether the UNIX System was brought down correctly the last time it was used.
- It boots the UNIX System and runs a file system check on system files if needed. The file system is checked for inconsistencies; if one should occur, the program will try to repair it.

You will then see the login prompt when the system checks are complete and everything is satisfactory.

Starting a Session With Your Computer

A login name is one way of ensuring security on your computer system. Everybody using the computer has a different login name assigned to him/her. See Chapter 4, "System Administration," to learn how to assign login names. When you turn on your computer, the following prompt will be displayed:

```
Console login:
```

If you are logging in from a remote terminal, your login screen will look like the following:

```
System name: <name>

Please login:
```

You can now enter your login name, followed by striking Enter. Your computer will now prompt you for your password. As you type in the password, it will not be displayed on the screen:

```
Password:
```

After you have entered the password and struck Enter, your computer will display the shell prompt ($) and wait for the next command.

The Shell

The UNIX System shell is the starting point for your work with the computer. Each time you log in to the computer, you're in the shell. From here, you can perform user-oriented procedures.

Shutting Down Your System

If you want to turn your computer off, always shut down the computer first or your files could be damaged. Shutting down your computer before turning off the power is necessary to ensure the integrity of your file system.

In Case of Trouble

The UNIX System is designed to run your programs without difficulty. Occasionally, faulty software or hardware may cause problems. If you cannot get your computer to work correctly, try the following:

1. If your keyboard is locked up and will not respond to input, strike Ctrl q.

   If Ctrl q does not work and you are at the console, strike Ctrl Break. If you are at a terminal, turn the terminal off.

2. Consult the documents provided with the computer.

3. Log off and then log back in again. If this does not help, go to the next item.

4. Shut down the computer according to the shutdown procedure in Chapter 4, "System Administration." At the end of the shutdown procedure, strike the HARDWARE RESET button to reboot.

5. If this does not work, strike the HARDWARE RESET button.
   Do this step only if procedures 1 through 4 are unsuccessful.

Chapter 2: Software Installation

Introduction to Software Installation

This chapter tells you how to install the UNIX System V/386 Operating System, Release 3.2, Version 1, on your system. First, you are provided with the detailed steps required to install the Base System, which is the set of UNIX System utilities required for most computing environments. Then, you are directed through the procedure for installing the optional add-on packages. In addition, this chapter tells you how to display installed software packages, as well as how to remove add-on packages.

NOTE

Please consult the *Release Notes* before starting the software installation. The *Release Notes* contain last minute information that may affect software installation.

Read Error Condition

If, when reading any diskette during installation of the Base System, a read error occurs (e.g., the diskette is not inserted, is misinserted, or is the wrong density), this shall be referred to in this document as a "Read Error" condition.

You will see the following message a maximum of three times until the error is corrected or you strike [Break] (a break causes the installation process to abort, as discussed on the next page).

```
A floppy diskette read error has occurred. If necessary, please
consult your "Operations/System Administration Guide".

After correcting the error, strike ENTER to continue.
```

If you strike [Enter], the attempt will be made again. If three attempts are

made and the diskette still cannot be read, you will see the following termination message and you will be left in single-user mode with the hard disk unmounted:

```
Unable to read floppy diskette. Installation terminated.
You may attempt to restart the installation process,
but if this problem recurs, please contact your support
representative immediately.
```

NOTE

If the hard disk is unmounted, it is no longer accessible by the system. Refer to *mount*(1M) in the *User's/System Administrator's Reference Manual.*

Aborting the Installation Procedure

If at any point during the installation of diskettes you strike Break thereby aborting the installation, you will see the following "Abort Message":

```
You have aborted the installation of the UNIX
System. If you wish to rerun it, type INSTALL at
the prompt.  Please consult your "Operations/System
Administration Guide" for further information.
```

The installation process aborts but may be restarted at any time by typing **INSTALL**. You are left in the single-user mode with hard disk unmounted.

Hard Disk Formatting

The assumption is that your hard disk has been low-level formatted by you or the manufacturer. Usually the hard disk is customarily low-level formatted for your convenience. You can verify that your hard disk is formatted and determine the interleave by using the "Low Level Format Utility" diskette available with your 386 hardware. The low level format utility allows you to prepare a new disk or reformat a previously formatted one. For more information, refer to your *User's Guide*.

Install Base System Diskette Number 1

Diskette number 1 contains the full Base System kernel and all commands necessary to install the remaining diskettes in the Base System. The following section describes how to install diskette number 1.

Boot System to Single-User Mode

1. Insert diskette number 1 into the diskette drive.
2. Press the hardware reset button or turn the system on to boot from the diskette.

   First, you will see the hardware diagnostics and then a message resembling the following:

```
total real mem = 3145728
total avail mem = 2031616

UNIX System V/386 Release 3.2

Copyright (c) 1988 AT&T
ALL RIGHTS RESERVED

Copyright (c) 1987, 1988 Microsoft Corporation
ALL RIGHTS RESERVED

386/ix Drivers Copyright (C) 1986 Interactive Systems Corporation
ALL RIGHTS RESERVED
```

NOTE

The amount of memory displayed will vary according to how much memory you have in your system.

3. When the Base System kernel has booted successfully into single-user mode, the system verifies that there is a minimum of 2 megabytes of system memory installed in the computer.

   If the 2-megabyte minimum is not installed, you will see the following message:

```
WARNING: Your system does not have the recommended minimum
2 Megabytes of memory. You may wish to power down the
machine, add memory, and begin the installation process again.
```

4. You will see the following prompt message (whether or not 2 megabytes are available):

```
Strike ENTER to install the UNIX System on your hard disk.
```

5. Strike [Enter] to install the UNIX System on your hard disk.

Partition the Hard Disk

Your hard disk can contain both a UNIX System partition and an MS-DOS partition. If you want to have both, start the MS-DOS partition on cylinder 0, and place the UNIX System partition on the cylinders above the MS-DOS partition.

The following steps describe how to partition your hard disk.

1. Your screen should look similar to Screen A or Screen B shown on the next page.

2. Determine which screen layout is shown on your display.

Screen A

If you want to partition as shown on Screen A, type **y**, strike [Enter], and then refer to the section "Prepare Hard Disk For Surface Analysis," for further instructions.

If not, type **n** and strike [Enter] to tell the system that you do not want to set up your hard disk, as shown in the message. Then go to Step 3.

Screen B

If the screen shows an active UNIX System partition, proceed to Step 8.

If the screen shows an MS-DOS partition that uses all of the hard disk, you need to delete the MS-DOS partition before continuing. Deleting a partition destroys all files in that partition. Before you delete the partition, copy any files you want to save to diskettes or optional tape. To delete the partition, type **3** and strike [Enter], and then type the number of the partition you want to delete and strike [Enter] again. After the partition has been deleted, go to Step 3.

NOTE

It is only necessary to delete the MS-DOS partition if you do not have space for the UNIX System partition.

SCREEN A

```
Do you want to partition your hard disk as follows:

   90% "UNIX System"


   10% "MS-DOS(v. 3.2 or later) only"


To do this, please type "y". To partition your hard
disk differently, type "n" and the "fdisk" program will
let you select other partitions.
```

SCREEN B

```
Total hard disk size is 980 cylinders

                                          Cylinders
Partition  Status  Type              Start  End  Length   %
---------  ------  --------          -----  ---  ------  ---
    1           Active  MS-DOS            0  979     980   100

SELECT ONE OF THE FOLLOWING

   1. Create a partition
   2. Change Active (Boot from) partition
   3. Delete a partition
   4. Exit (Update disk configuration and exit)
   5. Cancel (Exit without updating disk configuration)

Enter selection:
```

NOTE

The actual numbers shown in Screen B for hard disk size will depend on the size of your hard disk.

3. Make sure your screen looks like this with no partitions defined:

```
Total hard disk size is 980 cylinders

                                    Cylinders
Partition  Status  Type      Start  End  Length   %
---------  ------  --------  -----  ---  ------  ---

THERE ARE NO PARTITIONS CURRENTLY DEFINED

SELECT ONE OF THE FOLLOWING

   1. Create a partition
   2. Change Active (Boot from) partition
   3. Delete a partition
   4. Exit (Update disk configuration and exit)
   5. Cancel (Exit without updating disk configuration)

Enter selection:
```

4. Type *1* and strike [Enter] to select "Create a Partition."

   You will see the message:

```
Indicate the type of partition you want to create
(1=UNIX System, 2=MS-DOS only, 3=Other, x=Exit).
```

5. Type *1* and strike Enter to select "UNIX System."

   You will see the message:

```
The UNIX System partition must use at least nnn% of the
hard disk.  Indicate the percentage (nnn-100) of
the hard disk you want this partition to use
(or enter "c" to specify in cylinders).
```

NOTE

nnn depends on the size of your hard disk.

6. Type *100* and strike Enter.

   You will see the message:

```
Do you want this to become the Active partition?
If so, it will be activated each time you reset
your computer or when you turn it on again.
Please type "y" or "n".
```

NOTE

When your computer is turned on or reset, it looks for a diskette in the floppy diskette drive. If it does not find one, it searches the hard disk for an active partition from which it can load an operating system.

7. Type *y* and strike Enter to make the UNIX System partition active.

   At the bottom of your screen, you will see the message:

```
Partition 1 is now the Active partition.
```

After the partition is created, your screen should resemble this:

```
Total hard disk size is 980 cylinders


                                          Cylinders
Partition  Status  Type              Start  End  Length   %
---------  ------  --------          -----  ---  ------  ---
    1      Active   UNIX System          0  979     980   100

SELECT ONE OF THE FOLLOWING

   1. Create a partition
   2. Change Active (Boot from) partition
   3. Delete a partition
   4. Exit (Update disk configuration and exit)
   5. Cancel (Exit without updating disk configuration)

Enter selection:
```

8. Type *4* and strike Enter to select "Exit."

Prepare Hard Disk for Surface Analysis

1. You will see the following message:

```
Hard disk partitioning complete.
```

2. The system tests to determine whether the active UNIX System partition has already been prepared for creation of a UNIX System file system.

   If the above tests pass, you will see the following message:

```
Do you wish to check your UNIX System partition again for bad tracks?
NOTE: Whether you answer "y" or "n"
ALL DATA ON YOUR SYSTEM PARTITION WILL BE LOST!
If you wish to abort the installation process at this point,
please strike DEL otherwise,
strike "y" for surface analysis or "n" to continue.
```

   If the active UNIX System partition had not been prepared, go directly to Step 4.

3. Set up your UNIX System partition as follows:

   - If you wish to set up a UNIX System partition, type *y* and strike [Enter] to continue.

     Go on to Step 4.

   - If you do not wish to set up the UNIX System partition, type *n* and strike [Enter]. You will see the following message:

```
UNIX System partition unchanged.
```

     Go directly to the "Create UNIX System File Systems" section.

     If you enter anything other than *y* or *n* followed by [Enter] or [Break] (to be treated as *n*), you will see the following message until you enter a valid answer:

```
Surface analysis (y or n)?
```

4. You will see the following message:

```
Checking for bad sectors in the UNIX System partition...
```

   This message tells you that the System is performing a surface analysis of the hard disk and generating a table of the defective blocks. While

the UNIX System partition is being scanned, you will see the following message:

```
Checking cylinder: nnn
```

where `nnn` is updated for each cylinder scanned.

If the table overflows during the identification of defective blocks, you will see the following message, the installation will abort, and you will be left at the UNIX System prompt:

```
Too many bad tracks found (nnn).
The UNIX System cannot be installed on this disk. Try the installation again
but make sure the UNIX System partition doesn't begin on cylinder nnn.

Installation aborted.
```

If this occurs, restart the installation process by typing **INSTALL**. When you perform the **fdisk** procedure (see the "Partition the Hard Disk" section), manually place the UNIX System partition somewhere else. If this does not clear the problem, contact the manufacturer regarding the integrity of the hard disk.

Create UNIX System File Systems

1. When the surface analysis is completed, the system tests to see if a valid file system exists on the active UNIX System partition. The system calculates the optimal amount of space on your hard disk for swap, user, and/or *root*(/) file systems. You will see a message that begins as follows:

```
The UNIX System partition has nnn cylinders assigned to it.
nn cylinders will be used for alternate sectors and tracks.
This leaves nnn cylinders (nnnnn bytes) available.

The following seems like a reasonable partitioning of
your UNIX System disk space:
```

NOTE

Throughout this procedure, `nnn` depends on the partition size.

2. You will see the file system size selections in the following messages:

   If no separate */usr* file system was called for, based on your partition size, you will see the following message:

```
A combined root/user filesystem of nnn cylinders(nnnnnnn bytes),
```

   Otherwise, you will see the following message:

```
A root filesystem of nnn cylinders (nnnnnnn bytes),
a user (usr) filesystem of nnn cylinders (nnnnnnn bytes),
```

In either case, if an additional */usr2* file system were called for, you will see the following message:

```
an extra user filesystem (/usr2) of nnn cylinders (nnnnnnn
bytes),
```

The message will be completed with:

```
and a swap/paging area of nnn cylinders (nnnnnnn bytes).
```

3. You will see the following prompt message:

```
Is this allocation acceptable to you (y/n)?
```

If this allocation is acceptable, type *y*, strike Enter, and then proceed to Step 12.

If unacceptable, type *n* and strike Enter.

4. You will see the following message:

```
Do you wish to have separate root and user filesystems (y/n)?
```

If you wish to have separate root and user file systems, type *y* and strike Enter.

If you wish that root and user be combined, type *n* and strike Enter.

NOTE

With many users and limited hard disk space (less than 68 megabytes), it is advisable to create separate root and user file systems.

5. You will see the following message:

```
Do you want an additional /usr2 filesystem (y/n)?
```

Generally, an additional */usr2* file system is not necessary. However, if you wish to have an additional */usr2* file system, type *y* and strike Enter.

If you do not want the above, type *n* and strike Enter.

6. You will see the following message:

```
You will now be given the opportunity to specify the
size, in cylinders, of each filesystem. (One megabyte
of disk space is approximately nn.n cylinders).
```

followed by the prompt message:

```
How many cylinders would you like for swap/paging (1-nnn)?
```

NOTE

In the above message, `nnn` is the maximum legal size of swap space, calculated as the total space (UNIX System partition less cylinders reserved for alternates) minus 20 megabytes.

Enter the desired parameter and strike Enter. If your answer was not in the given range, you will see the following message, and Step 7 will be repeated:

```
Illegal value: nnn; try again.
```

If only one file system was selected, you will see the following message:

```
The remaining nnn cylinders will be assigned to root/usr.
```

NOTE In the above message, nnn is the remaining space after swap space is subtracted.

If only one file system was selected, then proceed to Step 10.

7. You will see the following message:

```
How many cylinders would you like for root (1-nnn)?
```

NOTE In the above message, nnn is the maximum legal size for *root* (/), which is the amount of space remaining after swap space has been subtracted.

Enter the desired parameter and strike Enter.

If your answer is not in the given range, you will see the following message, and Step 8 will be repeated:

```
Illegal value: nnn; try again.
```

If there is no space left after subtracting the *root*(/) file system, only one file system shall be used, and you will see the following message. Then proceed to Step 10.

```
No space remaining for a user filesystem.
Assuming single root/usr filesystem.
```

If an additional */usr2* file system was not selected, you will see the following message. Then proceed to Step 10.

```
The remaining nnn cylinders will be assigned to /usr.
```

If an additional */usr2* file system was selected, you will see the following message:

```
The remaining nnn cylinders will be assigned to /usr2.
```

8. You will see the following message:

```
How many cylinders would you like for /usr (1-nnn)?
```

NOTE

In the above message, `nnn` is the maximum legal size for */usr*, which is the amount of space remaining after *root*(/) has been subtracted.

Enter the desired parameter and strike Enter.

If the answer is not in the given range, you will see the following message, and this step will be repeated:

```
Illegal value: nnn, try again.
```

If there is no space after subtracting the */usr* file system, only two file systems shall be used, you will see the following message, and the process will continue with Step 10:

```
No space remaining for a /usr2 filesystem.
Assuming just root and /usr filesystems.
```

Otherwise, you will see the following message:

```
The remaining nnn cylinders will be assigned to /usr2.
```

9. You will see the following message:

```
You have specified the following disk allocation:
```

Return to Step 3.

10. You will see the following message:

```
UNIX System file system(s) will now be created
  on your hard disk...
```

Verify Successful File System Creation

1. If an error was encountered in the creation of any of the UNIX System file system(s), you will see the following message:

```
An error has occurred while setting up your hard disk.
Strike ENTER to install again.
```

If you see this message, strike [Enter] to install again, and go back to the "Partition Your Hard Disk" section.

NOTE

If you strike [Break], the Abort Message and procedure in the "Aborting the Installation Procedure" section will be deployed.

2. If the file system was created successfully, you will see the following message:

```
UNIX System file system(s) have been created
in your active UNIX System partition.

A UNIX System will now be installed on your
hard disk.....

Please standby.
```

Go on to the next step.

Complete Installation of Diskette Number 1

1. The *root*(/) file system is mounted. If this fails, you will see the following message (up to three times):

```
Mounting root file system failed, trying again ...
```

   If the mount attempt fails three times, you will see the following message:

```
Cannot mount the root file system.
Please notify your AT&T services representative for further
assistance.
```

   If the above occurs, the installation will abort, and you will be left in single-user mode with the hard disk unmounted.

2. All files on diskette number 1 are copied to their respective directories on the hard disk.

3. You will see the following message:

```
When you are prompted to reboot your system,
remove the floppy diskette from the diskette drive,
and strike CTRL-ALT-DEL.
Please wait for the prompt.

Reboot the system now.
```

4. Strike Ctrl, Alt, and Del simultaneously to reboot the system.

NOTE

If you strike any key other than Ctrl, Alt, and Del simultaneously, it will be ignored by the system.

Install the Remainder of the Base System

When the system has rebooted from the hard disk, a procedure is automatically initiated that

- copies all files from the remaining diskettes to the hard disk
- sets the date and time
- sets the system name
- sets special login passwords for root and install
- presents the login prompt

If at any time you strike [Break], you will not receive a message and will be left in single-user mode at the prompt with the hard disk mounted and in the *root*(/) directory on the hard disk. You may restart this procedure by typing **INSTALL**.

The procedure to install the remainder of the Base System is as follows:

1. You will see the following message for each diskette remaining, beginning with diskette number 2:

```
Please insert the UNIX System "Base System Package"
Floppy Disk <n> of 8 and then strike ENTER.
```

   Strike [Enter], and the following "in-progress" message will appear:

```
Installation is in progress -- do not remove the floppy diskette.
```

2. The sequence number of the diskette shall be verified. If a floppy read error is detected, the Read Error procedure in the "Read Error Condition" section shall be used. If the error is corrected and you strike [Enter], the "in-progress" message will reappear.

3. If the sequence number of the diskette is not correct, you will see the following message until you insert the correct diskette or strike Break.

NOTE	A break leaves you in single-user mode at the prompt with no message and with the hard disk mounted.

```
The inserted floppy diskette is incorrect.  Please insert
the floppy diskette labeled <n> of 8 and strike ENTER.
```

When you strike Enter, the "in-progress" message appears. Again, if a read error occurs, the procedure in the "Read Error Condition" section is used.

4. Once the correct diskette has been inserted, its contents will be copied to the hard disk.

5. Repeat this procedure (beginning at Step 1) until all Base System diskettes are read into the system.

Wrapup Base System Installation

1. When you have completed the installation of the Base System diskettes, you will see the following message:

```
UNIX System files have been copied to the hard disk. It is now safe
to remove the floppy diskette. Additional system files will now be set up.
Please stand by ...
```

The system is checked. This will take a few minutes.

2. After the system time is displayed, you will see the following prompt message:

```
Enter a password for the "root" or superuser.

(Note: This password must be kept EXTREMELY secure):
New password:
```

Enter the "root" password and strike [Enter].

You will see the following message:

```
Re-enter new password.
```

Re-enter the "root" password and strike [Enter].

3. You will see the following message:

```
Enter a password for the "install" user.

(Note: This password must be kept EXTREMELY secure
and should be different from the root password):
New password:
```

Enter the "install" password and strike [Enter].

You will see the following message:

```
Re-enter new password.
```

Re-enter the "install" password and strike [Enter].

4. You will then see the following message:

```
The UNIX System installation process is now complete.

To install the Foundation Set Add-on packages, use
the "installpkg" command from the UNIX System prompt.

Be sure the floppy drive is empty and
strike CTRL-ALT-DEL to reboot your newly
configured UNIX System.

Reboot the system now.
```

Reboot the UNIX System

1. Strike Ctrl, Alt, and Del at the same time to reboot the UNIX System from the hard disk.
2. You will see the following message:

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

3. After the diagnostics have been successfully run, the following message will be displayed:

```
total real mem = 3145728
total avail mem = 2031616

UNIX System V/386 Release 3.2

Copyright (c) 1988 AT&T
ALL RIGHTS RESERVED

Copyright (c) 1987, 1988 Microsoft Corporation
ALL RIGHTS RESERVED

386/ix Drivers Copyright (C) 1986 Interactive Systems Corporation
ALL RIGHTS RESERVED

Console login:
```

4. After you have logged in, you should make a copy of the first diskette for backup.

NOTE	The first diskette contains necessary files in the event your UNIX System becomes corrupted.

Install Optional Add-On Packages

After you have completed the installation of the Base System, you are ready to install the add-on packages that make up the rest of the Foundation Set, as well as other third-party software packages.

This section provides an overview of the installation process and a general installation procedure that applies to installing all add-on packages except the Remote Terminal Package.

NOTE	The procedures to install the Remote Terminal Package are contained in a separate section in this chapter.

Overview of Installation Software

The Base System Package contains a System Add-On Installation Procedure program which installs add-on software packages. The first diskette in each add-on software package contains an Add-On Install Script that is used by the System Add-On Installation Procedure to physically install the package.

General Instructions

Use of the Enter and Esc Keys

Throughout the procedure to install Optional Add-On packages, the following will apply for error conditions (except as noted otherwise) whenever you are prompted to strike [Enter] or [Esc]:

- If you strike [Esc], you will see the following message:

```
The Installation is canceled.
```

- If you strike [Enter], the installation procedure will be restarted with the procedure in the "Install First Add-On Package Diskette" section.
- If you strike any keys other than [Enter] or [Esc], the system will beep.

Use of the Break Key

If you strike [Break] at any point during the software installation up to the point where the application's Install script begins executing, you will see the following message:

```
You have canceled the installation.
```

Your response to the above prompt will be as described in the preceding paragraph.

If you strike [Break] while the application's Install script is executing, the break will be ignored and no message displayed in order to leave the system in a sane state.

Diskette Handling

If the diskette drive door is open, the diskette has an error, or the diskette is incorrectly inserted, you will see the following message:

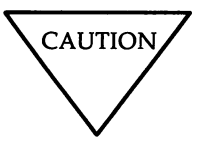
```
An error was encountered while reading in the
floppy diskette(s). Please be sure to insert them
in the proper order, that the drive door is
closed, and wait for the notification before
removing them.

If this problem reoccurs at the same floppy diskette,
the floppy diskette may be bad. Please re-insert
floppy diskette number 1 and try again.

Strike ENTER to continue
or ESC to stop.
```

You will be given the opportunity to correct the problem and start over.

When add-on packages are installed, files are copied to a temporary "holding" directory so that checks may be performed as the installation process continues.



Do not open the diskette drive door while files are being installed.

If, during the copying of files from the diskette to the hard disk, the hard disk runs out of space in the temporary directory, the temporary directory and any/all of its contents will be removed, the installation procedure will abort, and you will see the following message:

```
Your user (/usr) filesystem is out of space.
Please remove some files and try again.

Installation aborted.
```

If the system finds that the diskette's file header information is invalid, you will see the following message:

```
The floppy diskette you inserted is either not the correct floppy diskette
or you inserted it in the wrong order. If this problem reoccurs at
the same floppy diskette, the floppy diskette may be bad. Please re-insert
the first floppy diskette and try again.

Strike ENTER to continue
or ESC to stop.
```

If you wish to continue, remove the diskette and start again at Step 1 of the "Install the First Add-On Package Diskette" section.

If you wish to stop here, strike Esc.

Check for Install Permission

To install a software package, do the following:

1. Type **installpkg** at the UNIX System prompt, and strike Enter.
2. The system checks to see if you have permission to execute privileged operations. If you do not, you will see the following message:

```
You (<logname>) do not have permission to
perform software installation.
Please consult your Operations/System
Administration Guide for more
information on assigning permissions
to privileged operations.
```

   Refer to Chapter 4, "System Administration."

3. The system also checks to see if you are at the console. If not, you will see the following message:

```
You must be on the console to run installpkg.
```

   The procedure terminates.

Install First Add-On Package Diskette

1. You will see the following message:

```
Confirm

Please insert the floppy diskette.

If the program installation requires more than
one floppy diskette, be sure to insert the disks in
the proper order, starting with disk number 1.
After the first floppy diskette, instructions will
be provided for inserting the remaining floppy
diskettes.

Strike ENTER when ready
or ESC to stop.
```

NOTE If your system has two disk drives, the following prompt will appear:

```
The system has two floppy drives.
Strike ENTER to install from drive
0
or 1 to install from drive 1.
```

2. Insert diskette number 1 into the drive and strike Enter.

   You will see the following message:

```
Installation is in progress -- do not remove the floppy diskette.
```

3. Your screen will be cleared, and the system will attempt to extract the file called *Size* from the first diskette. While this attempt is in progress, you will see the following message:

```
Searching for the Size file
```

4. If there is enough room on the hard disk to install the package, you will see the following message:

```
                Install in progress
```

When you see the above message, go directly to Step 7.

5. If the *Size* file is not found, you will see the following message:

```
Please enter the number of floppies in
the package followed by ENTER:
```

Enter the number of diskettes and strike [Enter].

NOTE

Until you enter a numeric value greater than or equal to 1 followed by [Enter], the system will beep for each invalid response.

You will see the following message:

```
Please enter 1 (for 360 KB) or 2 (for 1.2 MB)
for disk density followed by ENTER:
```

Enter *1* or *2* and strike Enter.

NOTE

Until you enter either a *1* or *2* followed by Enter, the system will beep for each invalid response.

Go back to Step 1 of the "Install First Add-On Package Diskette" section. After Step 1, go directly to Step 8.

6. If the *Size* file is found but the system determines that it is invalid, you will see the following message:

```
Invalid Size file found. Cannot determine disk requirements.
```

Enter the number of floppies and density as directed in the preceding step.

7. If the package's hard disk requirement exceeds the space available on either the *root*(/) or *usr* file system, you will see the following message:

```
There is not enough room on the hard
disk to install the package. Please
remove some files from the <filesystems>
filesystem(s) and try again.
```

NOTE

In the above message, "<filesystems>" may be replaced with "*root*(/)" to indicate the *root*(/) file system or "user (*/usr*)" for the *usr* file system. If both file systems lack adequate space, "*root*(/) and user(*/usr*)" shall be used.

8. When loading of the diskette is completed and the package has more than one diskette, you will see the following message:

```
Reached end of medium on input.
You may remove this floppy diskette.
To QUIT - strike <q> followed by ENTER.
To continue - insert floppy diskette number <n+1>
and strike the ENTER key.
```

NOTE

n is the current diskette number. Example: If this is diskette number 1, the n+1 diskette will be diskette number 2.

When you see this message, remove the diskette from the drive.

9. If the package consists of only one diskette, go directly to the "Complete Installation of Add-On Package" section.

Install Additional Add-On Package Diskettes

1. When loading of a diskette has completed and there are more diskettes in the package, you will see the following message:

```
Reached end of medium on input.
You may remove this floppy diskette.
To QUIT - strike <q> followed by ENTER.
To continue - insert floppy diskette number <n+1>
and strike the ENTER key.
```

2. If you have additional diskettes to install for this package, insert the next diskette in the drive and strike Enter.
3. If this was the last diskette in the package, go directly to the "Complete Installation of Add-On Package" section.

Complete Installation of Add-On Package

1. The system tests for the presence of the required installation files. If any are missing, you will see the following message:

```
Confirm

The software package is missing the
necessary installation programs. Please
check to make sure you have the right
floppy diskette(s).

Strike ENTER to restart installation
or ESC to stop.
```

   If you receive the above message and wish to restart the installation, strike [Enter]. If you wish to cancel the installation, strike [Esc].

2. If the system determines that an identical software package has already been installed, you will see the following message:

```
Confirm

The <package name> package has already
been installed. The new installation
will now replace the original <package name>
files.

Strike ENTER to continue
or ESC to stop.
```

   You may strike [Enter] to replace the previous "instance" of the package with the new one and continue the installation, or you may strike [Esc] to cancel the installation.

   If you strike [Esc], you will see the following message:

```
The Installation is canceled.
```

and you will be left at the command line prompt.

NOTE

If you type anything other than [Enter] or [Esc], the system will beep.

3. Your screen will clear and the package's **Install** script will begin execution.

NOTE

If the package just installed contained a UNIX System driver, you will see messages informing you that the UNIX System kernel is being reconfigured. This takes a few minutes. After reconfiguring the kernel, the UNIX System will automatically be shut down, forcing you to reboot the UNIX System with the newly configured kernel.

4. If no driver was installed and the installation is successful, you will see the following message:

```
The installation of the <package name>
package is now complete.
```

You will be left at the command line prompt.

5. If the installation is not successful, you will be left at the UNIX System prompt with the installation terminated.

Install the Remote Terminal Package

The following sections describe installing the Remote Terminal package in the Foundation Set (i.e., adding a terminal definition or definitions to your system for use with applications).

Check for Install Permission

1. Type **installpkg** at the UNIX System prompt and strike Enter.
2. The system checks to see if you have permission to execute privileged operations. If you do not, you will see the following message:

```
You (<logname>) do not have permission to
perform software installation.
Please consult your Operations/System
Administration Guide for more
information on assigning permissions
to privileged operations.
```

   Refer to Chapter 4, "System Administration."

3. The system also checks to see if you are at the console. If not, you will see the following message:

```
You must be on the console to install software.
```

   The procedure terminates.

Install Remote Terminal Package Diskette

1. You will see the following message:

```
Confirm

Please insert the floppy diskette.

If the program installation requires more than
one floppy diskette, be sure to insert the disks in
the proper order, starting with disk number 1.

After the first floppy diskette, instructions will
be provided for inserting the remaining floppy
diskettes.

Strike ENTER when ready
or ESC to stop.
```

2. Insert diskette number 1 of 1, containing the Remote Terminal Package, into the drive and strike Enter.

Install Terminal Files

You will see the following message:

```
Installation is in progress--do not remove the floppy diskette.

Installing the Remote Terminal Package Version 1.0.

The following files are being installed:
/usr/options/terminf.name
Please install the terminal files you wish from the diskette.

Selective installation of the Remote Terminal Package Version 1.0 database.

    0     Terminate installation

    1     Install terminfo file(s)

    2     Locate a specific terminal within terminfo file(s)

    3     Compile a SINGLE terminal entry

Enter option:
```

Locate a Terminal File

If you wish to locate the terminal information file for a specific terminal within the *terminfo* data base on the diskette, type 2 and strike Enter.

You will see the following message:

```
Enter terminal name to be located:
```

Suppose you wish to locate the file for an AT&T 5425 terminal. In that case, you would type the terminal name *5425* and strike Enter.

You will see the following message:

```
Terminal 5425 is located within terminfo file "att.ti"
```

Select File(s)

You may wish to display the *terminfo* data base file listing and select a file(s) to install. In that case, you will type *1* and strike Enter.

You will see the following message:

```
The following terminfo files may be selected for installation:

adds.ti          annarbor.ti      ansi.ti          att.ti
beehive.ti       cdc.ti           colorscan.ti     contel.ti
datamedia.ti     dec.ti           diablo.ti        fortune.ti
general.ti       hardcopy.ti      hazeltine.ti     hds.ti
heath.ti         homebrew.ti      hp.ti            lsi.ti
microterm.ti     misc.ti          pc.ti            perkinelmer.ti
print.ti         special.ti       sperry.ti        tektronix.ti
teleray.ti       televideo.ti     ti.ti            tymshare.ti
visual.ti

Enter a file name, "all", "done", or "files":
```

You may enter a file name to install an entire file or type one of the following and strike Enter:

- **all** to install all files in the *terminfo* data base
- **done** to return to the main menu
- **files** to relist the files (see above)

Compile a Single Terminal Entry

You may wish to compile a single terminal entry. To do that, type *3* and strike Enter.

You will see the following message:

```
Enter terminal name:
```

If, for example, you wish to enter an AT&T 5425 terminal, type the terminal name *5425* and strike Enter.

You will see a message similar to the following:

```
Working in /usr/lib/terminfo
Created 5/5425
Linked 4/4425
Linked A/ATT4425
Linked A/ATT5425
Linked a/att4425
Linked a/att5425
Linked t/tty5425
```

Complete Installation of Remote Terminal Package

When you are ready to complete the installation of the Remote Terminal Package, type *0* and strike Enter.

You will see the following message:

```
The installation of Remote Terminal Package Version 1.0 is now complete.
```

You will be left at the UNIX System prompt.

Display Installed Software Packages

To get a sorted list of the software packages currently installed on your system, do the following:

1. Type **displaypkg** at the UNIX System prompt and strike Enter.

   A sorted listing of the currently installed software packages will be displayed.

2. Use

   - /
   - ?
   - -

   and all documented features of the **pg** command to view the list. Refer to the *User's/System Administrator's Reference Manual* for more information on the *pg*(1) command.

   NOTE

   pg is normally installed as part of the Editing Package. You do not need to install the Editing Package to use **displaypkg**.

3. When you are finished, enter **q** at the ":" prompt. This will leave you at the UNIX System prompt.

Remove Add-On Software Package

If your hard disk is low on space, you may wish to remove some software packages that are not commonly used. The **removepkg** command lets you do this.

You may type **removepkg** followed by "*package name*" (in double quotes) to remove a package, or you may type **removepkg** without arguments and select the package that you wish to remove from a list of installed packages.

1. Enter the **removepkg** command as follows:

   - To remove a package on the command line, type **removepkg** "*package name*" and strike Enter.

   NOTE: The package names are identified in the */usr/options* directory.

   - To select the package from a list, type **removepkg** (without arguments) and strike Enter.

2. The system checks to see if you have permission to execute privileged operations. If you do not, you will see the following message:

```
You (<logname>) do not have permission to
perform software removal.
Please consult your Operations/System
Administration Guide for more
information on assigning permissions
to privileged operations.
```

Refer to Chapter 4, "System Administration."

3. The system also checks to see if you are at the console. If not, you will see the following message:

```
You must be on the console to run removepkg.
```

The procedure terminates.

4. If you entered the **removepkg** command without arguments, you will see a sorted listing of installed packages.

NOTE You may halt and continue the display using Ctrl s and Ctrl q, respectively.

You will see the following message:

```
Select a number (1-n) from this list to remove:
```

NOTE n is the number of packages currently installed.

Type the number of the package that you wish to remove and strike Enter.

5. If you correctly entered the **removepkg** command and the system located the package that you specified for removal, you will see the following message:

```
Confirm

Do you really want to remove
<package name>?

Strike ENTER when ready
or ESC to stop.
```

If the above message did not occur, go directly ahead to Step 8 to resolve the problem.

Strike [Enter] to remove the package.

If you decided not to remove this package or specified the wrong one, then strike the [Esc] key.

NOTE: If you strike anything other than [Enter] or [Esc], the system will beep.

6. If the system did not find a valid removal script for the package, you will see the following message:

```
Cannot find removal script for
<package name>. You will have
to remove this package manually
using UNIX System tools from
the UNIX System Shell.

The file /usr/lib/installed/Files/<installed filename>
contains a list of the files and
directories installed or created
by the package. You may wish to
use this file to help in removing
the package.
```

NOTE

The <installed filename> will be a *filename* of the form <package name>*filename*.

7. If the package is removed successfully, you will see the following message:

```
The <package name> package is now removed.
```

If the removal is unsuccessful, the removal process is halted, and steps 8 and 9 do not apply.

8. If the system found no optional add-on packages installed on your system, you will see the following message:

```
There are currently no software
applications installed that can be
removed.
```

9. If the system did not find the package name you supplied as an argument to **removepkg**, you will see the following message:

```
There is no software package
currently installed resembling:
<argument(s)>.
```

Installing and Removing XENIX Application Packages

UNIX System V/386 provides full support for applications written to run on the XENIX System V operating system. An application written for XENIX may be installed and run under UNIX System V/386 with no loss in functionality.

To install a XENIX application package, follow the directions that accompany the application. Most applications require that the person installing the package have *root* permissions.

To remove a XENIX package from the UNIX System V/386 filesystem, use the **rm** command (see *rm*(1) in the *User's/System Administrator's Reference Manual*).

NOTE

Some XENIX applications use the **custom** command during the installation and removal process. For compatibility, this command is included with UNIX System V/386. Based on the application, you may be able to use the **custom** command to install upgrades to the application, customize the application, or remove the application from the UNIX System.

Chapter 3: Using the UNIX System Shell

Commands

Since the target user is an experienced UNIX System user who is a programmer, you can skip this chapter. However, if you happen to be a novice UNIX System user, read this chapter and Chapter 7, Shell Tutorial, in the *User's Guide*.

The UNIX System shell serves as the interface between you and the computer. The shell accepts commands from you. In the UNIX System, a command (executable program) is a program that can be executed by the computer without a need for translation. Commands or requests to the shell are usually entered as a single line typed on the keyboard. This single line is called a command line. A command line is divided into two major parts:

- the program name
- the arguments

The first word of the command line is the name of the program to be executed. The other words on the line are referred to as arguments. Arguments are used to provide information required by the program. The command line looks like this:

command *argument argument argument ... ...* Enter.

Entering Commands

After receiving the prompt $ (# if root), you can type in your command line and strike [Enter]. When the shell prompt returns, the program is finished running, and you can enter another command line.

UNIX System commands are specified in lowercase letters. Enter all these commands in lowercase letters. If you enter a command in uppercase letters, the computer will not recognize it.

The files you create to execute at a later time are called executable files. These filenames can have uppercase or lowercase letters. When you invoke an executable file that includes uppercase letters, you should always type the filename just the way it appears.

If you make a mistake while typing in a command, just backspace (strike the [Backspace] key) to the beginning of the mistake and retype.

If you log in using uppercase letters, the system assumes that your terminal (console included) is not capable of handling lowercase. All input and output for the remainder of the session will be uppercase. If this happens, log off and make sure your [Caps/Lock] key is not engaged. Then log in again.

Using Control Characters

The control key is used in combination with other keyboard characters. These keys are used to initiate a controlling action such as backspacing or tabbing across a line. In addition, some control characters define UNIX System-specific commands, such as temporarily halting output from displaying on your screen.

The control key on your keyboard is labeled Ctrl.

You can type a control character by holding down the Ctrl key and then striking the appropriate alphabetic key. Control characters do not print on the screen when typed. In this guide, if you're instructed to strike Control s, the text will read "strike Ctrl s."

Let's take a look at some of the control character combinations you'll be using regularly when working with the UNIX System.

Stopping a Command

If you want to stop a command from executing on your console, simply strike [Del] or [Rubout]. You will receive the UNIX System prompt, indicating that the UNIX System is ready to accept your next command. You can do this before or after a command has started to execute.

Temporarily Stopping Output

At times, you may wish to temporarily stop the UNIX System from displaying output on your display screen. This could be the case when information is scrolling freely across your screen. If you type [Ctrl] and [s] simultaneously, the information on the screen will stop scrolling. When you type [Ctrl] and [q] simultaneously, the information will again start to scroll freely across the screen.

Running a Command in the Background

Within the UNIX System, you can run two or more programs or commands at one time. You can run commands or programs in the foreground (where you can see the results on the screen) and in the background (where you can't see the results on the screen). When a command is running, it is known as a process. Therefore, a foreground command is called a foreground process. An example of a foreground process is as follows:

spell *filename* [Enter]

The **spell** command will display on your screen any misspelled words in the file. If there are no misspellings, there will be no output, and the prompt will be returned.

You can run a command in the background (background process) by adding an ampersand (&) to the end of a command line before striking [Enter].

When the shell reads the **&**, the shell starts running the command in the background, displays an identification number, and displays the **$** prompt so you can continue to work in the foreground.

To save the output of the process you are running in the background, you must redirect the output into another file. Redirecting command output puts the results into a specified file so you can look at the file later. For example, if you want to run the **spell** command on a large file but want to look at it later, type the following:

spell *filename* > *newfile* **&** [Enter]

The shell first gives you an identification number and then the UNIX System window prompt. Then you can go ahead and run another command. When your process in the background is finished, the output will be contained in the specified file.

When a process is running in the background, it cannot be temporarily stopped with [Ctrl] [s] or halted with [Del] or [Rubout] like a foreground process. Only when you log off or terminate the background process will it stop. To stop the background process while you're still logged in, type **kill** *id_number* [Enter], where *id_number* is the identification number of the process.

To see if your process is running in the background, use the **ps** (report process status) command. After you type a **ps**, a list of the processes that are currently running is displayed. Look through these processes and find the identification number that was assigned to your background process. If you find it, the process is still running. If you do not find it, then the process has completed running.

For example, suppose you run **spell** in the background and your id number is 29570. Type the **ps** command to see if the process is running. The output will look something like the following:

```
  PID TTY      TIME COMMAND

29047 sxt002   0:03 sh
29570 sxt002   0:00 spell
29572 sxt002   0:00 tee
29573 sxt002   0:00 sh
29575 sxt002   0:03 sort
29577 sxt002   0:00 spellpro
29578 sxt002   0:02 sed
29579 sxt002   0:00 comm
29580 sxt002   0:01 ps
```

Notice the second line from the top matches your id number. That means your background process is still running. See the *ps*(1) command in the *User's/System Administrator's Reference Manual* for additional information.

Printing a File

To print a file, you first need a printer connected to your computer. The *User's Guide* provides information on using printer commands. To print a file, use the **lp** command as shown in the following example:

lp *filename* Enter

filename is the file you want to print. As mentioned before, use the **lpstat** command to see the print jobs queued on the printer. Refer to Chapter 4, "System Administration," for information on setting up your printer, and Chapter 7, "LP Print Service Administration," for information on printer administration.

Stopping a Session With Your Computer

To complete a session, log off: strike [Ctrl] and [d] at the same time or type **exit** and strike [Enter]. Either one of these commands will log you off the UNIX System.

An Introduction to UNIX System Commands

The UNIX System software you receive with your computer is called the Foundation Set. This software set contains the files necessary to provide the UNIX System on your computer with many useful utilities (programs) and commands.

A program is a set of instructions that the computer follows to do a specific job. In the UNIX System, programs that can be executed by the computer without the need for translation are called executable programs or commands.

These commands allow you to

- process text
- manage information
- communicate electronically
- use a productive programming and software development environment

When you first log in to your computer, you will receive a shell prompt. The shell prompt is a signal from the shell command interpreter that it is ready to accept your request. The command you wish to execute is typed in on the keyboard followed by striking the [Enter] key. When the shell receives [Enter], it considers the input (whatever it is) as a command. The shell searches one or more directories to locate the program you specified. When the program is found, the shell brings your request to the attention of the kernel. The kernel then follows the program's instructions and executes your request. After the program runs, the shell asks you for more information or tells you it is ready for your next command.

For an explanation of all the UNIX System commands, refer to the *User's/System Administrator's Reference Manual* and the *Programmer's Reference Manual*. Also, refer to the *User's Guide* for more information about the most commonly used UNIX System commands.

Basic File Operations

Your Foundation Set includes many commands. Some of the more common commands used for file and directory manipulation are included here:

- **cat** (concatenate): Displays the contents of one or more files on your screen.
- **cd** (change directory): Changes your current working directory to the directory you specify. If a directory is not specified, your current working directory will be your home directory.
- **chmod** (change mode): Changes the permission modes of a file or directory. See "Access Permissions" in Chapter 5, "Customizing Your Computer," for a discussion of permission modes.
- **cp** (copy): Copies the contents of a file to another specified file.
- **ls** (list): Lists the contents of one or more files or directories.
- **mkdir** (make directory): Creates a directory of a specified name.
- **mv** (move): Moves (renames) a file or directory.
- **rm** (remove file): Removes a specified file or files.
- **rmdir** (remove directory): Removes a specified directory or directories.

System Maintenance Commands

This group of commands is used for system administration. The following is a list of some of the more common system maintenance commands:

- **df** (free disk block report): Displays the number of free disk blocks and free i-nodes available for on-line file systems.
- **du** (disk usage summary): Displays the number of blocks contained in all files and directories.
- **fsck** (file system check): Runs a consistency check on a specified file system.
- **mount**: Mounts a file system.
- **ps** (process status report): Displays all active processes that are running on the UNIX System.
- **umount**: Unmounts a file system.
- **uname** (current UNIX System display): Displays the node name of the current UNIX System when used with the **-n** option. Refer to the "Setting Up Electronic Mail" section in Chapter 4, "System Administration," for information on naming your computer.

Text and Text Editing Commands

This group of commands is used to edit and manipulate text. The following is a list of these text and text editing commands:

- **vi**: Visual screen editor.
- **ed**: Text editor.
- **diff** (file comparator): Runs a file comparison on two files and displays the lines of text that are different.
- **grep** (pattern search): Searches for a pattern of text in one or more files.
- **spell**: Runs a spelling check on the specified files.

File Encryption Commands

This group of commands is used in protecting files. These encryption commands are packaged separately for use in the United States only. You'll have to install them after the initial UNIX System installation with the procedures given in the "Installing UNIX System Applications Software From Floppy Disk" section of Chapter 4, "System Administration." See Chapter 5, "Customizing Your Computer," to learn how to use the **crypt** command.

- **crypt**: Encrypts a file.
- **vi -x**: Edits a file (with the visual screen editor) where that file has already been encrypted. This is also used to create an encrypted file.
- **ed -x**: Edits a file (with the text editor) where that file has already been encrypted. This is also used to create an encrypted file.

Basic Networking Commands

This group of commands allows you to use Basic Networking on your computer. Refer to Chapter 8, "Basic Networking Administration," for a more detailed description of Basic Networking. The following is a list of some of the more common Basic Networking commands:

- **cu** (calls another UNIX System): Calls up another UNIX System or terminal.
- **mail** (sends or reads mail): Sends mail to someone on your UNIX System or another UNIX System and allows you to read mail sent by someone else.
- **uuto** (sends files): Sends files to a remote computer and places the files in *PUBDIR/receive/user/mysystem*, where *PUBDIR* is defined as */usr/spool/uucppublic*.
- **uupick** (retrieves files): "Picks up" files that are sent to a computer using **uuto**.
- **uux** (executes UNIX System-to-UNIX System commands): Gathers necessary files from a UNIX System, executes the command on a specified system, and sends the output to a file on a specified system.

Line Printer Spooler Commands

This group of commands is used for printer spooling on your computer. Refer to Chapter 7, "LP Print Service Administration," for a more detailed description of printer spooling. The following is a list of some of the more common printer spooling commands:

- **lp** (line printer): Arranges your file to be printed on a specified printer.
- **cancel**: Stops a print job from printing and removes that job from the printer queue when the print job id is entered.
- **enable**: Restarts a printer if it was stopped with the **disable** command. All print jobs in the printer queue will start printing again, in turn.
- **lpstat** (lp status information): Displays the information about the status of the LP line printer system.

What is a Manual Page

Manual pages describe computer commands. Each command has independently numbered manual pages describing that command in detail. The manual pages are arranged alphabetically and grouped into sections. Each section contains the commands for a specific function as follows:

- Section 1 — System Commands
- Section 1M — System Maintenance Commands
- Section 2 — System Calls
- Section 3 — Subroutines
- Section 4 — File Formats
- Section 5 — Miscellaneous Facilities
- Section 7 — Special Files

The *User's/System Administrator's Reference Manual* contains Sections 1, 1M, and 7. The *Programmer's Reference Manual* contains sections 2, 3, 4, and 5.

Chapter 4: System Administration

Introduction

UNIX System V/386 is a powerful multi-user, multi-tasking operating system. Careful control and regular maintenance of the operating system is necessary to keep it running smoothly. This careful control and regular maintenance of the operating system is called *system administration.*

Some of the tasks associated with system administration include:

- starting and stopping the UNIX System
- setting the system date and time
- managing user logins
- backing up and restoring files
- using the floppy disk drive
- setting up peripheral devices
- setting up electronic mail

Depending on the system size, number of users, and system load, the administrative tasks may need to be performed weekly or even daily. Regardless, system administration should be performed faithfully to prevent unnecessary loss of data, breach of security, or reduced system performance.

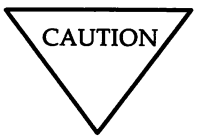
The System Administrator

In a multi-user, multi-tasking operating system such as UNIX System V/386, at least one person should be identified as the *system administrator* and be responsible for maintaining the system's integrity, security, and performance. The system administrator has privileges to execute special programs and access files restricted from ordinary system users. Restricting these powerful commands to a privileged user ensures a secure system with organized maintenance.

Many tasks presented in this chapter can be performed only by the system administrator. Other non-restricted tasks (such as floppy operations) are tools used by the system administrator but are also available to ordinary users. The tasks restricted to the system administrator are noted as such in the section describing the task.

The Superuser Login

The superuser login is a special login for use by the system administrator. It allows the system administrator access to any file regardless of the file permissions and to special commands used in performing administrative tasks.



Due to the powerful privileges associated with the superuser login, it should only be used when performing administrative tasks. Use an ordinary user login when simply working on the system.

The superuser login called **root** is present when the system is installed and is denoted by the user-id 0. This unique user-id is what provides the user of this login with special privileges.

Becoming the Superuser

There are two ways to become the superuser. The first is to log in at the console terminal with the **root** login and password. Any attempt to log in as **root** on a serial terminal will be denied.

The second way to become the superuser is with the **/bin/su** command. The **su** command lets you temporarily become the superuser while logged in as an ordinary user. You can use **su** at the console or a serial terminal if you know the **root** password.

Refer to the *User's/System Administrator's Reference Manual* for a complete description of the **su** command.

Maintaining the Superuser Login

Since the superuser login is extremely powerful, the following guidelines should be observed:

- Restrict the **root** password to those who really need it.
- Change the **root** password often.
- Use the **root** login only for system administration.
- Exercise extreme caution when using the **root** login.

Starting and Stopping the UNIX System

The UNIX Operating System is started with a *boot* procedure and stopped with a *shutdown* procedure. Special tasks are automatically performed each time the system is booted or shut down.

Boot Procedure

There are two ways to boot (start up) the UNIX System, depending on its current state. A cold boot is used when power has been removed from the machine, and a warm boot is used when the system has been shut down but not powered down. Both boot methods yield identical results.

A cold boot is performed simply by restoring power to the machine, and a warm boot is initiated by pressing the **CTRL-ALT-DEL** keys simultaneously. A warm boot will work only after the system has been shut down properly.

NOTE	Before booting the system, make sure the floppy disk drive is empty. Otherwise, the computer will attempt to find a bootable operating system on the inserted floppy diskette.

The system automatically performs the following tasks each time it is booted:

- Runs resident diagnostics to check your computer's hardware
- Starts the boot procedure and displays the message:

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

- Displays memory and copyright information
- Determines if the system was shut down properly
- Checks the file systems for inconsistencies and attempts to make any necessary repairs
- Starts various system processes such as **gettys**, **cron**, **lpscheds**, etc.

When the boot procedure is completed the login prompt is displayed, and the system is ready for use.

NOTE

Under some circumstances (i.e., a system crash), you might be prompted with a message asking if you want to save a *system dump*. A system dump is used by experienced system administrators to obtain detailed information about the state of the system when it crashed. Under most circumstances, you would answer "n" to this prompt.

Shutdown Procedure

The **shutdown** command is the normal method used to stop the system and should be used each time you stop the system. The **shutdown** command warns all current users that the system is being shut down. This gives users an opportunity to stop their current activities before the system is stopped, thus preventing loss of data. The **shutdown** command also performs various tasks to assure that the system is sane the next time it's booted.

The system should never be powered down without first issuing the **shutdown** command and waiting for it to complete.

CAUTION

If the system is powered down without being properly shut down, data loss can occur.

To use the **shutdown** command, you must be logged in as **root** at the console terminal and be in the root directory. The following is an example of a **shutdown** procedure.

1. Log in as **root** at the console.
2. Enter **cd /** to ensure that you are in the root directory.

3. Enter **shutdown** to initiate the shutdown procedure.

   The **shutdown** command will inform all users that the system is being shut down and will prompt you with the following continue message:

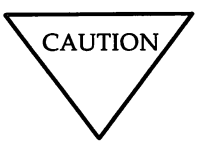
   ```
   Do you want to continue? (y or n):
   ```

   If you answer *n*, the shutdown procedure will abort, and all the users will be notified that the system will not be shut down. If you answer *y*, the shutdown procedure will proceed and additional messages will appear on the console.

   When the following message appears on the console, the system shutdown is completed, and it is safe to power down or reboot using **CTRL-ALT-DEL**:

   ```
   The system is down.

   Reboot system now.
   ```



Do not attempt to power down or reboot the system until the above completion message is displayed.

Refer to the *User's/System Administrator's Reference Manual* for a complete description of the **shutdown** command.

Setting System Date and Time

The system clock can be changed using the **date** command. Although several options of this command can be used by ordinary users, only the superuser can use the command to change the system date and time. To set the system date and time:

1. Log in as **root**.
2. Enter **date** *MMddhhmmyy*, where:

   MM = month (01 for January, 12 for December)

   dd = day of month

   hh = hour of day (24-hour clock: 00 for midnight, 23 for 11:00 p.m.)

   mm = minute of hour

   yy = year (88 for 1988).

For example, to set the date to January 1, 1988 and the time to 1:30 p.m., you would enter:

```
# date 0101133088
```

Refer to the *User's/System Administrator's Reference Manual* for a complete description of the **date** command.

Managing User Logins

Access to your system is restricted to those people (or other computers) assigned logins and passwords. This enables the system administrator to keep track of those using the system, control access to the system, and control system resources. Other computers generally use generic logins such as **mail** or **nuucp** to log in on your computer and exchange information. However, all users should use a private login.

A user login consists of the following information stored in the */etc/passwd* file:

- Login-ID: This is the name used to log in to the system. The 1-8 character Login-ID must be unique for each user.
- Password: This is a secret code used when logging in to the system.
- User-ID: This is a number used by the system to associate files and processes to a login ID. The User-ID is usually unique for each user login.
- User-Name: This is the real name of the user (i.e., John Doe, Jane Smith, etc.).
- Group-ID: This is a number used by the system to associate files and processes to a group of users.
- Home Directory: This is the directory where the user is automatically placed when logging in. It is the user's own personal part of the file system and is where private files are stored. The Home Directory must also be unique for each user login.

It is the responsibility of the system administrator to maintain the user logins. Maintenance of user logins includes adding users, aging user passwords, changing user passwords, and removing user logins no longer needed. You must be the superuser to administer user logins.

Adding a User

A new user login is created with the **adduser** command. This command performs several tasks such as updating the */etc/passwd* file, creating a home directory, creating a *.profile* and assigning an initial password.

The **adduser** command syntax is as follows:

adduser *Login-ID User-Name User-ID Home_Directory*

For example, to add a login for John Doe with *Login-ID* assigned to **hotshot**, the following command would be used:

```
# adduser hotshot "John Doe" 200 /usr/hotshot
```

Remember that *Login-ID*, *User-ID* and *Home_Directory* must be unique for each user. The *User-Name* parameter is the only parameter that can contain multiple words. If you choose to assign a multiple word *User-Name*, the entire *User-Name* parameter must be enclosed in quotes as shown in the example.

If the new *Login-ID* is not unique, the following error message will be displayed:

```
<Login-ID> already exists on your system.

Choose another login name.
```

Similar messages will be displayed if the *User-ID* or the *Home_Directory* is not unique. If this happens, try the command again using different values.

If the command arguments were acceptable, the system prompts you for an initial password for the user:

```
Enter password for login:
New password:
```

When this prompt appears, type in an initial password for the new user login. Since the password is secret, it will not appear while you are typing. After you have entered the password, strike ENTER and the following confirmation prompt will appear:

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

You must then enter the initial password exactly as you entered it in the above step. Again, the password will not be displayed. If the two passwords do not match, you will be prompted to re-enter the password and reconfirm it. If they match, then the new login will be added to the system. You can now tell the new user the *Login-ID* and temporary password used to access the system.

NOTE

The system administrator should advise a new user to change the temporary password to their own personal password upon logging in. Refer to "Changing User Passwords" in this chapter.

Refer to the *User's/System Administrator's Reference Manual* for a complete description of the **adduser** command.

Password Guidelines

To log in to the system, both the *Login-ID* and password must be used. Since the *Login-IDs* are known to other users on the system, the password is the key to system security. Therefore, the following guidelines should be observed when assigning passwords:

- Choose a password that is hard to guess. Do not use names, birthdays, telephone numbers, etc.
- Passwords must be at least six characters long with at least two alphabetic characters and one non-alphabetic character.
- Never record logins and passwords on paper.
- Emphasize that user passwords must be kept secret.

Aging User Passwords

Since passwords are considered to be the key to a system's security, the system administrator (**root**) can force users to periodically change their password to ensure its secrecy. This is called password aging and can be set using the **passwd** command.

The syntax of the **passwd** command used to set password aging is as follows:

passwd -l -x*max* **-n***min name*

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

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

For example, to force user **hotshot** to change his/her password every 30 days and to keep any assigned password for at least a week, you would enter

```
# passwd -l -x30 -n7 hotshot
```

Refer to the *User's/System Administrator's Reference Manual* for a complete description of the **passwd** command.

Changing User Passwords

An ordinary user can change his/her password any time (depending on password aging) using the **passwd** command. To change a password, you would enter

$ passwd *user*

where *user* is the associated login ID. If *user* is not specified, the system assumes you are changing the password for the current login. The system asks for the old password before allowing it to be changed. It then asks for the new password twice to confirm that it was typed correctly. The superuser can change passwords for any login on the system without knowing the old password.

Refer to the *User's/System Administrator's Reference Manual* for a complete description of the **passwd** command.

Removing User Logins

From time to time, it may be necessary to remove user logins. The **deluser** command is used to remove user logins from the system. Only the superuser can use the **deluser** command.

The **deluser** command syntax is as follows:

deluser *Login-ID Save-flag Home_Directory*

With the exception of the *Save-flag* argument, the other arguments are the same as those used with the **adduser** command.

Save-flag designates whether or not files in the user's home directory are saved before the login is removed. This can be useful if the user has work to be completed by another user. If *Save-flag* is **Yes**, then all files under the user's *Home_Directory* are moved to the */lost+found/<Login-ID>* directory. If set to **No**, the files are removed from the system.

For example, if you want to remove the user added in the **adduser** example, the following command would be used:

```
# deluser hotshot Yes /usr/hotshot
```

This would save all files in the */usr/hotshot* directory to */lost+found/hotshot* and remove the **hotshot** login.

Refer to the *User's/System Administrator's Reference Manual* for a complete description of the **deluser** command.

Backing Up and Restoring Files

To minimize the loss of data caused by hardware failures or operator error, files on the hard disk should be copied to a removable media (cartridge tape or floppy diskette) on a regular basis. The copying of files to removable media is called a *backup*.

After performing a backup, the removable media should be kept in a secure place in case of any future hardware failure or operator error. If such a failure or error should occur, there will always be an up-to-date copy of the files that reside on the hard disk. If the system should become corrupted or a file is mistakenly removed from the hard disk, the files on the removable media can be copied from the removable media to the hard disk. The procedure of copying the backup files from the removable media to the hard disk is called a *restore*.

Backing Up Files

It is recommended that files on the hard disk be backed up to a removable media on a regular basis. Practicing frequent backups will help minimize data loss due to hardware failure or operator error.

Backups are performed with the **backup** command. The **backup** command is very flexible and can be used to perform complete, incremental, or user backups.

The following is an example of using the **backup** command to back up all files that have been changed since the system was installed. This example will back up the files to 1.2 MB floppy diskette(s). The **backup** command will estimate the number of diskettes necessary to complete the backup. Each floppy must be formatted before using them as backup media (see "Formatting Floppy Diskettes" in this chapter).

1. Log in as **root**.
2. Insert a formatted 1.2 MB floppy diskette into the drive.
3. Enter the following command:

```
# backup -c -d "/dev/rdsk/f0q15dt"
```

4. Label the backup diskettes with the date of the backup and the command used to perform the backup.

Since it is recommended to perform backups frequently, it may be desirable to create a shell script containing the commands necessary to perform the backup. In this case, the shell script would contain only one command. The following example is a shell script called **sysbackup**:

```
# Sample backup shell script

backup -c -d "/dev/rdsk/f0q15dt"
```

NOTE

This shell script should be created in the */usr/lbin* directory for convenience. The shell script must also be set to execution mode with the **chmod** command, described in the *User's/System Administration Reference Manual.*

Refer to the *User's/System Administrator's Reference Manual* for a complete description of the **backup** command and its options.

Restoring Files

If you experience any loss of data or data corruption due to a hardware failure or operator error, you can recover the lost or corrupted data from backup diskettes, provided it was backed up. Restoring lost or corrupted data can be performed directly from the UNIX System shell prompt.

NOTE

Remember that only those files backed up to removable media can be restored from the removable media, and any modifications made to the files since the last backup will not be applied to the restored files.

Backup diskettes can be restored using the **restore** command, provided you know the options used to create the backup diskettes. For instance, if the backup diskettes were created using

```
backup -c -d "/dev/rdsk/f0q15dt"
```

then the appropriate command to restore the backup diskette would be

```
restore -c -o -d "/dev/rdsk/f0q15dt"
```

Refer to the *User's/System Administrator's Reference Manual* for a complete description of the **restore** command.

NOTE

UNIX System V/386 includes the **xrestore** command, which restores backups that were made under the XENIX Operating System. Refer to the **xrestore** command in the *User's/System Administrator's Reference Manual* for information on restoring XENIX filesystem backups.

Using the Floppy Disk Drive

Utilities are provided with the UNIX System to read and write data to and from floppy diskettes. These utilities allow you to perform the following basic floppy diskette operations:

- format floppy diskettes
- copy files from the hard disk to floppy diskettes
- copy files from floppy diskettes to the hard disk
- duplicate floppy diskettes

UNIX System V/386 provides access to several types of floppy diskettes. A floppy diskette is accessed through either UNIX System block or character (raw) device names. The following table presents the types of floppy diskettes supported by your operating system and their device names.

Diskette Type	Block Device	Character Device
1.2 MB DSHD	/dev/dsk/f0q15dt	/dev/rdsk/f0q15dt
360 KB DSDD	/dev/dsk/f0d9dt	/dev/rdsk/f0d9dt
AT&T UNIX PC	/dev/dsk/f0d8dt	/dev/rdsk/f0d8dt
AT&T 3B2 (720KB)	/dev/dsk/f05qt	/dev/rdsk/f05qt
3.5" (1.44MB)	/dev/dsk/f03ht	/dev/rdsk/f03ht
3.5" (720KB)	/dev/dsk/f03dt	/dev/rdsk/f03dt

Formatting Floppy Diskettes

Before a floppy diskette can be used in any floppy diskette operation, it must be formatted with the **format** command. To format a floppy diskette:

1. Determine the diskette type.
2. Insert the diskette into the drive.
3. Enter the **format** command followed by the appropriate character device name for that type of diskette.

The following command will format a 1.2 MB double-sided, high-density (DSHD) diskette:

```
$ format /dev/rdsk/f0q15dt
```

Copying Files to Floppy Diskette

Several methods can be used to copy files to a floppy diskette. One method is to format a floppy diskette, create a file system on the disk, mount the file system, and access the diskette as part of the UNIX System directory structure. This might seem a bit complicated, but it is a very convenient way to create a portable file system. This process is described in Chapter 6, "All About File Systems."

More simple methods involve using the **cpio** command with other commands, such as **find** or **ls**, or by using the **tar** command.

Copying Files Using cpio

The **cpio** command copies files to and from floppy diskettes by taking a list of files from standard input and creating a file archive that can be redirected to a floppy diskette. Several commands can be used to supply the file names to be archived by using the UNIX System V shell pipe facilities. For example, to copy all the files in the current directory to a 1.2 MB floppy diskette:

1. Insert a formatted 1.2 MB floppy diskette into the drive.
2. Change directory to the directory containing the files to be copied.
3. Enter the command line:

```
$ ls | cpio -oc > /dev/dsk/f0q15dt
```

In this example, **ls** will list the files in the current directory and pipe the list to **cpio**. Then **cpio** will archive them and redirect the output to the floppy drive.

Similarly, any files copied onto a floppy diskette using the **cpio** command can be copied back to the hard disk using different **cpio** options. The following example will copy the files copied to a floppy diskette in the above **cpio** example back to the hard disk into the current directory.

1. Insert the floppy diskette containing the **cpio** archive into the drive.
2. Enter the following **cpio** command:

```
$ cpio -ic < /dev/dsk/f0q15dt
```

Refer to the *User's/System Administrator's Reference Manual* for a complete description of the **cpio** command.

Copying Files Using the tar Command

As an alternative to **cpio**, you can use the **tar** command to transfer files to and from floppy diskettes. The following example will copy files from the hard disk onto a 1.2 MB floppy diskette:

1. Insert a formatted 1.2 MB floppy diskette into the drive.
2. Change directories to the directory containing the files to be copied.
3. Enter the following command:

```
# tar -cf /dev/dsk/f0q15dt
```

These files can then be copied back to the hard disk by using the **x** option to **tar** as follows:

1. Insert the floppy diskette containing the **tar** archive into the drive.
2. Change directories to the destination directory where you want to copy the files.
3. Enter the following command:

```
$ tar -xf /dev/dsk/f0q15dt
```

Refer to the *User's/System Administrator's Reference Manual* for a complete description of the **tar** command.

Copying Diskettes

From time to time, it will be necessary to make backup copies of diskettes. This is a two-step process using the **dd** command. The first step is to copy the data from the source diskette to the hard disk. The second step is to copy the data from the hard disk to the formatted destination diskette.

The following example shows how to make a copy of a 1.2 MB floppy diskette:

1. Insert the source diskette into the disk drive.
2. Copy the floppy diskette onto the hard disk using the **dd** command as follows:

```
$ dd if=/dev/dsk/f0q15dt of=/tmp/tmpflop
```

3. Remove the source floppy diskette, and insert a formatted 1.2 MB floppy diskette.
4. Copy the temporary image of the original floppy diskette onto the destination floppy diskette as follows:

```
$ dd if=/tmp/tmpflop of=/dev/dsk/f0q15dt
```

Refer to the *User's/System Administrator's Reference Manual* for a complete description of the **dd** command.

Setting Up Peripheral Devices

The system administrator can add peripheral devices such as terminals, line printers, or a second hard disk to the system. These peripheral devices can increase the number of users or disk space, or expand the overall capabilities of the system.

Most computers supported by UNIX System V/386 are configured with one serial (RS-232) port and one parallel (CENTRONICS) interface port. While the serial port can be used to connect any RS-232 device to your computer, the parallel port can only be used with printers. Typical RS-232 devices include terminals, modems, line printers, and other computers.

This section explains how to add RS-232 devices, parallel line printers and a second hard disk to your system.

Setting Up an RS-232 Connection

When connecting an RS-232 device to your system, a connection must be made from your computer to the device either directly (hard-wired) or through a modem. Once the connection is made, the system must know what type of connection exists and what type of device is on the other side of that link.

The following sections provide information on making the physical RS-232 connection and explain how to configure the RS-232 port. In addition, connecting line printers to the serial port will be discussed.

RS-232 Direct Connection

The direct, hard-wired connection between your computer and an RS-232 device is dependent upon the type of device. The specific type of cable and pin assignments will be explained in the device's hardware manual. However, the following hints generally apply when connecting an RS-232 device to your computer:

- RS-232 cables should be less than 50 feet long. Cables longer than 50 feet may introduce line noise. In some cases, you can use a longer cable by reducing the line speed (baud rate).
- An in-line adapter called a null modem adapter is usually required when connecting a terminal or computer directly to the serial port on your computer.

- A null modem should not be used when connecting a modem to your computer.

Configuring a Terminal

When connecting a terminal directly to the serial port, you must configure the port to work with a terminal. This involves adding the appropriate entries to the */etc/inittab* file. The */etc/inittab* file contains instructions that control the system initialization process, **init**. The **init** process tells the system what to do with each of the computer's devices, including ports, at the end of the system boot process.

The format of */etc/inittab* entries is

```
id:rstate:action:process
```

where:

- **id** is a unique identifier for the *inittab* entry. This identifier may be up to four characters long.
- **rstate** defines the run-level in which this entry is to be processed. The **rstate** may range from 0-6 but is usually 2 and/or 3 multi-user states. Multiple run-levels (rstate) can be specified.
- **action** tells **init** how to treat the process. There are several keywords that can be used in this field. For setting up a terminal, the keyword **respawn** is used. This keyword tells **init** to restart the process any time it terminates.
- **process** is the process that **init** will start if the system run-level matches one of the entry's **rstates**. When setting up a terminal, the **getty** process must be executed with the desired flags. The **getty** process is what displays the login prompt on your screen and allows you access to the UNIX System.

To add a terminal to the serial port, an entry similar to the following must be added to the */etc/inittab* file:

```
I00:23:respawn:/etc/getty tty00 1200
```

where:

- **I00** is the unique ID given to the entry. The label **I00** is given as an example. The label can be any unique character string up to four characters.

- **23** is the run-level(s) in which the terminal connected to tty00 will receive a **getty** and can be used to access the system. In this example, the terminal will be enabled for run-levels 2 and 3.
- **respawn** means that the process will be respawned each time it terminates. In this case, a *getty* will be restarted when a user logs off the terminal, hence initializing the terminal for the next session.
- **/etc/getty tty00 1200** will start a getty process on **/dev/tty00** (the first serial port) with line speed set to 1200 baud. The **/etc/getty** process will set the port attributes such as line speed, parity, etc. These attributes are stored in a file called */etc/gettydefs*. After the port attributes are set, **/etc/getty** will issue the login prompt (also stored in *gettydefs*) to the terminal.

After this entry is added to the */etc/inittab* file, the **init** process must be notified to re-examine its instructions. To force **init** to do this, the following command must be issued:

```
# init q
```

If the */etc/inittab* entry and the physical connection were made correctly, a login prompt should appear on the terminal. If the login prompt does not appear or if garbled characters appear on the terminal, the terminal speed may not match the speed in the */etc/inittab* file. You should refer to your hardware manual for setting the terminal's baud rate and either set it to match the */etc/inittab* entry or modify the */etc/inittab* entry to match the terminal. Then notify **init** to re-examine its instructions.

Refer to Section 4 of the *Programmer's Reference Manual* for complete descriptions of the */etc/inittab* and */etc/gettydefs* files. Also, refer to the *User's/System Administrator's Reference Manual* for a description of the **init** and **getty** processes.

Configuring a Modem

Modems can be used to connect your computer to remote terminals and computers using ordinary telephone lines. Terminals attached to a modem can dial into your system from a remote location, and UNIX System machines with a modem can use **uucp** to exchange information with your computer. Configuring your system to support a modem is similar to configuring a terminal, but more complex. The similarity is that an entry must be made in the */etc/inittab* file. However, the entry could differ, depending on the intended use of the modem.

A modem can be configured to support either single-directional or bi-directional communications. Single-directional communication means that only incoming or outgoing calls are allowed through the modem, while bi-directional communications allow your system to receive incoming and initiate outgoing calls using the modem. Therefore, you must determine the desired type of communications.

NOTE Your modem must have auto-dial capability if you want outgoing or bi-directional communications.

If only incoming communication is desired, the calling terminal or computer must be issued a **getty** process. Therefore, an entry similar to the following would be added to */etc/inittab*:

```
I00:23:respawn:/etc/getty tty00 1200
```

Single-directional, outgoing communication requires that neither *getty* nor *uugetty* be attached to the port. In this case, the action field should be set to off instead of respawn.

The */etc/inittab* entry for bi-directional communication must be different since **getty** supports only incoming communication. Instead, the **uugetty** process is used to control the communication. The **uugetty** process allows incoming calls to be answered, and if the port is not busy, outgoing calls to be initiated.

The following */etc/inittab* entry will enable the port to support bi-directional communication:

```
I00:23:respawn:/usr/lib/uucp/uugetty -r -t60 tty00 1200
```

NOTE The line (tty00) and line speed (1200) arguments to **uugetty** are used the same as with **getty**.

This *inittab* entry will issue a **uugetty** process to the serial port, enabling bi-directional communication through the modem.

In addition to the */etc/inittab* entry, the */usr/lib/uucp/Devices* file must be updated to configure the system to support modem dialing. A complete description of the */usr/lib/uucp/Devices* file is presented in Chapter 8, "Basic Networking Administration."

Refer to the *User's/System Administrator's Reference Manual* for a complete description of the **uugetty** process.

Configuring Other Directly Connected Devices

Other devices, such as another computer, can be connected to your computer using an RS-232 cable. Configuring your system to support such a connection is similar to configuring your system to support a modem. The same type of entry must be made in the */etc/inittab* file to allow bi-directional or single-directional communication.

The difference lies in the */usr/lib/uucp/Devices* file entry. Chapter 8, "Basic Networking Administration," describes the */usr/lib/uucp/Devices* file and the proper entries for directly connected devices.

Configuring a Serial Line Printer

Serial line printers can be attached to the system by using an RS-232 cable. To add a serial line printer to your system, you must first stop any **getty** or **uugetty** that may be running on the port. This is because the printer is an output device and the communication is one way, outgoing.

To stop **getty** or **uugetty**, change the action field of the port's **/etc/inittab** entry from respawn to off.

After the **getty** or **uugetty** is removed from the line, you must change the owner of the device node to which the line printer is attached to **lp**. For example, if you are connecting a line printer to */dev/tty00*, you should execute the following command to change the owner to **lp**:

```
# chown lp /dev/tty00
```

Then notify the **init** process to re-examine the */etc/inittab* file by entering **init q**. Upon re-examination, **init** will free the port for use with a line printer.

Once you are certain that the port is not being used by another device, make the physical connection and configure the system to recognize the line printer. Refer to Chapter 7, "LP Print Service Administration," for instructions on configuring the system to use the line printer.

Setting up a Parallel Line Printer

To set up a parallel line printer, you must connect the printer to your parallel port using a parallel (CENTRONICS) cable. Then refer to Chapter 7, "LP Print Service Administration," for instructions on configuring the system to use a parallel line printer.

Adding a Second Hard Disk

A second hard disk can be added to the system to provide additional disk space. A utility called **diskadd** is provided to configure your system with the second hard disk. This utility will prompt the system administrator for information about the disk setup and allow for specification of known media defects.

NOTE	To use a UNIX System partition created on the second hard disk, that partition must be made active. This does not mean that the computer will boot from this partition.

Refer to the *User's/System Administrator's Reference Manual* for a complete description of the **diskadd** command.

Setting up Electronic Mail

Your UNIX System V/386 has the ability to exchange electronic mail and files with other computers. Before mail can be exchanged, the following tasks must be performed:

- Establish a physical connection between the computers. This connection can be established using modems or direct connections.
- Assign a system name to your computer.
- Assign a password to the **mail** or **nuucp** login for other computers to use when sending mail to your computer. This will ensure that only friendly computers can log into your system.
- Provide system administrators for computers with which you will be exchanging mail, your system name, login and password, data telephone number and communication attributes.
- Obtain the same type of information from other system administrators concerning their system name, login and password, telephone number and attributes.

Setting up the Communication Line

Before sending mail to or receiving mail from other computers, you must configure the serial port for a modem or direct connection. Determine the type of communication you will be using, then refer to the "Setting up Peripheral Devices" section for instructions on configuring the communication line.

Assigning a System Name

Your system name can be up to eight characters long and must be different from the system names of the other computers with which you will be exchanging mail. The **uname** command is used to set the system name. This command can be executed by the superuser only. The following example will assign your system the name **mysys**:

```
# uname -S mysys
```

Refer to the *User's/System Administrator's Reference Manual* for a complete description of the **uname** command.

Assigning a Mail Login

Typically, system administrators use a login called **mail** or **nuucp** for exchanging mail. If you want to use the **nuucp** or **mail** login, you must use the **adduser** utility to add the desired login. You must then edit the */etc/passwd* file and change the **nuucp** or **mail** entry to use */usr/lib/uucp/uucico* as the program to execute as its shell instead of */bin/sh*.

Refer to Section 4 of the *Programmer's Reference Manual* for a complete description of the */etc/passwd* file.

Adding Other Computers

The **uucp** utility is used to exchange electronic mail with other computers. This utility needs the following information about the systems with which it will be exchanging information:

- the system name
- the **mail** or **nuucp** login and password
- the telephone number (if modems are to be used)
- communication attributes
- days and times the system is available to exchange information

After obtaining the above information, refer to Chapter 8, "Basic Networking Administration," to set up **uucp** on your system.

Chapter 5: Customizing Your Computer

Tailoring Your Environment

This chapter describes how to tailor or customize your computer environment to perform the duties unique to you and the other people who use it.

The System Default Profile

When you first log in, the UNIX System establishes your working environment as defined by the default system *profile*. The default system *profile*, located in the */etc* directory (*/etc/profile*), contains the commands needed to initialize your environment and commands common to all users. You must be logged in as **root** to modify the default system *profile*. However, the *.profile* (located in your login directory) contains the commands specific to you that set up your own environment. The variable MAIL, PATH, T2, erase character, etc, are set up here. So, your *.profile* is used to tailor your environment to your needs. For example, if you frequently log on with a specific remote terminal, you can set TERM to that terminal. You can use your own *.profile* (just like a regular file) to include other initialization commands.

After creating or making changes to the *.profile*, you can initiate the changes without logging off and logging back in again by typing the following:

```
.profile Enter
```

The shell will reinitialize your environment. The dot (**.**) is a special shell command used to execute commands in *.profile*.

The following is an example of a *.profile*. The lines that start with a # sign are comment lines. The comment lines are only in this example to explain what some of the commands are.

```
# Set command search path.
PATH=$PATH:$HOME/bin:
# Set the shell prompt to something other than $.
PS1=prompt
# Have mail printed when you log in.
mail
# Output the system date and time.
date
```

During login, the */etc/profile* file will be read to initialize your environment, and then your *.profile* will be read to execute any other commands you may have specified.

Some commands are executed based on the variables defined in the default environment. The following is a list of those variables:

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

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

HOME Defines pathname of your login directory. The value of **HOME** is set when you log in and should not be changed. **HOME** is specified in */etc/passwd*.

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

IFS Defines the internal field separator characters. The shell initially sets these characters to include the space (blank), tab, and new-line characters.

MAIL Defines the full pathname where you will receive mail from other users. **MAIL** is usually kept in */usr/mail* (i.e., **/usr/mail/***LOGNAME*).

PATH	Defines the directory search path for commands. By default, this variable includes the current directory, the **/bin** directory, and the **/usr/bin** directory.
PS1	Defines the primary shell prompt. By default, the shell prompt is set to **$** for regular UNIX System users and **#** for users that log in as **root**.
PS2	Defines the secondary shell prompt. By default, the secondary shell prompt is set to **>**. This prompt means that additional information (input) is needed for the command to run.
TERM	Defines your terminal type for certain programs (such as screen editors). By default, **TERM** is set to **AT386**.
TZ	Defines the time zone.

The following are a few hints to use when you're working with a *.profile*:

- To change a variable, type the following:

  variable=new_variable Enter

  export *variable* Enter

  For example, to change your secondary shell prompt (PS2) to **MORE**, type the following:

  PS2=MORE Enter

  export PS2 Enter

- To look at the variables in your environment, type the following:

  env Enter

- To reinitialize your environment, type the following:

  Enter.

Changing the News

News is a type of news brief displayed to users as they log in. The **news** directory is located in */usr/news* and contains news files for people to read. There is one **news** file per news item. To add or change **news**, create a file in the **news** directory and enter the news you want system users to read. To read the news after logging in, just type **news** Enter. See *news*(1) in the *User's/System Administrator's Reference Manual.*

Changing Message of the Day

The message of the day is a brief item of information displayed to all users as they log in. If you are going to update the message of the day, you must use the **root** login. The message is contained in the */etc/motd* file and can be changed using any text editor. You should try to keep the message of the day relatively short and to the point. Save your longer messages for the */usr/news* file(s).

Changing the Path

You can add your own directory of commands to your path by adding the following to your *.profile*:

PATH=$PATH:$HOME/*bin*; export PATH

where *bin* is the name of your command directory.

Automatic Program Execution

The UNIX System allows you to have programs run automatically at specified times. This is done with the **cron** program. The **cron** program and, more specifically, the **crontab** command allow you to run programs during off-hours such as

- file system administration
- long-running, user-written shell procedures
- cleanup procedures

Any task that needs to be done repeatedly at a specified time is a candidate for your *cron* file located in the */usr/spool/cron/crontabs* directory. You can use the **crontab** command to establish the entries you want.

The **crontab** command is used as follows:

crontab *file* Enter

crontab -r Enter

crontab -l Enter

The **crontab** command copies the specified *file* or standard input if no file is specified into a directory that holds all users' crontabs. The **-r** option removes a user's crontab from the *crontab* directory. The **-l** option will list the *crontab* file for the invoking user. See the *crontab*(1) command in the *User's/System Administrator's Reference Manual* for additional information.

Each line in the *crontab* file defines one procedure. The line entry format looks like the following:

minute hour day month day-of-week command

Each field is defined as follows:

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

The following rules apply to the first five fields:

- Two numbers separated by a hyphen indicate a range of numbers between the two specified numbers.

- A list of numbers separated by commas indicates only the numbers listed will be used.
- An asterisk specifies all legal values.

For example, 0 0 1,14 * 2 indicates a command will be run on the first and fourteenth of each month, as well as on every Tuesday. If a percent sign (%) is placed in the command field (sixth field), the UNIX System will translate it as a new-line character. Only the first line of a command field (character string up to the percent sign) is executed by the shell. Any other lines are made available to the command as standard input.

For example, let a file called *anyfile* contain the following **cron** entry:

```
0 0 1 * * mailx $LOGNAME % Subject: Call Mom! % now
```

When the command line **crontab** *anyfile* [Enter] is executed, the user whose login is **$LOGNAME** will get a reminder mail message with `Call Mom!` as the subject the first of every month.

Automatic System Cleanup

The UNIX System has to be cleaned up occasionally. Fortunately, you can get out of some cleaning with the help of the **crontab** command and the *crontab* file. You can specify cleanup jobs (e.g., remove aged files) and the time you want them to execute in the *crontab* file.

Your computer comes with some default cleanup procedures already defined. These cleanup procedures are done by the **root** login under the control of **crontab** each Sunday morning at 5:17. The file */etc/cleanup* defines what cleanup procedures are done.

Some of the files cleaned up each Sunday morning are as follows:

- */etc/wtmp*: This file contains a history of system logins. Every time a user logs in, a record is made in this file. As you can see, the size of this file grows forever, and it needs to be limited. Instead of deleting the contents of this file yourself, you can have **cron** do it for you.

- */usr/adm/sulog*: This file contains a history of users that use the **su** command to switch logins. As a security measure, this file should not be readable by other users. See the *su*(1) manual pages in the *User's/System Administrator's Reference Manual* for additional information.
- */usr/adm/cronlog*: This file contains a history of all actions taken by **cron**.

By logging in as **root** and executing **crontab -l**, you will see the **crontab** entry that executes */etc/cleanup* as well as other cleanup routines for UUCP (Basic Networking). By examining */etc/cleanup*, you will see the default routines done each Sunday morning. You can edit */etc/cleanup* and modify the **root crontab** (the **root crontab** is stored in */usr/spool/cron/crontabs*) to do cleanup jobs differently if you wish.

Customizing the Boot Process

The UNIX System **boot** program interactively loads and executes stand-alone UNIX System programs. While **boot** is used primarily for loading and executing the UNIX System kernel, it can load and execute any other programs that are linked for stand-alone execution. The system invokes the **boot** program each time the computer is started. It tries to locate the **boot** program on the floppy disk drive first; if the floppy disk drive is empty, the system invokes the hard-disk boot procedure.

When first invoked, **boot** displays the following status message:

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

The next step depends on whether you want to load the default program or load a different stand-alone program.

Loading the Default Program

To instruct **boot** to use the default program (kernel) and values specified in the boot default file, */etc/default/boot*, press RETURN. If you have just loaded the **boot** program from the distribution diskette, press RETURN so **boot** will use the default values. If you press any key other than RETURN, **boot** pauses and prompts you for custom information. For more information about */etc/default/boot* and providing custom boot information, see *boot*(1M)

in the *User's/System Administrator's Reference Manual.*

Loading a Different Stand-Alone Program

To load a program that is not the default program, press any key (except RETURN) at the "Booting the UNIX System" prompt to interrupt **boot**. The **boot** program pauses and prompts you with the following message for the name of the program you want to load:

```
Enter the name of a UNIX System kernel
to boot (e.g., /unix):
```

The system waits at this point for you to type the name of the program you want to load and press RETURN.

To load a program other than the **boot** program on the distribution diskette, you must specify the location of the program by providing a filename (if the program you want to load is on the default boot device), or by providing a device name and a filename (if the program you want to load is not on the default device). The filename must include the full pathname of the file containing the stand-alone program. The location of the program you want to load must be first on the command line and must be present if other **boot** options are specified either on the command line or in */etc/default/boot*. To indicate a program other than the **boot** program on the distribution diskette, use one of the following two formats:

filename or *xx(m,o)filename*

where

— *filename* is the standard UNIX System pathname. The *filename* argument must start with a slash if the program is not in the *root* directory. If *filename* is the only argument typed at the boot prompt, **boot** looks for the *filename* on the default boot device and tries to boot from it.

— *xx* is the device name (*hd* for the hard disk or *fd* for the floppy diskette device).

— *m* is the minor device number (*1* for the *root* file system on the hard disk).

— *o* is the offset in the partition (usually *0*).

Note that all numbers are in decimal. For minor device numbers of these devices, see *hd*(7) and *fd*(7) in the *User's/System Administrator's Reference Manual.*

Booting Automatically

You can set the boot process up to be automatic. To set up **boot** to run automatically, using the default configuration information in the */etc/default/boot* file, set **AUTOBOOT** to **YES** in the */etc/default/boot* file on the default *root* file system. This causes **boot** to display the default boot message and load the program. If an error occurs or a key is pressed during this automatic boot process, **boot** returns to the boot prompt and tries to load the program again. The **boot** program on the UNIX System installation diskette performs this automatic boot procedure.

If **AUTOBOOT** is set to **NO** in the */etc/default/boot* file on the default *root* file system, **boot** displays the "Booting the UNIX System kernel" message and gives you an opportunity to specify custom boot information before it begins loading the program. If you do not type something at the prompt, **boot** assumes you want the default configuration. At this point, **boot** behaves as though **AUTOBOOT** is set to **YES** in the */etc/default/boot* file. The **boot** program reads the configuration in the */etc/default/boot* file, then displays the default boot message (specified with the **BOOTMSG** option) and begins loading the program. For detailed descriptions of the **boot** options and procedures, see *boot*(1M) in the *User's/System Administrator's Reference Manual.*

Administering System Security

Security should be given special consideration on a multi-user system. The information you have stored on your computer belongs to you and no one else. Your information is a valuable resource that requires protection. The computer provides some of the most sophisticated security features available among personal computers. This section will point out the ways you can help keep the information in your computer secure. Four security features provided for your computer are

- system backups
- access permissions
- passwords
- data encryption

Sources of Potential Damage

To provide adequate security for your system, you first need to consider what kinds of problems might arise. It is important to consider every possible contingency: theft of the hardware, breakdowns in the hardware or software, tampering with data in the computer by unauthorized people, theft of data, and simple human error. To determine the types of problems to which your system may be vulnerable, ask yourself the following questions.

- Are you using your computer in a large or a small office? Is it possible for someone to steal your computer?

  Prevent theft by setting up your computer in a secure place.

- How many people have ready access to your system?

  Restrict the number of people with access to the computer by using passwords.

- How sensitive or valuable is the information you are storing?

  Restrict the number of people who have access to sensitive data with access permissions. Protect your data by encryption.

- Do you transmit data over telephone lines?

  Protect your data by encryption. Limit access to your computer telephone numbers.

The way you answer these questions will help you decide the measures you want to take to safeguard your data.

Basic Precautions

The following safeguards are recommended to everyone for ensuring the security of their computer and data.

- Set up your computer in an area that can be secured when you are not using it.
- Never leave your computer terminal logged in and unattended.
- If you are using your computer in an office environment, restrict the number of people who have access to the computer.

System Backups

If a breakdown occurs in the system, you may lose information that you had stored in the computer. You can avoid this type of loss if you have copies of your files stored on removable storage devices. Copy your data periodically onto floppy diskettes or cartridge tapes and store the diskettes or tapes in a safe place, away from the computer site.

This procedure is known as "backing up the system." With your files backed up regularly and stored safely off site, you won't have to worry about losing your information, even if your files are damaged.

There are two types of backups, system and incremental. In a system backup, you copy all mounted file systems that have been modified or created since the system was installed. In an incremental backup, you make copies of only those files that have been modified since the last backup.

Access Permissions

If you are sharing your computer with other users, you may want to share some information while keeping other information private. The UNIX Operating System allows you to satisfy both these needs by letting you define the following:

- the users that have permission to access data
- the types of permission they have (that is, how they are allowed to use the data)

The UNIX Operating System recognizes three categories of users of stored data: the owner of the data, the group to which the owner belongs, and all other people who use the system. Users can be given permission to use data in any or all of three ways: to read, write, and/or execute it.

Whenever you create a file or directory, the system automatically identifies the users who will be allowed to read, write, and/or execute the file and the users who will be allowed to access the directory. However, as the owner of the file, you have the ability to change the access permission after it has been created. The only person besides you who can change the permission on your file is the superuser, usually the system administrator.

One way to restrict access to your files is to change the permission modes. There are three sets of permission modes assigned to a file or directory: you (the user), the group, and everybody else. The permission modes are displayed every time you use the long list command (**ls -l**). See the *ls*(1) and *chmod*(1) manual pages in the *User's/System Administrator's Reference Manual* for additional information.

The following is an example of the permission modes of a file or directory:

```
-rwx rwx rwx (file)
drwx rwx rwx (directory)
```

Note that the first set of `rwx` refers to the permission mode for you (the owner), the second for the group, and the last for everybody else.

`r` allows you to read a file or to copy its contents

`w` allows you to write changes into a file or copy a file to a directory

`x` makes the file executable and allows a directory to be searched

Whenever a file or directory is created, the permission modes are automatically set for everyone to have access. Each permission mode is based on a number:

4 = read
2 = write
1 = execute

You can modify the permission modes of your file with the **chmod** command.

For example, to change the file *mycmd* so everyone can only execute it, enter the following command:

```
chmod 751 mycmd Enter
```

The permission modes for the file *mycmd* are now changed to `rwxr-x--x`. The 7 assigns the owner read, write, and execute permissions [4 (read) + 2 (write) + 1 (execute) = 7]. The 5 assigns the group read and execute permissions [4 (read) + 1 (execute) = 5]. The 1 assigns everybody else execute permission. See the *chmod*(1) manual page for additional information.

Also, to keep people from looking at the contents of your directory, you can deny execute permission to that directory. This has the effect of preventing others from visiting your directory.

You can also use the **umask** command to change the default permission modes of a file (for example 777) that you will be creating. The **umask** command can be placed in your *.profile* or the system default *profile*. The */etc/profile* is the system default profile. The **umask** command is specified as the following:

```
umask 000
```

The three octal digits (0 0 0) refer to read, write, and execute permission for owner, group, and other. The value of each permission digit is subtracted from the corresponding permissions digit. For example, if you entered **umask 022** in your system default */etc/profile*, a file normally created with 777 permission will be created with 755, and a file created with 666 will be created with 644. This is a way of ensuring that only you can write the file. See the *umask*(1) manual page in the *User's/System Administrator's Reference Manual* for additional information.

Password Administration

Whether or not users should have passwords is up to the administrator. For security reasons, it's a good idea to have a password. When a login is first established, a password can be assigned, or one can be assigned later. The person using the **root** login can assign or change a password for any login. See Appendix A, "Change Other User's Password."

You can also change your password by using the **passwd** command. See *passwd*(1M) in the *User's/System Administrator's Reference Manual*. While in the shell, type

```
$ passwd Enter
```

The system will prompt you for your old password, then the system will prompt you to enter the new password twice. If you are superuser, you will not be prompted for the old password. New passwords should be at least six characters long and must contain at least one numeric or special character. The space character is considered to be a special character. Other special characters include

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

The Password Files

The */etc/passwd* file identifies each user to the system. Every time a login is established, a new entry is added to this file. Each entry is one line that has seven fields separated by colons. There are two password files on your UNIX System, */etc/password* and */etc/shadow*. The */etc/password* file contains information for each user's login id, user id number, group id number, a comment on the user, the default program that is executed when the user logs in (usually **/bin/shell**), and the home directory. The */etc/shadow* file contains each user's encrypted password and password aging information.

All modifications to the *password* files should be done through the user interface or the **passwd** and **passmgmt** shell-level commands. The *password* files themselves should never be edited.

Password Aging

Password aging allows you to set time requirements on passwords. After a specified period of time, your password will expire, and you will be required to enter a new one. This forces you to change your password periodically. Provisions are made to prevent you from changing a new password before a specified time.

Normally, password aging is assigned by the system administrator using the **passwd** command.

When a login is assigned, there is no aging. Password aging has to be assigned for passwords to expire. The password aging information consists of the following:

- The duration of the password—how often the password must be changed.

  max is the duration of the valid password in days. See *passwd*(1M) in the *User's/System Administrator's Reference Manual.*

  min is the minimum number of days before a change can be made to a new password. See *passwd*(1M) in the *User's/System Administrator's Reference Manual.*

- The minimum time interval between password changes.
- The day when the password was last changed. You do not enter this information; the system automatically manages this information for each user.

When establishing the password aging information, there are three variables to keep in mind:

- MINWEEKS = *number*, where *number* defines how soon you can change a password after it was last changed. MINWEEKS is defined in */etc/default/passwd*. The system administrator can also change this variable for each user account using the **passwd** command (see *passwd*(1M) in the *User's/System Administrator's Reference Manual.*
- MAXWEEKS = *number*, where *number* defines the maximum amount of time you can retain a password for a user account. MAXWEEKS is defined in */etc/default/passwd*. The system administrator can also change this variable for each user account using the **passwd** command.

- IDLEWEEKS = *number*, where *number* defines how long a user's password account can remain idle without changing. IDLEWEEKS is defined in */etc/default/login*.

Data Encryption—Commands and Descriptions

If you have sensitive data that requires greater protection than that provided by access permission, you can encrypt the data. The encrypted file can not be read without a password. If somebody tried to read the encrypted file without a password, it could not be understood. The computer would display information in such a strange way no one could understand it.

NOTE

You will only have data encryption capabilities if you have installed the security package floppy diskette called Security Administration Package containing data encryption. Refer to Chapter 2 on installing optional add-on packages.

There are seven different commands used in data encryption. A brief summary of these commands appears in the following table.

COMMAND LINE	DESCRIPTION
crypt	This command is used to encode and decode files. The **crypt** command reads from the standard input or keyboard and writes to the standard output or terminal.
makekey	This command generates an encryption key.
ed -x	This command line edits a file that has already been encrypted or creates a new encrypted file using the **ed** editor.
vi -x	This command line edits a file that has already been encrypted or creates a new encrypted file using the **vi** editor.
ex -x	This command line edits a file that has already been encrypted or creates a new encrypted file using the **ex** editor.
edit -x	This command line edits a file that has already been encrypted or creates a new encrypted file using the **edit** editor.
X	This command encrypts a file while in the editor mode (**ed**, **ex**, or **edit**).

Crypt—Encode/Decode Files

The **crypt** command encodes and decodes files for security. When using **crypt**, you have to assign a password (key) to encode the file. The same password is used to decode the file. An encrypted file cannot be read unless the correct password is used to decode it.

If no password is given with the **crypt** command, the system will prompt you for one. For security, the screen does not display the password as you type it in.

Password security is the most vulnerable part of the **crypt** command. Anyone who figures out your password can look at your files. The best way to ensure your security is to select an uncommon group of characters. As with your login password, the password should be no more than eight letters or numbers long.

A file can be encrypted in the shell mode using **crypt** or in the edit mode using the **-x** or **X** option. When you are ready to decrypt the file, you can use the **crypt** command in the shell mode. The following is the command format to encrypt a file:

crypt < *oldfile* > *newfile* Enter

Before removing the unencrypted *oldfile*, make sure the encrypted *newfile* can be decrypted using the appropriate password. The *oldfile* is the file to be encrypted. The *newfile* is the name of the destination file for the encrypted text. The *oldfile* should now be removed. The system will prompt you for a password.

NOTE

Always remember to remove the file (*oldfile*) you're encrypting from because it will not be encrypted. Only the *newfile* will be encrypted.

Without any arguments, the **crypt** command takes standard input from the keyboard and encodes it before directing it to the standard output (the display). To encode an existing file, you must tell **crypt** to take its input (<) from a file instead of the keyboard. Similarly, you must tell **crypt** to send its output (>) to a new file instead of the display.

To decrypt a file, redirect the crypted file to a new file you can read. The command to decrypt a file is as follows:

crypt < *crypted_file* > *new_filename* Enter

NOTE Always encrypt and decrypt files separately.

Encrypting and Decrypting With Editors

The editors (**ed**, **edit**, **ex**, or **vi**) can be used to either edit an existing file that has been encrypted or to create a new encrypted file by using the **-x** option. When encrypting a file, you have to assign a password to encode the file. The same password is used to decode the file. An encrypted file cannot be read unless the correct password is used to decode it.

Select an uncommon group of characters for the password. It should be no more than eight characters long.

The following is the command format for the editors (**ed**, **edit**, **ex**, or **vi**) using the **-x** option:

$ ed -x [*filename*] Enter

$ edit -x [*filename*] Enter

$ ex -x [*filename*] Enter

$ vi -x [*filename*] Enter

The **-x** option is used to either edit an existing file that has been encrypted or to create a new encrypted file. The *filename* variable is the name of the file that is being created or edited. The system will prompt you for a password.

When you get ready to decrypt the file, you must use the **crypt** command from the shell.

The editor **X** command is another way to encrypt a file while in the editor mode. The **X** command will only work with the **ed**, **edit**, or **ex** editors. (For the **vi** editor, type **:X**.) This command also needs a password to encrypt and decrypt files.

After you have edited the file, you can easily encrypt it again by using the **X** command as follows:

1. While still in the editor, enter **X** on a line by itself.
2. The system will prompt you for a password.
3. Quit the file.

Helpful Hints

When working with a single-user computer or in a multi-user environment, there are a few things to keep in mind about system security. Here are some suggestions that may help you:

- Check your files and directories to be sure the permission modes are the way you want them. Set the permission modes to allow only the necessary permissions for owner, group, and others.
- Encrypt sensitive files. The **crypt** command can be used with different editors. Be sure to remember the passwords to your encrypted files.
- Assign passwords to all logins. Change your login password regularly. Don't make your passwords obvious; use a combination of numbers and letters instead of standard names.
- Remove or lock logins that are not needed (to keep others from using them). The **passwd -l** *user_login* command can be used to do this. See *passwd*(1M) in the *User's/System Administrator's Reference Manual.*
- Any system with dial-up ports is less secure than one without dial-up ports. If your system has dial-up ports, you should turn on the **login logging** feature as described later in this section. The feature allows you to monitor unsuccessful login attempts.

- Check the */usr/adm/sulog* file to monitor the use of the **su** command. The **su** command (changing to another login) can be dangerous since the knowledge of another login/password is required by the user. When more users know the login and password, the information in the system becomes less secure. Because of this, a log is kept on the use of the **su** command. Always enter the complete */bin/su* path name when changing to another login.
- Check the */usr/adm/loginlog* file to monitor login attempts that have failed more than five times. By default, the feature login logging is not turned on. To start it, enter:

```
# >/usr/adm/loginlog Enter
```

  This file (*/usr/adm/loginlog*) grows forever. Since it is not automatically deleted, you must manually delete it or write a shell script that is called by **cron** to do it for you.

- Be sure any files you have in the */bin*, */usr/bin*, or */etc* directories are write secured.
- Log off the system if you're going to be away from the terminal. Don't leave a logged in terminal unattended, especially if you're logged in as **root**.
- Keep your computer in an area that can be secured when you're not using it.
- Make backup copies of your files on floppy diskettes. Store your floppies in a safe place. If your files on the computer should be destroyed, you will always have a backup copy. See "Backing Up Files" in Chapter 4, "System Administration," for more information.
- Keep your boot diskette in a secure place (under lock and key).

Configuring Your Computer With Additional Terminals

Your computer can support multiple users. A terminal can be connected to the built-in serial port on the back of the computer. This allows two people to use the computer at one time. Of course, more than two people can have logins on the computer.

When the computer is powered up or rebooted, it will automatically be initialized to support both the console and the remote terminal if the serial port is appropriately set up as described in "Setting up Peripheral Devices" in Chapter 4, "System Administration."

Requirements for Multi-User Operation

Follow the procedures in "Setting up Peripheral Devices" in Chapter 4, "System Administration," for administering the serial port. The following are requirements for the computer to support a remote terminal:

- The shell variable TERM must match the type of terminal attached to your computer. This will ensure correct operation of screen-oriented applications, such as **vi**. The default profile (*/etc/profile*) automatically sets TERM to at386, corresponding to the computer console. At each remote terminal, you should set TERM as follows:

  TERM=*terminal_name*
  export TERM

  If each terminal is used frequently, you might want to set up the TERM command line in your *.profile*. If you frequently use several different types of remote terminals, you may want to modify your *.profile* to handle this.

- The terminal must have the correct baud rate set.
- The remote terminal must have a null modem cable.

Stopping a Command From a Remote Terminal

By default at login, the DEL character is the interrupt character. You should use the Del key on the remote terminal to stop execution of a command or program.

Installing Software Support for Additional Terminals

Part of the software included in the Foundation Set defines the terminals you can add to your computer. This group of terminal characteristics is known as the Terminal Information Library (terminfo). Each entry in the library represents one supported terminal.

The following are the terminals the computer will support.

AT&T Terminals:	**TERM name:**
Personal Terminal (510)	510
BCT 513	513
UNIX PC	unix_pc
6386WGS	AT386
4410/5410.	4410

Other Terminals:

ANSI X3.64	ansi
HP 262 series	2621
DEC VT 100.	vt100

If you're adding a terminal to your computer that is not on this list and you want the software support, you must add the additional terminal information. These entries are packaged separately in the Software Installation/Remote Terminal Package. The Remote Terminal Package is used to install the "terminfo" data base. The Remote Terminal Package installation has a separate procedure that involves selecting a terminal type from a list. Also refer to the "Setting Up Peripheral Devices" section in Chapter 4, "System Administration," for information on enabling the serial port.

Tunable System Parameters

The tunable parameter files contain kernel tunable parameters. These files are */etc/conf/cf.d/mtune* and */etc/conf/cf.d/stune*. These files can have a profound effect on system performance, and occasionally an add-on driver or kernel software module may have to modify an existing parameter or define a new tunable parameter that is accessible by other add-on drivers.

The *User's/System Administrator's Reference Manual* contains manual pages for *mtune* and *stune*. The *mtune* manual page defines a default value along with a minimum and maximum value for each kernel parameter. An add-on package should never modify a predefined system parameter in the *mtune* file.

Before proceeding, it may be useful to become familiar with the following manual pages for the Installable Drivers/Tunable Parameters (ID/TP) commands relating to tunable parameters.

- **idbuild**(1M) - A shell script that does the complete system reconfiguration
- **idtune**(1M) - A shell script that specifies system tunable parameters
- **idspace**(1M) - A command that interrogates free space in one or more file systems
- **mtune**(4) - A tunable parameter master file
- **stune**(4) - A tunable parameter system file

The remainder of this section describes procedures for modifying system tunable parameters for the computer. Adjusting these parameters can have a significant impact on system performance. Certain tunable parameters are normally adjusted upward when additional memory is installed to allow the system to support more users. For a computer used as a high-powered personal computer, or dedicated processor, however, it may not be necessary to increase kernel tunable parameters when additional memory is installed. In fact, tuning certain parameters normally associated with adding additional memory to support more users (NBUF, NCLIST, etc.) can actually decrease overall performance since these parameters increase kernel data space requirements, thus making less of the new memory available for user processes. Simply stated, the intended use of your computer and your observations on how well it is performing should be used as a guide in determining the need to adjust tunable parameters.

UNIX System V/386 uses an ID/TP scheme that was ported from the AT&T PC 6300 PLUS computer.

Additional information can be found in the *Integrated Software Development Guide*. In addition to the information provided in the *Integrated Software Development Guide*, there are other aspects of system configuration that can have a dramatic effect on system performance as well. For example:

- hard disk interleave
- file system organization
- use of text-bits (sticky bits)
- directory organization
- user $PATH efficiency
- use of larger file system block sizes (2K file system)
- use of **ps**, **sar**, process accounting, kernel profiling, and other system utilities to determine system utilization

Tunable system parameters are used to set various table sizes and system thresholds to handle the expected system load. For the most part, the initial tunable parameter values for your new computer are acceptable for most configurations and applications. If your application has special performance needs, you may have to experiment with different combinations of parameter values to find an optimal set. To modify kernel parameters, the UNIX System kernel will have to be reconfigured and the system rebooted.

The computer has an ID/TP scheme that places all system tunable parameters in a file *mtune* in the kernel configuration directory */etc/conf/cf.d*. The format of *mtune* is defined in the *mtune*(4) manual page in the *Programmer's Reference Manual*. A sample *mtune* file delivered with the initial UNIX System is shown in Figure 5-1.

```
April 1 9:01 1988  mtune Page 1

* General Kernel Parameters
NBUF          250    200    1000
NCALL         30     30     250
NINODE               150    100    600
NS5INODE      150    100    600
NFILE         150    100    600
NMOUNT               25     25     25
NPROC         100    50     400
NREGION       210    210    350
NCLIST               120    120    400
MAXUP         25     15     60
NOFILES       60     20     100
NHBUF         64     32     256
NPBUF         20     20     20
NAUTOUP       10     0      20
BDFLUSHR      1      1      1
MAXPMEM       0      0      4096
SHLBMAX       2      2      6
FLCKREC       100    100    100
PUTBUFSZ      2000   2000   10000
MAXSLICE      100    100    100
ULIMIT               3072   2048   12288
SPTMAP               50     50     50
PIOMAP               50     50     50
PIOMAXSZ      64     4      64
DO387CR3      0      0      1
* Device Driver Parameters ---------
NUMXT         3      1      3
NUMSXT               6      1      6
NCPYRIGHT     10     10     10
NKDVTTY       8      8      8
PRFMAX               2048   2048   2048
```

Figure 5-1: Sample *mtune* file (Sheet 1 of 4)

```
* Paging Parameters -----------------
VHNDFRAC    16     8      32
AGEINTERVAL        9      2      100
GPGSLO      25     0      25
GPGSHI      40     1      40
GPGSMSK     0x420  0x420  0x420
MAXSC       1      1      1
MAXFC       1      1      1
MAXUMEM     2560   2560   8192
MINARMEM    25     25     40
MINASMEM    25     25     40
MINHIDUSTK  4      4      32
MINUSTKGAP  2      2      32
* STREAMS Parameters ----------------
NQUEUE      96     0      4096
NSTREAM     32     0      512
NSTRPUSH    9      9      9
NSTREVENT   256    256    512
MAXSEPGCNT  1      0      32
NMUXLINK    87     0      87
STRMSGSZ    4096   4096   4096
STRCTLSZ    1024   1024   1024
NBLK4096    0      0      2048
NBLK2048    20     0      2048
NBLK1024    20     0      2048
NBLK512     8      0      2048
NBLK256     8      0      2048
NBLK128     8      0      2048
NBLK64      40     0      2048
NBLK16      40     0      2048
NBLK4       40     0      2048
```

Figure 5-1: Sample *mtune* file (Sheet 2 0f 4)

```
April 1 9:01 1988  mtune Page 2

STRLOFRAC  80    0     80
STRMEDFRAC 90    80    100
NLOG       3     3     3
NUMSP      32    5     50
NUMTIM     16    0     512
NUMTRW     16    0     512
* Message Parameters ---------------
MSGMAP     100   10    200
MSGMAX     2048  512   8192
MSGMNB     4096  4096  4096
MSGMNI     50    50    50
MSGSSZ     8     8     8
MSGTQL     40    40    40
MSGSEG     1024  1024  1024
* Semaphore Parameters -------------
SEMMAP     10    10    10
SEMMNI     10    10    10
SEMMNS     60    60    60
SEMMNU     30    30    30
SEMMSL     25    25    25
SEMOPM     10    10    10
SEMUME     10    10    10
SEMVMX     32767 32767 32767
SEMAEM     16384 16384 16384
* Shared Memory Parameters ----------
SHMMAX     524288      131072     524288
SHMMIN     1     1     1
SHMMNI     100   100   100
SHMSEG     6     6     15
SHMALL     512   256   512
```

Figure 5-1: Sample *mtune* file (Sheet 3 of 4)

```
* RFS Parameters ---------------------
NRCVD       150   40    500
NSNDD       100   100   350
NSRMOUNT    10    0     50
NADVERTISE  25    0     25
MAXGDP      24    10    32
MINSERVE    3     3     3
MAXSERVE    6     3     6
NRDUSER     250   0     700
RFHEAP      3072  1024  3072
NLOCAL      0     0     10
NREMOTE     0     0     10
RCACHETIME  10    -1    10
RFS_VHIGH   1     1     1
RFS_VLOW    1     1     1
* S52K (2K file system) Parameters --
S52KNBUF    100   100   400
S52KNHBUF   32    32    256
* XENIX Parameters ------------------
DSTFLAG           1     0     1
NSCRN       0     0     10
NEMAP       10    10    10
TIMEZONE    400   0     1140
XSEMMAX     60    20    60
XSDSEGS     25    1     25
XSDSLOTS    3     1     3
* DMA Parameter ---------------------
DMAEXCL     3     1     3
```

Figure 5-1: Sample *mtune* File (Sheet 4 of 4)

As noted in the *mtune* manual page, each system tunable parameter is assigned a default value along with a minimum and maximum value. You can examine *mtune* to determine the tunable parameter settings for your computer; however, you should never modify the *mtune* file. A second file in the configuration directory *stune* is used to modify a parameter to any value between the minimum and maximum value defined in the *mtune* file. See the "Reconfiguring the Kernel to Enable New Parameters" section for step-by-step procedures to modify system tunable parameters.

When to Tune and What to Tune

Some UNIX Systems running on superminicomputers or mainframe computers support dozens of users simultaneously. As additional users are added to those systems, additional memory is often added, and system parameters are adjusted to allow the UNIX System kernel to operate more efficiently. That often includes allocating more memory to kernel data space by increasing the size of kernel data structures. This generally allows the system to support more users. However, as these kernel data structures are increased in size, it takes more time for the kernel to scan those structures, and in fact, increasing certain parameters unnecessarily can actually slow the system down. For example, increasing the parameter NPROC allows the system to maintain a larger list (the proc table) of active processes. This can have an adverse effect on the kernel scheduler since it now must repeatedly scan this larger table every time it checks to see which process to run next. Additionally, since the kernel data space requirements increase when table sizes are increased, there will be less memory space available for user processes, which can also lower overall performance.

Special Case Needs

Often your system usage will present you with the need to tune certain parameters for particular circumstances. A common need is the ability to create very large files. This can be accomplished by becoming superuser and modifying the "ulimit" for the particular shell process that you are running as superuser. An alternate solution is to modify the system ULIMIT for all users. The ULIMIT parameter and other commonly encountered limits are summarized in Figure 5-2.

Desired Improvement	Parameters
Improve system performance(*) when additional memory installed.	NBUF, NHBUF
Other performance-related system parameters.	NAUTOUP, MAXSLICE, BDFLUSHR, AGEINTERVAL (also see paging parameters)
Increase system limits when additional memory installed (support more users; reduce chances of system problems at times of heavy load, etc.).	NCALL, NINODE, NSINODE, NFILE, NPROC, NREGIONS, NCLIST (also see message, semaphore, and shared memory parameters)
Users need to create bigger files.	ULIMIT
Each user needs to open more files.	NOFILES
Each user needs to run more processes.	MAXUP
Other system limits that may be encountered.	SHLBMAX, FLCKREC, SPTMAP, NUMXT, NUMSXT, PRFMAX (also see STREAMS, and RFS parameters)
Miscellaneous.	PUTBUFSIZE, DO387CR3

(*) Note that increasing the size of the buffer cache will increase the chances that data frequently read will be found in memory rather than on the disk. Depending on your system usage, an increase in the chance of reusing a data block may not yield an overall system performance improvement. In some system usage scenarios, it can provide a significant performance improvement. See the "Buffer Cache" section.

Figure 5-2: Special Case Tuning Needs.

Kernel Messages That System Limits Are Being Exceeded

There are cases when the UNIX System kernel will advise you that system limits are being exceeded. These messages are in the form of console printouts. Some of the messages are advisory only. Others precede a system panic in which case additional diagnostic messages are printed, and the system will "hang" requiring you to reboot. If you encounter any of the messages listed in Figure 5-3, refer to the appropriate tunable parameter for additional information.

Kernel Console Message	Parameter
iget - inode table overflow	NINODE
Timeout table overflow	NCALL
File table overflow	NFILE
mfree map overflow *n*(*)	SPTMAP
Region table overflow	NREGION
Configured value of NOFILES *n*(*) is less than minimum (greater than maximum).	NOFILES
stropen: out of streams	NSTREAM
swapdel - too few free pages	MINASMEM
stropen: out of streams	NSTREAM
stropen: out of queues	NQUEUE

(*) The value *n* indicates the actual value encountered by the kernel.

Figure 5-3: Kernel Messages and Associated Tunable Parameter

Buffer Cache

The NBUF parameter specifies the number of 1K buffers in the system buffer cache. These buffers hold recently used data on the chance that it will be needed again. If a read or a write can be satisfied using the buffer cache instead of the disk, system performance improves because memory operations are much faster than disk operations. NHBUF specifies the number of hashing buckets in the buffer cache. The more buffers, the greater chance that data can be found in the buffers without the system having to do a time-consuming disk read. The read and write cache hit ratios listed by the **sar -b** command

indicate how effective the system buffers are. The value for NHBUF is a power of 2 that is roughly one-quarter of the value of NBUF.

The values of NBUF and NHBUF given in the *mtune* file are a good starting point for the buffer cache. These values come close to optimum for most system workloads. Increasing NBUF and NHBUF, up to a point, may improve system performance. A system with 2 megabytes of memory can typically devote roughly 250K bytes of memory to buffers, while a system with 4 megabytes of memory can devote 400K bytes of memory to buffers. However, if too many buffers are allocated, there may not be enough memory space for efficient operation of user processes, and the amount of swapping done by the system will increase. The swapping activity usually costs more in system efficiency than is gained by having a large amount of buffer space. If the **sar -w** command shows that your system has **swpot/s** greater than 1.0, adding buffers may not be beneficial. Additionally, by increasing the number of buffers the kernel must manage, the kernel routines dealing with managing the allocation and freeing of buffers may take longer to execute.

If you choose to modify the number of buffers, after the UNIX System has run for a day or so, check system performance, particularly excessive swapping activity. If such activity is found, reduce the number of buffers.

The parameters S52KNBUF and S52KNHBUF perform analogous functions for 2K buffers. When you increase 2K buffers, you may want to lower the number of 1K buffers so that you maintain the memory available for user processes.

What to Do When You Add More Memory

In the past, most UNIX System administrators would routinely increase all system tunable parameters when additional memory was installed in minicomputers and superminicomputers. This would usually allow the UNIX System to support more users without encountering system limits during heavy system activity. For a single-user PC environment, however, there may be no need to increase kernel tunable parameters at all. And, for the reasons previously stated, keeping system limits at the default value may deliver optimum performance even when additional memory is installed.

As Figure 5-4 shows, the default parameters defined in the base system *mtune* file are intended for a 2-megabyte memory system. This is the minimum memory size recommended for a computer running the UNIX Operating System. If your system will be used in a multi-user configuration,

that is, five users or more, you may wish to add more memory and increase selective parameters to be sure system limits are not reached, and to increase the buffer cache hit ratio. The values for selective parameters are given for a 3-megabyte and 4-megabyte memory configuration. You should try to establish a performance baseline before making these changes, modify your system parameters, and again determine your system's performance. This is the best approach to see if parameter changes increase or decrease your systems performance.

	Memory Size		
Parameter	2 Meg	3 Meg	4 Meg
NBUF	250	300	400
NHBUF	64	64	128
NCALL	30	40	50
NINODE	150	200	300
NS5INODE	150	200	300
NFILE	150	200	300
NREGION	210	250	300
NCLIST	120	140	170
NPROC	100	120	150

Figure 5-4: Suggested Parameter Values Based on Memory Size

Parameter Descriptions

The following sections provide a breakdown of system tunable parameters defined in the file */etc/conf/cf.d/mtune*. The parameter categories are as follows:

General Kernel Parameters
Device Driver Parameters

Paging Parameters
Streams Parameters
Message Parameters
Semaphore Parameters
Shared Memory Parameters
Remote File Sharing Parameters
S52K (2K File System) Parameters
XENIX Parameters
DMA Parameters

Note that the Streams Parameters determine configuration of the Network Support Utilities (NSU) add-on package, the RFS Parameters determine configuration of the Remote File Sharing add-on, the STARLAN Parameters control configuration of the STARLAN add-on, and the S52K (2K File System) parameters control configuration of the S52K add-on. If these packages are not installed, adjusting the parameter values will have no effect upon your system's configuration.

General Kernel Parameters

NBUF — Specifies how many 1K system buffers to allocate. The UNIX System buffers form a data cache. The data cache is a memory array containing disk file information. Cache hit rate increases with the number of buffers. Cache hits reduce the number of disk accesses and thus may improve overall performance. The entries are normally in the range of 100 to 600. Each buffer contains 1076 bytes. 1K hash buffers (NHBUF) should be increased along with system buffers (NBUF) for optimal performance.

NCALL — Specifies how many call-out table entries to allocate. Each entry represents a function to be invoked at a later time by the clock handler portion of the kernel. This value must be greater than 2 and is normally in the range of 10 to 70. The default value is 30. Each entry contains 16 bytes.

Software drivers may use call entries to check hardware device status. When the call-out table overflows, the system crashes and displays the following message on the system console:

```
PANIC: Timeout table overflow
```

NINODE

Specifies how many i-node table entries to allocate. Each table entry represents an in-core i-node that is an active file. For example, an active file might be a current directory, an open file, or a mount point. The file control structure is modified when changing this variable. The number of entries used depends on the number of opened files. The entries are normally in the range of 100 to 400. The value for NINODE pertains directly to the NFILE value. (NINODE is equal to or greater than NFILE). NINODE must always be less than or equal to NS5INODE. NINODE greater than NS5INODE results in an unusable system. When the i-node table overflows, the following warning message is displayed on the system console:

```
WARNING: i-node table overflow
```

NS5INODE

Must always be equal to or greater than NINODE.

NFILE

Specifies how many open file table entries to allocate. Each entry represents an open file. The entry is normally in the range of 100 to 400. Each entry contains 12 bytes. The NFILE entry relates directly to the NINODE entry. (NFILE is less than or equal to NINODE). The NFILE control structure operates in the same manner as the NINODE structure. When the file table overflows, the following warning message is displayed on the system console:

```
NOTICE: file table overflow
```

As a reminder, this parameter does not affect the number of open files per process (see the NOFILES parameter).

NMOUNT — Specifies how many mount table entries to allocate. Each entry represents a mounted file system. The *root* (/) file system is always the first entry. When full, the **mount** system call returns the error EBUSY. Since the mount table is searched linearly, this value should be as low as possible.

NPROC — Specifies how many process table entries to allocate. Each table entry represents an active process. The swapper is always the first entry, and */etc/init* is always the second entry. The number of entries depends on the number of terminal lines available and the number of processes spawned by each user. The average number of processes per user is in the range of 2 to 5 (also see MAXUP, default value 25). When full, the **fork** system call returns the error EAGAIN. The NPROC entry is in the range of 50 to 200.

NREGION — Specifies how many region table entries to allocate. Each NREGION entry contains 36 bytes. Most processes have 3 regions: text, data, and stack. Additional regions are needed for each shared memory segment and shared library (text and data) attached. However, the region table entry for the text of a "shared text" program will be shared by all processes executing that program. Each shared memory segment attached to one or more processes uses another region table entry. A good starting value for this parameter is about 3.5 times NPROC. If the system runs out of region table entries, the following message is displayed on the system console:

```
Region table overflow
```

NCLIST — Specifies how many character list buffers to allocate. Each buffer contains up to 64 bytes. The buffers are dynamically linked to form input and output queues for the terminal lines and other slow-speed devices. The average number of buffers needed per terminal is in the range of 5 to 10. Each entry (buffer space plus header) contains 72 bytes. When full, input and output characters dealing with terminals are lost, although echoing continues.

MAXUP

Specifies how many concurrent processes a non-superuser is allowed to run. The entry is normally in the range of 15 to 40. This value should not exceed the value of NPROC (NPROC should be at least 10% more than MAXUP). This value is per user identification number, not per terminal. For example, if 12 people are logged in on the same user identification, the default limit would be reached very quickly.

NOFILES

Specifies the maximum number of open files per process. The default is 60. Unless an application package recommends that NOFILES be changed, the default setting of 60 should be left as is.

/bin/sh uses three file table entries: standard input, standard output, and standard error (0, 1, and 2 are normally reserved for stdin, stdout, and stderr, respectively). This leaves the value of NOFILES minus 3 as the number of other open files available per process. If a process requires up to three more than this number, then the standard files must be closed. This practice is not recommended and must be used with caution, if at all.

If the configured value of NOFILES is greater than the maximum (100) or less than the minimum (20), the configured value is set to the default (20), and a NOTICE message is sent to the console.

NHBUF

Specifies how many "hash buckets" to allocate for 1K buffers. These are used to search for a buffer given a device number and block number rather than a linear search through the entire list of buffers. **This value must be a power of 2.** Each entry contains 12 bytes. The NHBUF value must be chosen so that the value NBUF divided by NHBUF is approximately equal to 4.

NPBUF

Specifies how many physical I/O buffers to allocate. One I/O buffer is needed for each physical read or write active. Each entry contains 52 bytes. The default value is 20.

NAUTOUP	The NAUTOUP entry specifies the buffer age in seconds for automatic file system updates. A system buffer is written to the hard disk when it has been memory-resident for the interval specified by the NAUTOUP parameter. Specifying a smaller limit increases system reliability by writing the buffers to disk more frequently and decreases system performance. Specifying a larger limit increases system performance at the expense of reliability. This parameter controls behavior of the **bdflush** daemon process.
BDFLUSHR	Specifies the rate in seconds for checking the need to write the file system buffers to the disk. The default is 1 second. This parameter controls behavior of the **bdflush** daemon process.
MAXPMEM	Specifies the maximum amount of physical memory to use in pages. The default value of 0 specifies that all available physical memory be used.
SHLBMAX	Specifies the maximum number of shared libraries that can be attached to a process at one time.
FLCKREC	Specifies the number of records that can be locked by the system. The default value is 100. Each entry contains 28 bytes.
PUTBUFSZ	Specifies the size of a circular buffer, **putbuf**, that is used to contain a copy of the last PUTBUFSZ characters written to the console by the operating system. The contents of **putbuf** can be viewed using **crash**.
MAXSLICE	Specifies in clock ticks the maximum time slice for user processes. After a process executes for its allocated time slice, that process is suspended. The operating system then dispatches the highest priority process and allocates to it MAXSLICE clock ticks. MAXSLICE, is normally one second (100 clock ticks on the WGS 6836).
ULIMIT	Specifies in 512-byte blocks the size of the largest file that an ordinary user may write. The default value is 2048; that is, the largest file an ordinary user may write is one megabyte. The superuser may write a file as large as the

	file system can hold. The ULIMIT parameter does not apply to reads: any user may read a file of any size.
SPTMAP	Determines the size of the map entry array used for managing kernel virtual address space. Users should not modify this parameter.
PIOMAP	Determines the size of the map entry array used by the kernel programmed I/O (PIO) breakup routine. This routine allows device drivers to do programmed I/O of large data blocks at interrupt level by breaking the data blocks into smaller data units. Users should not modify this parameter.
PIOMAXSZ	Maximum number of pages to use at one time for programmed I/O. Users should not modify this parameter.
DO387CR3	Controls the setting of high order bits of Control Register 3 (CR3) when an 80387 math coprocessor is installed.

Device Driver Parameters

The following parameters control various data structure sizes and other limits in base system device drivers.

NUMXT	Determines the number of layers a subdevice can configure to support bitmapped display devices such as the BLIT or the AT&T 5620 terminal.
NUMSXT	Determines the number of shell layers a subdevice can configure.
NCPYRIGHT	Defines the size of a kernel data structure used to print console initialization messages. Users should not modify this parameter.
NKDVTTY	Determines the number of virtual terminals (ttys) supported by the console keyboard driver. Users should not modify this parameter.
PRFMAX	Maximum number of text symbols that the kernel profiler (*/dev/prf*) will be able to properly process.

Paging Parameters

A paging daemon, **vhand**, is responsible for freeing up memory as the need arises. It uses a "least recently used" algorithm to approximate process working sets, and it writes out those pages that have not been modified during some period of time to the disk. The page size is 4096 bytes. When memory is exceptionally tight, the working sets of entire processes may be swapped out.

The following tunable parameters determine how often **vhand** and **bmapflush** run and under what conditions. The default values should be adequate for most applications.

AGEINTERVAL — Specifies the number of clock ticks a process runs before its pages get aged.

VHNDFRAC — Determines the initial value for the system variable VHANDL. VHANDL is set to the maximum user-available memory divided by VHNDFRAC or the value of GPGSHI, whichever is larger. The value of VHANDL determines when the paging daemon **vhand** runs. The amount of available free memory is compared with the value of VHANDL. If free memory is less than VHANDL, then the paging daemon **vhand** is awakened.

The default for VHNDFRAC is 16. Decrease the value to make the daemon more active; increase the value to make the daemon less active. (The value must be > 0 and < 25 percent of available memory.)

VHANDL — See VHNDFRAC above.

GPGSLO — Specifies the low water mark of free memory in pages for **vhand** to start stealing pages from processes. The default is 25. Increase the value to make the daemon more active; decrease the value to make the daemon less active (must be an integer ≥ 0 and $<$ GPGSHI).

GPGSHI — Specifies the high water mark of free memory in pages for **vhand** to stop stealing pages from processes. The default is 40. Increase the value to make the daemon more

active; decrease the value to make the daemon less active. (The value must be an integer > 0, > GPGSLO, and < 25 percent of the number of pages of available memory.)

GPGSMSK — Mask used by the paging daemon. The default is 0x00000420. This value should not be changed.

MAXSC — Specifies the maximum number of pages that will be swapped out in a single operation. The default value is 1.

MAXFC — Specifies the maximum number of pages that will be added to the free list in a single operation. The default value is 1.

MAXUMEM — Specifies the maximum size of a user's virtual address space in pages. This value cannot be greater than 8192. The default is 2560.

MINARMEM — Specifies the minimum number of memory pages reserved for the text and data segments of user processes.

MINASMEM — Threshold value that specifies the number of memory and swap pages reserved for system purposes (unavailable for the text and data segments of user processes).

MINHIDUSTK — Specifies the minimum data relocation value such that the user stack and data can share a page table. These values should not be changed.

MINUSTKGAP — See MINHIDUSTK above.

Streams Parameters

The following tunable parameters are associated with Streams processing. The values have no effect on the system unless the Network Support Utilities (NSU) package is installed.

NQUEUE — The number of Streams queues to be configured. Queues are always allocated in pairs, so this number should be even. A minimal Stream contains four queues (two for the Stream head, two for the driver). Each module pushed on a Stream requires an additional two queues. A typical configuration value is 4*NSTREAM.

NSTREAM — The number of "Stream-head" (stdata) structures to be configured. One is needed for each Stream opened, including both Streams currently open from user processes and Streams linked under multiplexers. The recommended configuration value is highly application-dependent, but a value of 32-40 usually suffices on a computer for running a single transport provider with moderate traffic.

NSTRPUSH — The maximum number of modules that may be pushed onto a Stream. This is used to prevent an errant user process from consuming all of the available queues on a single Stream. By default this value is 9, but in practice, existing applications have pushed at most four modules on a Stream.

NSTREVENT — The initial number of Stream event cells to be configured. Stream event cells are used for recording process-specific information in the **poll** system call. They are also used in the implementation of the STREAMS I_SETSIG **ioctl** and in the kernel **bufcall()** mechanism. A rough minimum value to configure would be the expected number of processes to be simultaneously using **poll** times the expected number of Streams being polled per process, plus the expected number of processes expected to be using Streams concurrently. The default is 256. Note that this number is not necessarily a hard upper limit on the number of event cells that will be available on the system (see MAXSEPGCNT).

MAXSEPGCNT — The number of additional pages of memory that can be dynamically allocated for event cells. If this value is 0, only the allocation defined by NSTREVENT is available for use. If the value is not 0 and if the kernel runs out of event cells, it will under some circumstances attempt to allocate an extra page of memory from which new event cells can be created. MAXSEPGCNT places a limit on the number of pages that can be allocated for this purpose. Once a page has been allocated for event cells, however, it cannot be recovered later for use elsewhere. It is recommended that the NSTREVENT value be set to

accommodate most load conditions and that MAXSEPGCNT be set to 1 to handle exceptional load cases should they arise.

NMUXLINK — The maximum number of multiplexer links to be configured. One link structure is required for each active multiplexer link (STREAMS I_LINK **ioctl**). This number is application-dependent; the default allocation guarantees availability of links.

STRMSGSZ — The maximum allowable size of the data portion of any Streams message. This should usually be set just large enough to accommodate the maximum packet size restrictions of the configured Streams modules. If it is larger than necessary, a single **write** or **putmsg** can consume an inordinate number of message blocks. The recommend value of 4096 is sufficient for existing applications.

STRCTLSZ — The maximum allowable size of the control portion of any Streams message. The control portion of a **putmsg** message is not subject to the constraints of the minimum/maximum packet size, so the value entered here is the only way of providing a limit for the control part of a message. The recommended value of 1024 is more than sufficient for existing applications.

NBLK*n* — NBLK4 through NBLK4096 control the number of Streams data blocks and buffers to be allocated for each size class. Message block headers are also allocated based on these numbers: the number of message blocks is 1.25 times the total of all data block allocations. This provides a message block for each data block, plus some extras for duplicating messages (kernel functions **dupb**(), **dupmsg**()). The optimal configuration depends on both the amount of primary memory available and the intended application. The default values provided in the NSU package are intended to support a moderately loaded configuration using Remote File Sharing (RFS) and UUCP/CU over STARLAN.

STRLOFRAC — The percentage of data blocks of a given class at which low-priority block allocation requests are automatically failed. For example, if STRLOFRAC is 40 and there are forty-eight 256-byte blocks, a low-priority allocation request will fail when more than nineteen 256-byte blocks are already allocated. The parameter is used to help prevent deadlock situations by starving out low-priority activity. The recommended value of 40 works well for current applications. STRLOFRAC must always be in the range 0 <= STRLOFRAC <= STRMEDFRAC.

STRMEDFRAC — The percentage cutoff at which medium priority block allocations are failed (see STRLOFRAC discussion above). The recommended value of 90 works well for current applications. STRMEDFRAC must always be in the range STRLOFRAC <= STRMEDFRAC <= 100.

NOTE — There is no cutoff fraction for high-priority allocation requests; it is effectively 100.

NLOG — The number of minor devices to be configured for the log driver; the active minor devices will be 0 through (NLOG–1). The recommended value of 3 services an error logger (**strerr**) and a trace command (**strace**), with one left over for miscellaneous usage. If only an error logger and a tracer are to be supported, this number can be set to 2. If there are several daemons for an application that may be submitting log messages, this number can be increased to accommodate the extra users.

NUMSP — Determines the number of Streams pipe devices (*/dev/sp*) supported by the system. Users should not modify this parameter.

NUMTIM — The maximum number of Streams modules that can be pushed by the Transport Library Interface (TLI). This value controls the number of data structures used to hold pushed Streams modules configuration data. Users should not modify this parameter.

NUMTRW — The number of Transport Library Interface (TLI) read/write data structures to allocate in kernel data space. Users should not modify this parameter.

Message Parameters

The following tunable parameters are associated with interprocess communication messages:

MSGMAP — Specifies the size of the control map used to manage message segments. The default value is 100. Each entry contains 8 bytes.

MSGMAX — Specifies the maximum size of a message. The default value is 2048. Although the maximum possible size the kernel can process is 64 kilobytes –1, the *mtune* limit is 8192.

MSGMNB — Specifies the maximum length of a message queue. The default value is 4096.

MSGMNI — Specifies the maximum number of message queues system-wide (id structure). The default value is 50.

MSGSSZ — Specifies the size, in bytes, of a message segment. Messages consist of a contiguous set of message segments large enough to fit the text. The default value is 8. The value of MSGSSZ times the value of MSGSEG must be less than or equal to 131,072 bytes (128 kilobytes).

MSGTQL — Specifies the number of message headers in the system and, thus, the number of outstanding messages. The default value is 40. Each entry contains 12 bytes.

MSGSEG — Specifies the number of message segments in the system. The default value is 1024. The value of MSGSSZ times the value of MSGSEG must be less than or equal to 131,072 bytes (128 kilobytes).

Semaphore Parameters

The following tunable parameters are associated with interprocess communication semaphores:

SEMMAP — Specifies the size of the control map used to manage semaphore sets. The default value is 10. Each entry contains 8 bytes.

SEMMNI — Specifies the number of semaphore identifiers in the kernel. This is the number of unique semaphore sets that can be active at any given time. The default value is 10. Each entry contains 32 bytes.

SEMMNS — Specifies the number of semaphores in the system. The default value is 60. Each entry contains 8 bytes.

SEMMNU — Specifies the number of undo structures in the system. The default value is 30. The size is equal to 8 times (SEMUME + 2) bytes.

SEMMSL — Specifies the maximum number of semaphores per semaphore identifier. The default value is 25.

SEMOPM — Specifies the maximum number of semaphore operations that can be executed per **semop** system call. The default value is 10. Each entry contains 8 bytes.

SEMUME — Specifies the maximum number of undo entries per undo structure. The default value is 10. The size is equal to 8 times (SEMMNU) bytes.

SEMVMX — Specifies the maximum value a semaphore can have. The default value is 32767, which is the maximum value for this parameter.

SEMAEM — Specifies the adjustment on exit for maximum value, alias **semadj**. This value is used when a semaphore value becomes greater than or equal to the absolute value of **semop**, unless the program has set its own value. The default value is 16384. The default value is the maximum value for this parameter.

Shared Memory Parameters

The following tunable parameters are associated with interprocess communication shared memory:

SHMMAX — Specifies the maximum shared memory segment size. The default value is 524288.

SHMMIN — Specifies the minimum shared memory segment size. The default value is 1.

SHMMNI — Specifies the maximum number of shared memory identifiers system wide. The default value is 100. Each entry contains 52 bytes.

SHMSEG — Specifies the number of attached shared memory segments per process. The default value is 6. The maximum value is 15.

SHMALL — Specifies the maximum number of in-use shared memory text segments. The default value is 512.

Remote File Sharing (RFS) Parameters

There are several parameters you can tune to best suit the way you use Remote File Sharing (RFS). RFS parameters control the amount of system resources you devote to RFS service. Each network transport provider may also have some tunable parameters that may affect performance characteristics of that particular network. See the network documentation for your network for more details.

All parameters have set default values that should work well for an average system; however, if the values are too small, you may not be providing enough resources to properly handle your RFS load. Requests for mounts, advertises, or even a file could fail if either of those values reach the maximum number allowed for your machine. If these parameters are too large, you could be allocating more system resources than you need to use.

Note that these parameters have no effect on your system unless the RFS add-on package is installed.

NRCVD (maximum number of receive descriptors) — Your system creates one receive descriptor for each file or directory being referenced by remote users and one for

each process on your machine awaiting response to a remote request. If you limit the number of receive descriptors, you limit the number of local files and directories that can be accessed at a time by remote users. The result of exceeding the limit would be error messages for remote user commands.

NSNDD (maximum number of send descriptors)

For each remote resource (file or directory) your users reference, your system creates a send descriptor. A send descriptor is also allocated for each server process and each message waiting on the receive queue. You can change this value to limit how many remote files and directories your machine can access at a time. This would, in effect, limit the amount of RFS activities your users can perform. The result of exceeding the limit would be error messages for user commands.

NSRMOUNT (server mount table entries)

Each time a remote machine mounts one of your resources, an entry is added to your server mount table. This number limits the total number of your resources that can be mounted at a time by remote machines.

NADVERTISE (advertise table)

An entry is placed in your advertise table for each resource you advertise. This parameter sets the maximum resources you can advertise.

MAXGDP (virtual circuits)

There are up to two connections (virtual circuits) set up on the network between you and each machine with which you are currently sharing resources. There is one for each computer whose resources you mount and one for each computer that mounts your resources. A virtual circuit is created when a computer first mounts a resource from another, and it is taken down when the last resource is unmounted.

This parameter limits the number of RFS virtual circuits your computer can have open on the network at a time. It limits how many remote computers you can share

resources with at a time. Note that a given network may have a limited number of circuits on any one computer, so this parameter influences the maximum percentage of those that might be used for RFS.

MINSERVE (minimum server processes)
Your system uses server processes to handle remote requests for your resources. This parameter sets how many server processes are always active on your computer. (See the **sar -S** command for information on monitoring server processes.)

MAXSERVE (maximum server processes)
When there are more remote requests for your resources than can be handled by the minimum servers, your computer can temporarily create more. This parameter sets the maximum total server processes your system can have (MINSERVE plus the number it can dynamically create).

NRDUSER
This value specifies the number of receive descriptor **user** entries to allocate. Each entry represents a client machine's use of one of your files or directories. While there is one receive descriptor allocated for each file or directory being accessed remotely (NRCVD), there can be multiple receive descriptor **user** entries for each client using the file or directory (NRDUSER). These entries are used during recovery when the network or a client goes down. This value should be about one and one-half times the value of NRCVD.

RFHEAP
This value specifies the size in bytes of an area of memory set aside for RFS information. It contains the following information:

- The user and group ID mapping tables and the domain name of each machine currently sharing a resource(s) with your machine.

- A list of machine names supplied as a client list when you advertise resources.

The appropriate size for RFHEAP depends on the following:

☐ UID/GID tables (size and number).

There will always be two global tables, one UID and one GID. Also, any machine with a **host** entry in *uid.rules* or *gid.rules* files will have a table corresponding to each of these entries while it is connected to this machine. Machines that do not have separate entries in one of these files do not take any extra space.

To estimate the size on an individual table, type **idload -n**. There will be one 4-byte table entry per line of output from **idload**, plus up to 24 bytes of overhead per table.

☐ Adv client lists (size and number).

Each advertise may have a list of authorized clients attached to it. This list is stored in this area, with its size unchanged, until the resource is unadvertised.

☐ Currently connected resources.

Each connection will use a maximum of 64 bytes to store the name of the connected resource. This memory is allocated dynamically, so some additional space is required to account for possible fragmentation as space is allocated and de-allocated. Since the total size is likely to be relatively small, 1 to 4 kilobytes, it is best to allow too much rather than too little space.

NLOCAL (local access buffers)

This parameter sets the minimum number of local buffers, available from the common buffer pool, reserved for local access. RFS client caching shares the common buffer pool with the local accesses (usually disk or tape). This value, therefore, protects local data from adverse effects of competition with RFS buffer use.

When this threshold is turned off (set to 0), it defaults to the recommended value of one third of the entire buffer pool (NBUF). A non-zero value of NLOCAL overrides this default.

Note that if RFS is not running or has had no recent activity, the entire buffer pool will be available to local access.

NREMOTE (remote access buffers)

This parameter sets the minimum number of local buffers, available from the common buffer pool, reserved for remote resource read data. When this threshold is turned off (set to 0), it defaults to the recommended value of one third of the entire buffer pool (NBUF). A non-zero value of NREMOTE overrides this default.

Note that the sum of NREMOTE and NLOCAL must not be greater than NBUF. If this condition is detected, a console warning message is printed, and the default value (one third of NBUF) is used for both NREMOTE and NLOCAL.

RCACHETIME (caching time off)

This parameter can be used in two ways: to turn off caching for your entire machine or to define the number of seconds that network caching is turned off when a file is modified.

To turn off caching for your entire machine, the parameter must be set to -1.

	The second use of RCACHETIME requires some explanation. When a write to a server file occurs, the server machine sends invalidation messages to all client machines that have the file open. The client machines remove data affected by the write from their caches. Caching of that file's data is not resumed until the writing processes close the file or until the seconds in this parameter have elapsed. The assumption is that write traffic is "bursty" and that the first write may be closely followed by other writes. Turning off caching avoids the overhead of sending invalidation messages for subsequent writes.
RFS_VHIGH	Highest RFS version number with which your machine will communicate.
RFS_VLOW	Lowest RFS version number with which your machine will communicate.

In addition to the above, the NHBUF parameter has implications for RFS. The value of NHBUF is used to specify how many "hash buckets" to allocate for remote data in the buffer pool, as well as for local data. The hash buckets are used to search for a buffer given a remote server machine ID and file ID, rather than a linear search through the entire list of buffers. (See the "General Kernel Parameters" section for further discussions of NHBUF.)

Figure 5-5 lists the key RFS parameters and recommended values for different uses of RFS. "Client Only" means that your machine will only be using remote resources, not sharing any from your own machine. "Server Only" means you will only offer your resources to other machines without mounting any remote resources. "Client+Server" means you will both offer local resources and use remote resources.

Parameter	Client Only	Server Only	Client+Server	Default Value	Size per Entry in Bytes
NSRMOUNT	0	50	50	24	
MAXGDP	10	24	24	24	104
NADVERTISE	0	25	25	25	32
NRCVD	40	300	150	150	48
NRDUSER	0	450	225	225	24
NSNDD	150	30	150	150	44
MINSERVE	0	3	3	3	9K
MAXSERVE	0	6	6	6	-
RFHEAP	2048	3072	3072	3072	1
NREMOTE	0	0	0	0	-
NLOCAL	0	0	0	0	-
RCACHETIME	10	10	10	10	-

Figure 5-5: RFS Tunable Parameter Settings

S52K (2K File System) Parameters

Note that these parameters have no effect on your system unless the S52K add-on package is installed.

S52KNBUF — Specifies how many 2K system buffers to allocate. This parameter performs the same function for 2K file systems that NBUF performs for 1K file systems. The entries are normally in the range of 100 to 400. Each buffer contains 2100 bytes. 2K hash buffers (S52KNHBUF) should be increased along with S52KNBUF for optimal performance. If you configure 2K buffers in your system, you should reduce the number of 1K buffers (NBUF) to keep available memory at an acceptable level.

S52KNHBUF — Specifies how many hash buckets to allocate for 2K buffers. These are used to search for a buffer given a device number and block number rather than a linear search through the entire list of buffers. **This value must be a power of 2.** Each entry contains 12 bytes. The S52KNHBUF value must be chosen so that the value of

S52KNBUF divided by S52KNHBUF is approximately equal to 4.

XENIX Tunable Parameters

The following describes the XENIX tunable parameters:

DSTFLAG — Specifies the **dstflag** described for the XENIX *ftime*(S) system call.

NSCRN — Specifies the maximum number of virtual terminals that can be used by VT and console drivers.

NEMAP — Specifies the maximum number of I/O translation mappings.

TIMEZONE — Specifies the **timezone** setting referred to in the XENIX *ftime*(S) system call. Note that the timezone value is a system default timezone and not the value of the **TZ** environment variable.

XSEMMAX — Specifies the maximum number of XENIX special semaphores allowed system wide. The minimum value for XSEMMAX is 20, the maximum value is 60, and the default value is 60.

XSDSEGS — Specifies the maximum number of XENIX special shared data segments allowed system wide. The minimum value for XSDSEGS is 1, the maximum value is 25, and the default value is 25.

XSDSLOTS — The maximum number of XENIX special shared data segment attachments system wide is XSDSEGS x XSDSLOTS. The minimum value for XSDSLOTS is 1, the maximum value is 3, and the default value is 3.

DMA Parameters

DMAEXCL — Specifies whether simultaneous DMA requests are allowed. Some computers have DMA chips that malfunction when more than one allocated channel is used simultaneously. For all installations on these computers, DMAEXCL is set to one by default. On computers that do

not suffer from this problem, set DMAEXCL to zero to allow simultaneous DMA on multiple channels.

Modifying an Existing Kernel Parameter

The **stune** command modifies a system tunable parameter from its default value in the *mtune* file. Not every system tunable parameter is contained in the *stune* file. Only those that are to be set to a value other than a system default need be entered there. Although the base UNIX System defines only a few values in *stune*, other add-on packages may have additional added entries into *stune*. Therefore, if the driver package you are building requires modifying a parameter value, the **idtune** command should be used. **idtune** will take individual system parameters, search the *stune* file, and modify an existing value if already there or add a parameter to *stune* if not defined. The value selected must always be within the minimum and maximum values in the *mtune* file.

idtune is particularly useful when preparing software add-on packages that need to modify a system parameter. Since others may have already changed a system parameter, you must be careful when installing your package so that a previous modification is not overwritten. For further information, see the *idtune*(1M) manual page in the *User's/System Administrator's Reference Manual*.

Although it is not recommended that a parameter be set outside the *mtune* limits, if it is determined that a parameter must be set higher than permitted in the *mtune* file, you can edit the limits directly. Extreme care must be taken when modifying *mtune* that other values are not modified or deleted.

You should also be aware that the UNIX System kernel forces some parameters to be within preset limits. For example, the parameter NOFILES (number of open files per user process) is forced to fall within the 20/100 limit regardless of how you adjust the *mtune* and *stune* values. You should never modify an *mtune* value unless you have a full understanding of how the parameter is used in the UNIX System.

Reconfiguring the Kernel to Enable New Parameters

After the *stune* and *mtune* files are modified, the system must be reconfigured using the **/etc/conf/bin/idbuild** command. If a build is required, the system must then be shut down and rebooted. If you are modifying the parameter as part of adding your DSP and your install script already involves **idbuild**, then no additional build is required. The **idbuild** command builds a new UNIX System kernel and sets a lock file that is detected on the next shutdown. The new UNIX System kernel will be linked to **/unix** and executed on the next reboot automatically. The specific steps to modify a parameter are as follows:

1. Modify the */etc/conf/cf.d/stune* file.
2. Execute the **/etc/conf/bin/idtune** command.
3. Execute the **/etc/conf/bin/idbuild** command.
4. Change directories to / and execute **/etc/shutdown**.
5. Reboot the system.

What to Do if the System Does Not Boot

There is a remote possibility that your new kernel will not boot properly. This can happen if a system parameter or some combination of parameters you have modified has built a UNIX System kernel too large to boot, or has built a kernel that will not initialize properly. In such an event, the following steps can be used to recover your system:

1. Insert floppy diskette #1 of your Base System Software.
2. Reset and reboot the system from the floppy diskette.
3. When the prompt "Strike RETURN to install the UNIX System on your hard disk" appears, hit the DEL key to break out of the installation program.

4. Check and mount the hard disk, then copy a good */unix* image with the following commands:

   fsck -y /dev/dsk/0s1
   mount /dev/dsk/0s1 /mnt
   cp /unix /mnt/unix
   umount /dev/dsk/0s1

5. Remove the floppy diskette and reset (use the RESET button or power down/up) the machine to reboot the recovered kernel.

6. Bring up the system as usual, modify */etc/conf/cf.d/stune* parameters back to the original values, and execute the **idbuild** followed by a system reboot sequence.

Chapter 6: All About File Systems

What Is a File System?

A file system consists of directories, files, and special files. These components allow you to structure and maintain a file system to suit your needs. Each time you log in to the UNIX System, you'll be placed in a specific place in the file system structure. From this point, you can move through the levels of the file system to work in any directory or file you own, or to access those files or directories belonging to others that you have permission to use.

You can find the disk space available for each file system on your computer with the **df** command. To use the **df** command, type the following from the UNIX System prompt:

 # df Enter

The file system mount point, special device, available space, and available i-nodes will be displayed for each mounted file system. You can also use the **df** command with a file system argument and display the used and available disk space for that file system. See the *df*(1M) manual page in the *User's/System Administrator's Reference Manual* for additional information.

File System Structure

Every time a file is modified, the UNIX System does a series of file system updates. These updates, when written to the disk, produce a consistent file system. Let's take a look at the components of a file system.

super-block	The super-block defines the internal structure and size of a file system. There is one super-block for each file system.
information nodes (i-nodes)	An i-node is the internal definition of a file or directory. An i-node contains information about the type of file, the number of directory entries linked to the file, a list of blocks claimed by the file, and the size of the file.
data blocks	A data block can contain either directory entries or file data. Each directory entry consists of a filename and an i-node number. Each data block contains 1024 bytes.
indirect blocks	Indirect blocks are needed to reference the data blocks of large files (over 10 blocks long). There are three types of indirect blocks: single-indirect, double-indirect, and triple-indirect.
first free-list block	The free-list blocks are lists of all the blocks not allocated to the super-block, i-nodes, or existing files. The super-block points to the first free-list block.

File System Reliability

The UNIX System is always checking to see if your file systems are in working order. The next few pages will tell you a little about file system reliability and how the UNIX System runs checks on file systems.

File System Integrity

Your computer has several reliability features built in. The following is a brief summary of these features:

- When a file is written to the hard disk, its i-node and blocks are written in an order that ensures maximum reliability. This is known as ordered writes.
- System buffers are periodically written to the hard disk to keep the file contents up to date. This is known as automatic update.
- If the file system becomes corrupted, you will be required to run the **fsck** program to clean up the file system before mounting it. This ensures the reliability of all computer-mounted file systems.

File System Checking and Repair

When the UNIX System is booted, your computer runs a consistency check on the status of the *root* file system. If a potential problem exists, the **fsck** program will run automatically to repair the file system. To ensure that file system inconsistencies are repairable, the **shutdown** command should always be used to bring the system down.

The **fsck** program is a file system check-and-repair program that makes several consistency checks on a specified file system. Because **fsck** runs automatically on the *root* file system, when the system is booted, you should not have to run **fsck** for the *root* file system.

The **fsck** program can also be run manually to check floppy diskettes that have UNIX System file systems on them; or if you suspect something is wrong with your file system, you may want to check it. This should only be attempted by expert users. (See Appendix C for **fsck** error messages.)

File System Corruption

A file system can become corrupt in a variety of ways. Three of the most common ways of corrupting a file system are as follows:

- improper system shutdown and startup
- removing media before unmounting its file system
- hardware failure

Using fsck

To run the **fsck** program manually, the file system must be unmounted with the exception of the *root* file system. The legal **fsck** options are **-b**, **-f**, **-y**, **-n**, **-s**, **-S**, **-t**, **-q**, and **-D**. The **-y** option is recommended for **fsck**. This option answers yes to all questions prompted by **fsck** and requires no intervention by you. Another recommended option is **-s**, which forces rebuilding of the free list in optimal order. The free list becomes disorganized with use. Rebuilding the free list improves performance on subsequently created files. Use the following command line for **fsck**:

 # /etc/fsck -s -y *special* Enter

You'll get a display on your screen similar to the following:

```
/dev/dsk/0s0
File System:    Volume:

**Phase1 - Check Blocks and Sizes
POSSIBLE FILE SIZE ERROR I=321

POSSIBLE FILE SIZE ERROR I=394

**Phase 2 - Check Pathnames
**Phase 3 - Check Connectivity
**Phase 4 - Check Reference Counts
**Phase 5 - Check Free List
 411 files 4394 blocks 8880 free
```

See the *fsck*(1M) manual page in the *User's/System Administrator's Reference Manual* for additional information.

fsck Phases

After the initial setup, **fsck** performs successive phases of tests over the file system, including cleanup and checking blocks and sizes, pathnames, connectivity, reference counts, and the free-block list (possibly rebuilding it).

When an inconsistency is detected, **fsck** reports the error condition to the user. If a response is required, **fsck** will print a prompt message and wait for a response. The following paragraphs explain the meaning of an error condition, the possible responses, and the related error conditions.

The error conditions are organized by the "phase" of the **fsck** program in which they can occur.

For a list and explanation of the error messages you could encounter when using **fsck**, refer to Appendix C.

Phase 1: Check Blocks and Sizes

This phase concerns itself with the i-node list. Activities include checking i-node types, setting up the zero-link-count table, examining i-node block numbers for bad or duplicate blocks, checking i-node size, and checking i-node format.

Phase 1B: Rescan for More DUPS

When a duplicate block is found in the file system, the file system is rescanned to find the i-node that previously claimed that block.

Phase 2: Check Pathnames

This phase concerns itself with removing directory entries pointing to error-conditioned i-nodes from Phase 1 and Phase 1B. Checks are run for root i-node mode and status, directory i-node pointers in range, and directory entries pointing to bad i-nodes.

Phase 3: Check Connectivity

This phase concerns itself with the directory connectivity seen in Phase 2. This part lists error conditions resulting from unreferenced directories and missing or full *lost+found* directories.

Phase 4: Check Reference Counts

This phase concerns itself with the link count information seen in Phase 2 and Phase 3. This part lists error conditions resulting from unreferenced files; missing or full *lost+found* directories; incorrect link count for files, directories, and special files; unreferenced files and directories; bad and duplicate blocks in files and directories; and incorrect total free i-node counts.

Phase 5: Check Free List

This phase concerns itself with the free-block list. This part lists error conditions resulting from bad blocks in the free-block list, bad free-block count, duplicate blocks in the free-block list, unused blocks from the file system not in the free-block list, and the total free-block count incorrect.

Phase 6: Salvage Free List

This phase concerns itself with the free-block list reconstruction. This part lists error conditions resulting from the blocks-to-skip and blocks-per-cylinder values.

Cleanup

Once a file system has been checked, a few cleanup functions are performed. This lists the following advisory messages about the file system and modifies status of the file system:

***** *FILE SYSTEM STATE SET TO OKAY* *****

A flag in the super-block will be set to indicate that the file system is not corrupted and can be mounted.

X files Y blocks Z free

This advisory message indicates that the file system checked contained X files using Y blocks leaving Z blocks free in the file system.

***** FSCK and the ROOT FILE SYSTEM *****

root is the only file system that can (and must) be checked while mounted. Automated mechanisms are provided for checking the *root* file system. These mechanisms handle a dirty *root* when booting and periodic checks during shutdown. You can also force a check on shutdown. These mechanisms hide the messages from **fsck**. If they were not hidden, you would see the error message described next.

***** ROOT FILE SYSTEM WAS MODIFIED *****

This advisory message indicates that the *root* file system was modified by **fsck**. If a system reboot is necessary, **fsck** with the **-b** option forces an automatic reboot and prints the following message:

**** *SYSTEM WILL REBOOT AUTOMATICALLY* ****

If you decide not to use the automated mechanisms, the **-b** option is not used, and a system reboot is necessary, strike RESET.

The automated procedures establish the proper environment (no processes fiddling with files) for checking *root*.

NOTE

Always use the automated procedures for *root*. Never use the **fsck** command on other file systems while they are mounted. If you attempt to use the **fsck** command on a mounted file system other than the *root* file system, the following message is displayed:

```
/dev/dsk/ ?? is a mounted file system, ignored.
?? is the special device name.
```

Recommendations for File System Reliability

There are a few things you can do to try to keep your file systems reliable. The following are some recommendations:

- Never remove a floppy diskette while the disk drive is on (red light on disk drive is on).
- Never remove a mounted UNIX System floppy diskette without unmounting it.
- Always use the **shutdown** procedure before turning off your computer. The shutdown procedure will unmount all file systems.

Recommendations for File System Performance

The UNIX System reads and executes files faster if they are sequential. Initially, the free list of the *root* file system is ordered so new files are sequential, but file creation and/or deletion activity can disorganize the free list. Automated mechanisms are provided that periodically rebuild the free list of the *root* file system. If you have other active file systems on your computer, periodically running **fsck -s** on them when they are unmounted will improve disk performance.

See the *mkfs*(1M) manual page in the *User's/System Administrator's Reference Manual* for additional information.

Bad Block Handling

The requirements of bad block handling fall into six categories:

- dynamic handling of bad blocks
- maintenance of a bad block mapping table
- detection of bad blocks
- mapping of bad blocks
- reporting of bad blocks
- initialization of the hard disk for bad block handling

Dynamic Handling of Bad Blocks

The basic requirement for the bad block handling feature is that it must be done dynamically, without user intervention. Dynamic handling provides immediate attention to the problem and thus minimizes data loss. It also avoids errors that may be introduced by the user. Our current implementation reports problems to the console as they are found without retaining the messages in a log.

Maintenance of a Bad Block Mapping Table

The bad block mapping table is created and stored on the hard disk when the disk is first formatted. It consists of a bad block list of alternate blocks commonly called the "alternate sector list or surrogate images." These two lists are in a one-to-one correspondence.

The bad block list is used to record the address, on disk, of the blocks that are bad. The alternate sector list is used to record the address, on disk, of all the reserved sectors to be used as alternates for bad blocks.

Detection of Bad Blocks

The bad block handling feature should be able to detect two different types of problems.

- marginal blocks
- unreadable blocks

A marginal block is a block that is readable, but with some difficulty. That is, the hard disk controller's Error Correction Code (ECC) algorithm has to be used to successfully read the block.

Once it has been determined that the block in question is marginal, the system will

- report the problem to the user
- copy the data of the marginal block into an available alternate sector
- mark the marginal block as bad
- inform the user that the block has been mapped

Detection of Unreadable Blocks

An unreadable block is a harder problem to solve. There are two possible solutions. One method deals with the possible reconstruction of data to minimize data loss. The other simply accepts that the data is lost.

Reconstruction of data requires that a thorough and extensive analysis of the block in question is done before any kind or form of data repair can be attempted.

While this method offers higher data conservation, its design and implementation will require a considerable amount of time and effort. Implementation of this method did not occur at this time; however, it is being considered as a future extension to the bad block handling feature.

It should be noted that by having an implementation in place that detects and takes care of marginal blocks, the incidence of potentially unreadable blocks is greatly reduced.

Dynamic Handling of Unreadable Blocks

Whenever a block is found to be unreadable, the system will

- inform the user that a bad block has been detected
- assign an available alternate sector to the bad block (whenever appropriate)
- warn the user about the data loss
- determine if the bad block is part of a file system
- if the bad block is part of a file system, mark the file system to enforce a file system check the next time the system is rebooted

Mapping of Bad Blocks

Mapping of bad blocks will be done dynamically by the system without user intervention. As the system finds potential bad blocks or actual bad blocks on the disk, it will inform the user of the occurrence, analyze the block, and take appropriate action.

Reporting of Bad Blocks

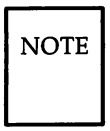
The system will always report the occurrence of potential bad blocks or actual bad blocks. The required error message to be displayed by the system for both types of occurrences are provided below.

Reporting of Marginal Blocks

Whenever a potential marginal or bad block is detected by the system, the block is analyzed to determine the kind of action to be taken. The user is then provided with the information.

The following is a list of possible conditions and the respective messages that will be displayed by the system:

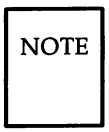
- Potential marginal block detected on the sacred area of the hard disk.



Soft read error corrected by the ECC algorithm: block x drive n.

A potential bad block has been detected (block x on drive n) on a sacred area of the hard disk. If this block goes bad, the UNIX System will be lost. Please backup your system.

- Potential marginal block detected in the UNIX System partition, outside the sacred area of the disk, and the system is out of unassigned alternate sectors.



Soft read error corrected by the ECC algorithm: block x on drive n.

A potential bad block has been detected (block x on drive n). The system is out of spare blocks for surrogates. If the block goes bad, it can not be mapped.

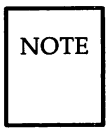
- Potential marginal block detected in the UNIX System partition, outside the sacred area.

Soft read error corrected by the ECC algorithm: block x on drive n.

- Verification of a potential marginal block is started.

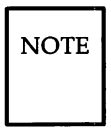
A potential bad block has been detected (block x on drive n). Starting verification to determine if an alternate block needs to be assigned to this block.

- Verification is complete and the block in question is determined to be marginal.



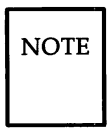
Verification completed. An alternate block will be assigned to block x on drive n.

- Verification completed and the block in question is determined to be a good block.



Verification completed. Block x on drive n is a good block.

- A marginal block has been mapped.



An alternate block has been assigned to block x on drive n.

Reporting Unusable Blocks

The system will always provide the appropriate information whenever an unusable block is detected. The list below include possible conditions together with the respective messages to be displayed by the system.

- Detection of an unusable block on the critical area of the disk.

A bad block (block x on drive n) has been detected on a critical area of the disk. The system can not recover from this failure. Must reinstall the UNIX System and restore from previous backup.

- Detection of an unusable block in the UNIX System partition, outside the critical area of the disk, and the system is out of unassigned alternate sectors.

Block x on drive n is unreadable. Data of this block has been lost.

The system is out of spare blocks for alternates. Block x on drive n can not be mapped.

- Detection of an unusable block on the UNIX System partition outside the critical area of the disk can yield any of three messages.

Block x on drive n is unreadable. Data of this block has been lost.

Requested block x on drive n not found. Data of this block has been lost.

Data from block x on drive n is not readable. Data of this block has been lost.

- An unusable block is being mapped.

An alternate block has been assigned to block x on drive n.

Hard Disk Layout for the UNIX System on the 80386

A description of the current hard disk layout, problems with the layout, and the required changes are provided below. Data structures used only by the UNIX Operating System (i.e., *pdinfo, vtoc,* and the bad block mapping table) are stored on the disk within the UNIX System partition as these pertain to this UNIX System partition and to no other part of the disk. The same

applies to the alternate sectors reserved for bad blocks because these are administered on a per partition basis.

A disk layout for this strategy is as follows:

- Reserve the first sector of cylinder 0 for the primary bootstrap and the *ipart table.*
- Reserve the first 29 sectors of the UNIX System partition for the first-stage and the second-stage bootstrap.
- Reserve the 30th sector of the UNIX System partition for the *pdinfo* and the *vtoc* table.
- Reserve the 31st to the 34th sectors of the UNIX System partition for the bad block mapping table.
- Reserve as many consecutive sectors as needed beginning with the 35th sector of the UNIX System partition for alternate sectors.

The implementation of this layout eliminates potential restrictions on the UNIX System. Consequently:

- Installation of the UNIX System will never cause the destruction of an MS-DOS partition, regardless of where the MS-DOS partition is located on the disk.
- If any of the sectors where *pdinfo, vtoc,* and the bad block mapping table are stored go bad, the hard disk can still be used for the UNIX System, provided that the UNIX System partition starts somewhere else on the disk.

Hard Disk Recovery

Even with bad track handling, it is possible that damage that cannot be repaired automatically could occur to the hard disk (fixed disk). When a hard disk error occurs, you may see a message like the following:

```
A hard disk operation of type x has failed at sector y.
```

The values of **x** are as follows:

0x02 = Failure to read a sector from disk into memory

0x03 = Failure to write a sector from memory onto disk

0x05 = Failure to format specified track

0x06 = Failure to format specified track and set bad sector flag

0x07 = Failure to format drive starting from track x.

If this happens, try to repair the file system using one of the procedures described in the following subsections.

Recovery From Minor Hard Disk Damage

When the hard disk cannot be repaired automatically and you still have access to the system, try the following:

1. Save the contents of the hard disk. (Refer to "Backing up and Restoring Files" in Chapter 4, "System Administration.")
2. If you have the System Test Diagnostics floppy diskette, use it to run a check for bad tracks on the hard disk.
3. Install the UNIX Operating System.

Recovery From Major Hard Disk Damage

When the file system becomes corrupted to the point where the system is inoperable, try the following:

1. Insert the first floppy diskette of the Foundation Set into the floppy disk drive and strike RESET.
2. When you see the message asking if you are ready to install the UNIX System, strike Ctrl and Break at the same time.
3. Run **fsck** from the *root* prompt by typing

   ```
   # /etc/fsck /dev/rdsk/0s1 Enter
   ```

   The **fsck** command will either run with no errors or request action from the user on repairing the file system. Most of the time answering "yes" to the questions ask by **fsck** will be sufficient, but be aware this could remove some files.
4. Remove the diskette and reboot the system by striking RESET.

Recovery of the UNIX System

There may be a time when booting up the computer you will see the message `/unix is missing or corrupted`. If this should occur, you'll need to replace */unix* with the default */unix*. When */unix* is corrupted, the results are unpredictable. In either case, try the following:

1. Insert the first floppy diskette of the Base Foundation Set into the floppy disk drive and strike RESET.
2. When you see the message asking if you are ready to install the UNIX System, break out by striking Ctrl Break at the same time.
3. Run **fsck** from the *root* prompt by typing

   # /etc/fsck /dev/rdsk/0s1 Enter
4. Mount the device **0s1** by typing

   # /etc/mount/ /dev/dsk/0s1 /mnt Enter
5. Copy the */unix* directory by typing

   # cp /unix /mnt/unix Enter
6. Unmount the device **0s1** by typing

   # /etc/umount /dev/dsk/0s1 Enter
7. Reboot the system by striking RESET.

When you get the UNIX System prompt, make sure you are logged in as root. You should then re-install all drivers previously installed. This can be done with the *idbuild* command. See the manual page *idbuild*(1M) in the *User's/System Administrator's Reference Manual*.

Alternate Recovery of the UNIX System

If you have added device drivers or changed configuration, you may want to use this alternate recovery procedure. You can copy */unix* to */unix.orig* and reboot from */unix.orig*. The advantage of doing this is that you do not have to re-install all drivers previously installed. The disadvantage is that */unix.orig* will occupy additional disk space.

If you see that the message */unix* is missing or corrupted, replace */unix* with the backup */unix.orig*. When */unix* is corrupted, try the following:

1. Insert the first floppy diskette of the Base Foundation Set into the floppy disk drive and strike RESET.
2. When you see the message asking if you are ready to install the UNIX System, break out by striking Ctrl Break at the same time.
3. Run **fsck** from the *root* prompt by typing

   # /etc/fsck /dev/rdsk/0s1 Enter
4. Mount the device **0s1** by typing

   # /etc/mount/ /dev/dsk/0s1 /mnt Enter
5. Copy */unix.orig* to */unix* by typing

   # cp /unix /mnt/unix Enter
6. Unmount the device **0s1** by typing

   # /etc/umount /dev/dsk/0s1 Enter
7. Reboot the system by striking RESET When the boot prompt is received, enter

   boot: /unix.orig Enter

Creating Backup Copies and Recovering Lost Files

The value of backing up a file is sometimes not appreciated until it's too late and data is lost. Backing up a system takes time, but recovering data that was not backed up takes much longer. The purpose of system backup is to back up your software on floppy diskettes (or tape) so that you will have it in case data is lost.

Refer to Chapter 4, "System Administration," for procedures on disk backup and restoral.

Creating and Using File Systems

Creating a File System and Making It Available

Once a disk is formatted, the next step is to define the file system. The **mkfs** command is used for this purpose.

NOTE

You must have the 2K File System Utilities package installed to make a 2K file system.

The **mkfs** command is used to create all file systems. One of its optional arguments is the rotational gap. For this computer, the gap should always be 2 (which is the default value). This puts the blocks in ascending order. Thus, new files are more likely to be in sequence and are read faster. Because of this ordering, another optional argument to **mkfs** (cylinder size) is unimportant since you get the same order in all cases.

Using mkfs

The **mkfs** command has two formats:

mkfs special blocks[:i-nodes] [gap blocks/cyl] [–b blocksize]

mkfs special prototype [gap blocks/cyl] [–b blocksize]

Notice that the file system is not given a name in either format; it is identified by the filename of the special device file on which it will reside. The special device file, traditionally located in the directory */dev*, is tied to the identifying controller and unit numbers (major and minor, respectively) for the physical device.

In the first format, the only other information that must be furnished on the **mkfs** command line is the number of 512-byte blocks the file system is to occupy. The second format lets you include that information in a prototype

file that can also define a directory and file structure for the new file system, and it even allows for reading in the contents of files from an existing file system.

Both formats let you specify information about the interrecord gap and the blocks per cylinder. If this information is not given on the command line, default values are used. The recommendations depend on the logical block size of the file system (see the discussion of the **-b** option at the end of this section). The recommended values are different from the defaults used by the command. In the first **mkfs** format, even though the number of blocks in the file is required, the number of i-nodes may be omitted. If the number of i-nodes is omitted, the command uses a default value of one i-node for every four logical storage blocks.

If you use the first format of **mkfs**, the file system is created with a single directory. If you use a prototype file, as noted above, it can include information that causes the command to build and initialize a directory and file structure for the file system. The format of a prototype file is described in the *mkfs*(1M) pages of the *User's/System Administrator's Reference Manual.*

The final option to **mkfs** lets you specify the logical block size to be used for the file system. By default, the file system has a logical block size of 1024 bytes. (With the **-b** option, you can specify a logical block size of 512 bytes, 1024 bytes, or 2048 bytes.)

Choosing Logical Block Size

Logical block size is the size of the chunks the UNIX System kernel uses to read or write files. The logical block size is usually different from the physical block size, which is the size of the smallest chunk that the disk controller can read or write, usually 512 bytes.

An administrator who uses the **mkfs** command to make a file system may specify the logical block size of the file system. By default, the logical block size is 1024 bytes (1K). The *root* and *usr* file systems are delivered as 1K file systems. Besides 1K file systems, the UNIX System also supports 512-byte file systems and 2048-byte (2K) file systems. To use a 2K file system, you must install the 2K file system package.

To choose a reasonable logical block size for your system, you must consider performance and space. For information on file system space requirements, use the file system block analyzer, **fsba**. For most systems, a 1K file system is a good compromise between system performance and use of space

in primary memory and on disk. For a system that uses lots of large executable files and data files, a 2K file system may be a better choice.

Creating a File System on a Floppy Diskette

You can create your own file system on floppy diskettes by using the **mkfs** and **labelit** commands. You will actually be specifying the file system that you want on the floppy disk and then mounting the file system as a directory under the UNIX System. See the **volcopy**(1M) manual page in the *User's/System Administrator's Reference Manual* for additional information on **labelit**.

Creating a file system on floppies can be very useful; that is, you can have portable file systems, and there will be more room on the hard disk. The maximum size of a file system that can be created on a floppy diskette is 702 blocks (512-byte blocks) for a 360 KB floppy diskette and 2370 blocks (512-byte blocks) for a 1.2 MB floppy diskette.

The following steps are used to create and identify a file system on a floppy diskette.

NOTE

You must login as root to do the following procedure. This assures that you have the proper read/write permissions.

1. Login as root.
2. Insert a formatted floppy diskette into the floppy disk drive. Refer to Chapter 4, "System Administration," to learn how to format a floppy diskette. The following are the format command lines for 1.2 MB and 360 KB floppy diskettes:

   For a 1.2 MB floppy, use

   ```
   # /etc/format /dev/rdsk/f0q15dt Enter
   ```

   For a 360-KB floppy, use

   ```
   # /etc/format /dev/rdsk/f0d9dt Enter
   ```

3. If you have a 360 KB floppy disk drive, proceed with the next step. If you have a 1.2 MB floppy disk drive, go to Step 5.

4. If you have a 360 KB floppy diskette, make a file system of 702 blocks and 160 i-nodes using the following command. The rotational gap is 2, and the blocks per cylinder is 18.

   # /etc/mkfs /dev/dsk/f0d9d 702:160 2 18 Enter

   The 360 KB floppy diskette has eighteen 512-byte blocks per cylinder.

5. If you have a 1.2 MB floppy diskette, make a file system of 2370 blocks and 592 i-nodes using the following command. The rotational gap is 2, and the blocks per cylinder is 30.

   # /etc/mkfs /dev/dsk/f0q15d 2370:592 2 30 Enter

   The 1.2 MB floppy diskette has thirty 512-byte blocks per cylinder.

Assume that for the rest of the example you'll be using a 1.2 MB floppy disk. Regardless of what size floppy you have, your screen will look similar to the one below.

NOTE

If the command output in the following screen is not what you want, type Ctrl and Break at the same time to cancel the command.

```
# /etc/mkfs /dev/dsk/f0q15d 702:160 2 18

Mkfs: /dev/dsk/f0q15d?
(strike Del if wrong)
bytes per logical block = 1024
total logical blocks = 1185
total inodes = 592
gap (physical blocks) = 2
cylinder size (physical blocks) = 30
```

6. Label the floppy diskette file system using the **labelit** command. For this example, assume the file system will be called *memo*. The volume name will be **memo2.0**. Type

   # /etc/labelit /dev/dsk/f0q15d memo memo2.0 Enter

The screen will look like the following:

```
/etc/labelit /dev/dsk/f0q15d memo memo2.0

Current fsname: , Current volname:  Blocks: 2370, Inodes: 592
FS Units: 1Kb, Date last modified: Thu Sep 10 13:24:03 1987
NEW fsname = memo, NEW volname = memo2.0 -- [Del] if wrong!!
```

On the computer, DEL refers to the key sequence [Ctrl] [Break] to cancel the process.

7. File systems are usually mounted in *root* (/) as directories. Make a directory in the *root* (/) directory with the same name as the file system you're mounting:

```
# mkdir /memo [Enter]
```

You must be *root* to **mount** or **umount** the UNIX System.

8. Mount the file system as follows:

```
# /etc/mount /dev/dsk/f0q15d /memo Enter
# /etc/mount Enter
```

The following will be displayed:

```
# /etc/mount /dev/dsk/f0q15d /memo
# /etc/mount
/ on /dev/dsk/0s0 read/write on Wed Jun 12 13:30:10 1987
/memo on /dev/dsk/f0q15d read/write on Wed Jun 12 13:34:10 1987
```

The file system */memo* is now associated with a directory in the *root* file system. As long as the *memo* file system is mounted on */memo,* you can create and modify files on it as if it were an extension to the hard disk.

A directory *lost+found* should be created on the file system for use by **fsck**.

Mounting a file system at a directory that does not match the file system name produces a warning message defining what has been mounted. For example, to mount */dev/dsk/f0q15d* (file system name is *memo*) as directory */mnt*, enter the following command line:

```
# /etc/mount /dev/dsk/ f0q15d /mnt Enter
```

The following warning message will be displayed:

```
/etc/mount: warning: <memo> mounted as </mnt>
```

Unmounting a File System

When you have finished using the file system, you can unmount it. This is done with the **umount** command. All files in the file system to be unmounted must be closed, and you must change to a directory not in this file system. For example, if your current directory (**pwd**) is in the file system you want to unmount, you must change out of the file system before executing the **umount** command. Otherwise, you will get the following message:

```
/etc/umount:device busy
```

To unmount a file system from a 1.2MB floppy diskette, type the following:

```
# /etc/umount /dev/dsk/f0q15d Enter
```

If the file system is unmounted cleanly, there will be no need to run **fsck** next time it's mounted. If it does not unmount cleanly, the next attempt to mount it will produce the following error message:

```
mount: possibly damaged or old file system
on /dev/dsk/f0q15d
mount: check file system or mount read only
```

If this should happen, you can do one of two things:

- Run **fsck** on the file system and mount it again as follows:

  # /etc/fsck /dev/dsk/f0q15d Enter

- Mount the file system with read permission only as follows:

  # /etc/mount /dev/dsk/f0q15d /memo *-r* Enter

Root File System Free Space

A predetermined and finite amount of disk space is allocated for the *root* file system. The unoccupied disk space within this area, called free space, allows for additional and temporary files and often serves as a scratch pad for certain system programs. System administration and other types of programs require *root* file system free space to run. It is recommended that you try to avoid using all the space in the *root* file system. If you should run out of space in *root*, the message `no space on Fixed Disk Device 0x1` will be displayed.

If you see this message, you should manually remove the files you do not need from the *root* file system. Since the system creates the file **/etc/mnttab** during start-up time, it is recommended that you save at least 10 free blocks in the *root* file system before shutting down the machine. The command **df** can be used to find out how many free blocks are in your file systems. Refer to the *Programmer's Reference Manual* for information on the *mnttab*(4) manual page. Refer to the *User's/System Administrator's Reference Manual* for information on the *df*(1M) manual page.

Chapter 7: LP Print Service Administration

Introduction

This chapter describes

- The LP print service
- Installation of the LP print service
- Commands used to administer the system
- Stopping and starting the LP print service
- Configuring the LP print service:
    - setting up printer configurations
    - managing the printing load
    - setting job priority limits for users
    - managing pre-printed forms
    - defining filters
- LP print service files and directories
- Writing customized filters and interface programs

How the LP Print Service Works

The LP print service, formerly known as the LP spooler, is a mechanism that allows you to send a file to be printed while you continue doing other work. The term "spool" is an acronym for "simultaneous peripheral output on-line," and "LP" was originally an acronym for Line Printer but has come to include many other types of printing devices. The LP print service system is software that

- Handles the task of receiving files users want printed
- Filters the files (if needed) so they can print properly
- Schedules the work of one or more printers
- Starts programs that interface with the printer(s)
- Keeps track of the status of jobs

- Alerts you to printer problems
- Keeps track of forms currently mounted and alerts you to mount needed forms
- Issues error messages when problems arise

Summary of User Commands

The LP print service has three regular user commands, which are shown in Figure 7-1.

Command	Description
cancel	cancels a request for a file to be printed
lp	sends a file or files to a printer
lpstat	reports the status of the LP system

Figure 7-1: User Commands for the LP Print Service

In addition to being able to send requests to the LP print service system, check the status of requests, and cancel requests, users may be given the ability to disable and enable a printer. The idea is that if a user finds a printer is malfunctioning in some way, it should not be necessary to call the administrator to turn the printer off. On the other hand, it may not be reasonable in your printing environment to allow regular users to disable a printer. You can control whether other users have access to the two commands shown in Figure 7-2.

Command	Description
disable	deactivates the named printer(s)
enable	activates the named printer(s)

Figure 7-2: Privileged User Commands for the LP Print Service

Summary of Administrative Commands

A separate set of commands available for the LP administrator is shown in Figure 7-3. These commands are found in the */usr/lib* directory. If you expect to use them frequently, you might find it convenient to include that directory in your PATH variable. To use the administrative commands, you must be logged in as either *root* or as *lp*. *lp* is a system login (see Chapter 4, "System Administration," for a description of how to set up a password for a system login).

You'll also probably need to use the commands for disabling and enabling a printer and the rest of the commands described under the section "Summary of User Commands" above.

Command	Description
/usr/lib/accept	Permits job requests to be queued for a specified destination.
/usr/lib/reject	Prevents jobs from being queued for a specified destination. Described on the same manual page as *accept*(1M).
/usr/lib/lpadmin	Sets up or changes printer configurations.
/usr/lib/lpfilter	Sets up or changes filter definitions.
/usr/lib/lpforms	Sets up or changes preprinted forms. (Use **/usr/lib/lpadmin** to mount a form.)
/usr/lib/lpmove	Moves output requests from one destination to another. Described on the same manual page as *lpsched*(1M).
/usr/lib/lpsched	Starts the LP print service.
/usr/lib/lpshut	Stops the LP print service. Described on the same manual page as *lpsched*(1M).
/usr/lib/lpusers	Sets or changes the default priority and priority limits the users of the LP print service can request.

Figure 7-3: Administrative Commands for the LP Print Service

In Figure 7-3 the administrative commands are listed in the order in which they appear in the *User's/System Administrator's Reference Manual*. In the sections that follow, we describe the commands in the order in which they are typically used to handle the tasks you'll face as you set up the LP print service to meet your needs.

Starting and Stopping the LP Print Service

Under normal operation, you should never have to start or stop the LP print service manually. It is automatically started each time the UNIX System is started and stopped each time the UNIX System is stopped. However, if you need to stop the LP print service without stopping the UNIX System as well, you can do so by following the procedure described below.

Stopping the LP print service will cause all printing to cease within seconds. Any print requests that have not finished printing will be printed in their entirety after the LP print service is restarted. The printer configurations, forms, and filters in effect when the LP print service is stopped will be restored after it is restarted.

NOTE

To manually start and stop the LP print service, you must be logged in as either the superuser *root* or the user *lp*.

NOTE

Jobs may pass through a printer that is not on line. If a printer is not on line or operating properly, you should disable the printer.

Manually Stopping the Print Service

To manually stop the LP print service, enter the following command:

```
/usr/lib/lpshut
```

The message

```
Print services stopped
```

will appear, and all printing will cease within a few seconds. If you try to stop the LP print service when it is not running, you will see the message

```
Print services already stopped
```

Manually Starting the Print Service

To manually restart the LP print service, enter the following command:

```
/usr/lib/lpsched
```

The message

```
Print services started
```

will appear. It may take a minute or two for the printer configurations, forms, and filters to be re-established before any saved print requests start printing. If you try to restart the LP print service when it is already running, you will see the message

```
Print services already active
```

NOTE The LP print service does not have to be stopped to change printer configurations or to add forms or filters.

Printer Management

Before the LP print service can start accepting print requests, you will have to define the configuration of each printer you have. This section describes how to do that.

Defining the Configuration of a Printer

The following table lists the information that can be given to define the configuration of each printer.

Printer Configuration Information

printer name
interface program
printer type
content types
connection method
printer port characteristics
character sets or print wheels
forms allowed
fault alerting
fault recovery
use restrictions
banner necessary
description
default printing attributes

You need to give very little of this information to add a new printer to the LP print service; however, the more information you provide, the better the printer will be managed for you and the better it will be represented to the people using the LP print service.

The descriptions in the sections below will help you understand what this printer configuration information means and how it is used so that you can decide how to configure your printers. In each section, you will also be shown how to specify this information when adding a printer. While you can follow each of the sections in order and correctly configure a printer in several steps, you may want to wait until you've read all the sections before adding a printer so that you can do it in one step.

Printer Name

The printer name and the connection method (described next) are the only items you must specify to define a new printer. The name is used to identify the printer, both by you when you want to change the printer configuration or manage the printer and by people who want to use the printer to print a file. The name may contain no more than fourteen characters and can include numbers as well as letters, but no special characters other than underscore.

You can choose any names you like, but it is good to choose names that mean something to the users of the LP print service. For example, `laser` is a good name for a laser printer, but if you have several laser printers you may want to number them, such as `laser1`, `laser2`, and so on.

You don't have to try to fit a lot of descriptive information into the name; there is a better place for this information (see the "Description" section below). You also don't have to make the name precisely identify the type of printer—people who need to use a particular type of printer can specify it by type, not name (see the "Printer Type" section below).

You will use the printer name every time you want to refer to the printer: when adding other configuration information for the printer, when changing the configuration of the printer, when referring to the status of the printer, and so on. Thus, the first thing you must do to add a printer is identify its name. You will do this as shown below; but don't do it yet because you'll also need to specify the connection method.

/usr/lib/lpadmin -p *printer-name*

There are no default names; you must name every printer.

Connection Method

The LP print service allows you to connect your printers in a variety of ways. The simplest way is to connect your printer directly to the computer. However, you may want to connect printers via a network or through a dialed modem, where they can be shared with other computers or workstations. Once you've connected the printer to the computer or connected it to a network, and connected the network to the computer, you should describe the connection method for the LP print service.

The default method by which printers are connected to the computer is the direct connection method. If you have used this method to connect your printer to your computer, you generally need to do only one other thing:

name the connecting port. Some directly connected printers, however, can also be used as terminals for login sessions. If you want to use a printer as a terminal, you will have to arrange for the LP print service to handle it as such. To do so, use the **-l** option to the **lpadmin** command, as described below.

There are two methods of making non-direct connections: through a dial-up modem or over any other type of network. The LP print service uses the Basic Networking Utilities to handle both methods of non-direct connections. When a dial-out modem is used, three prerequisites must be satisfied: the printer must be connected via a dialed modem, a dial-out modem must be connected to the computer, and the Basic Networking Utilities must know about this modem.

Printers connected via any other type of network require that a "system name" be given for each printer. This is the name of an entry in the *Systems* file or related file. Although the printer is not a UNIX System, the *Systems* file can still be used to record the access method (no login information will be given, of course).

Because the **cu** program accesses a printer in the same way the LP print service does, you should set up the files as though preparing access to the printer for **cu**. The **cu** command is not used to access printers but can serve as a yardstick when setting up files: if **cu** can access a printer, the LP print service will be able to access it, too. (See Chapter 8, "Basic Networking Administration," for details about setting up network connections.)

Adding a Directly Connected Printer

To add a directly connected printer, type the following:

/usr/lib/lpadmin -p *printer-name* **-v** *path-name*

path-name is the name of the special file representing the printer port. Typically this is one of the following files:

/dev/tty00
/dev/tty01
/dev/lp
etc.

Adding a Printer to be Used as a Login Terminal

To add a directly connected printer to your system for use as a login terminal:

/usr/lib/lpadmin -p *printer-name* **-v** *path-name* **-l**

As before, *path-name* is the name of the special file representing the printer port. The **-l** indicates that the printer should be automatically disabled when the LP print service is started to allow people to log in. The printer/terminal will have to be manually enabled before it can be used for printing. See the section "Enabling and Disabling a Printer" in this chapter for information.

Adding a Printer Connected Via a Modem or Network

To add a printer that is connected via a modem or network:

/usr/lib/lpadmin -p *printer-name* **-U** *dial-info*

dial-info is either the telephone number to be dialed to reach the printer's modem or the system name entered in the Basic Networking *Systems* file for the printer.

You must enter an **lpadmin** command with either the **-U** or **-v** option. And, unless you give the **-l** option, the LP print service will assume the printer is not to be used as a login terminal.

A note on dial-out or network printers: If the printer or port is busy, the LP print service will automatically retry later. This retry rate is 10 minutes if the printer is busy and 20 minutes if the port is busy. The rate is not adjustable. However, you can force an immediate retry by issuing an **enable** command for the printer. If the port or printer is likely to be busy for an extended period, you should issue a **disable** command.

The **lpstat -p** command reports the reason for a failed dial attempt. Also, if you are alerted to a dialing fault (see "Fault Alerting" below), the alert message will give the reason for the fault. These messages are identical to the error messages produced by the Basic Networking Utilities (BNU) for similar problems. See the section called "BNU STATUS Error Messages" in Appendix C ("fsck Error Messages") for an explanation of the reasons for failure.

Interface Program

This is the program the LP print service uses to manage the printer each time a file is printed. It has four main tasks:

- to initialize the printer port (the connection between the computer and the printer)
- to initialize the printer (restore it to a normal state in case a previously printed file has left it in an unusual state) and set the character pitch, line pitch, page size, and character set requested by the user
- to print a banner page
- to run a filter to print the file

If you do not choose an interface program, the standard one provided with the LP print service will be used. This should be sufficient for most of your printing needs. If you prefer, however, you can change it to suit your needs or completely rewrite your own interface program, and then specify it when you add a new printer. See the "Customizing the Print Service" section for details on how to customize an interface program.

If you will be using the standard interface program, you needn't specify it when adding a printer. However, if you will be using a different interface program, you can either refer to it by its full path name or by referring to another printer using the same interface program.

To identify a customized interface program by name, give the printer name and the path name of the interface program as follows:

/usr/lib/lpadmin -p *printer-name* **-i** *path-name*

To identify a customized interface program by reference to another printer, give the printer names as follows:

/usr/lib/lpadmin -p *printer-name*$_1$ **-e** *printer-name*$_2$

printer-name$_1$ should be replaced with the name of the printer you are adding; *printer-name*$_2$ should be replaced with the name of the printer already added that is using the customized interface program.

To identify an interface program by reference to a model interface program, give the printer name and model name as follows:

/usr/lib/lpadmin -p *printer-name* **-m** *model-name*

Printer Type

The printer type is important for the proper use of the printer. The LP print service uses the printer type to extract information about the printer from the Terminfo database. This information describes the capabilities of the printer so that you can be warned if some of the configuration information you provide isn't appropriate for the printer. The information also describes the control data to use to initialize the printer before printing a file. While you are not required to specify a printer type, you are urged to specify one so that better print services will be provided.

The printer type is the generic name for the printer. Typically it is derived from the manufacturer's name, such as `495` for the AT&T 495 Laser Printer. Appendix F in the *User's Guide* provides a description of how to determine a correct TERM variable for a user terminal and can be used as a guide for picking an acceptable name for your printer.

Specify the printer type as follows:

/usr/lib/lpadmin -p *printer-name* **-T** *printer-type*

If you do not define the printer type, the default `unknown` will be used. This will produce empty results when the LP print service looks up information about the printer, so the print service will not be able to verify certain requests or initialize the printer.

Content Types

While the printer type information tells the LP print service what type of printer is being added, the content type information tells the LP print service what types of files can be printed. Most printers can print only one type of file; for them, the content type is likely to be identical to the printer type. Some printers, though, can accept several different types of files and print their contents properly. When adding this kind of printer, you should list the names of the content types it accepts.

When a file is submitted to the LP print service for printing, the print service searches for a printer capable of handling the job. The print service can

identify an appropriate printer through either the content-type name or the printer-type name. Therefore, you may specify either name (or no name) when submitting a file for printing.

Content-type names may look a lot like printer-type names, but you are free to choose names that mean something to you and the people using the printer. (The names `simple`, `terminfo`, or `any` are recognized as having particular meanings by the LP print service; be sure to use them consistently.) The names must contain no more than fourteen characters and may include only letters, digits, and underscores. If the same content type is printable by several different types of printers, you should use the same content type names when you add those printers. This makes it easier for the people using the printers because they can use the same name to identify the type of file they want printed regardless of the printing destination.

For example, several manufacturers may produce printers that accept PostScript files. While these printers may need different printer types so that each can be properly initialized (assuming the initialization control sequences are different), they may all be capable of handling the same type of input file, which you may call, perhaps, `postscript`. As another example, several manufacturers may produce printers that accept ANSI X3.64 defined escape sequences. However, the printers may not support all the ANSI capabilities or may support different sets of capabilities. You may want to give different content-type names for these printers to differentiate them.

You do not have to list the content types for a printer. If you don't, the printer type will be used as the name of the content type the printer can handle. If you have not specified a printer type, the LP print service will assume the printer can print only files of content type `simple`. This may be sufficient if you will require people to pick the proper printer and make sure the files are properly prepared for the printer before they are submitted for printing.

One type of file often encountered on UNIX Systems is called `simple`. This file is assumed to contain just printable ASCII characters and the following control characters:

backspace	moves the carriage back one space except at the beginning of a line
tab	moves the carriage to the next tab stop, which is normally every 8 columns on most printers

linefeed	moves the carriage to the beginning of the next line (may require special port settings for some printers—see the "Printer Port Characteristics" section below)
form feed	moves the carriage to the beginning of the next page
carriage return	moves the carriage to the beginning of the same line (may fail on some printers)

The word "carriage" may be archaic for modern laser printers, but similar actions apply. If a printer can handle a `simple` type of file, you should include it in the content type list when you add the printer and specify the content type(s) the printer can handle. If you don't want a printer to accept files of type `simple`, you must give an alternate list of content types the printer can accept. (The printer type is a good name to use if no other type is appropriate.)

Another content type name is `terminfo`. This doesn't refer to a particular type of file but instead refers to all the types represented in the Terminfo database. It is not likely that any printer is capable of handling all the types listed in the database. However, this name is reserved for describing possible filter capabilities. Likewise, the content type `any` is reserved for describing the types of files a filter can accept or produce. These names should not be used as content types when adding a printer.

Specify the list of content types as follows:

/usr/lib/lpadmin -p *printer-name* **-I** *content-type-list*

The *content-type-list* is a list of names separated by a comma or space. If you use spaces to separate the names, enclose the entire list (but not the **-I**) in quotes. If you do not define the types of files a printer can accept, the LP print service will assume it can take type `simple` and a type with the same name as the printer type (if the printer type is defined).

Printer Port Characteristics

Printers connected directly to computers and those connected over some networks require that the printer port characteristics be set by the interface program. These characteristics define the low-level communications with the

printer. Included are the baud rate; use of XON/XOFF flow control; 7, 8, or other bits per byte; style of parity; and output post-processing. The standard interface program will use the **stty** command to initialize the printer port, minimally setting the baud rate and a few other default characteristics.

The default characteristics applied by the standard interface program are listed below.

Default	Meaning
`9600`	9600 baud rate
`cs8`	8-bit bytes
`-cstopb`	1 stop bit per byte
`-parenb`	no parity generation
`ixon`	enable XON/XOFF flow control
`-ixany`	allow only XON to restart output
`opost`	post-process data stream as listed below:
`-oluc`	don't map lowercase to uppercase
`onlcr`	map linefeed into carriage-return/linefeed
`-ocrnl`	don't map carriage-return into linefeed
`-nocr`	output carriage-returns even at column 0
`nl0`	no delay after linefeeds
`cr0`	no delay after carriage-returns
`tab0`	no delay after tabs
`bs0`	no delay after backspaces
`vt0`	no delay after vertical tabs
`ff0`	no delay after form-feeds

You may find that the default characteristics are sufficient for your printers. However, printers vary enough that you are likely to find that you have to set different characteristics. See the description of the **stty** command in the *User's/System Administrator's Reference Manual* to find the complete list of characteristics.

If you have a printer that requires printer port characteristics other than those handled by the **stty** program, you will have to customize the interface program. See the section "Customizing the Print Service" for help.

When you add a new printer, you can specify an additional list of port characteristics that should be applied when printing each user's file. The list you give will be applied after the default list so that you do not need to include in your list default items that you don't want to change. Specify the additional list as follows:

/usr/lib/lpadmin -p *printer-name* **-o "stty=**'*stty-option-list*'**"**

Note that both the double quotes and single quotes are needed if you give more than one item in the *stty-option-list*. If you do not include alternate printer port characteristics, the default list in the table will be used.

As one example, suppose your printer is to be used for printing graphical data, where linefeed characters should be output alone without an added carriage-return. You would enter the following command:

/usr/lib/lpadmin -p *printer-name* **-o "stty=-onlcr"**

Note that the single quotes are omitted because there's only one item in the list.

As another example, suppose your printer requires odd parity for data sent to it. You would enter the following command:

/usr/lib/lpadmin -p *printer-name* **-o "stty='parenb parodd cs7'"**

Character Sets or Print Wheels

Printers differ in the way they can print in different font styles. Some have changeable print wheels, some have changeable font cartridges, others have preprogrammed, selectable character sets. The LP print service, with your help, can minimize the impact of these differences on the users of the LP print service.

When adding a printer, you can specify what print wheels, font cartridges, or character sets are available with the printer. Only one of these is assumed to apply to each printer. From the point of view of the LP print service, however, print wheels and changeable font cartridges are the same because they require you to intervene and mount a new print wheel or font cartridge. Thus, for ease of discussion, only print wheels and character sets will be mentioned.

When you list the print wheels or character sets available, you will be assigning names to them. These names are for your convenience and the convenience of the users. Because different printers may have similar print wheels or character sets, you should use common names for all printers. This allows a person to submit a file for printing and ask for a particular font style, without regard for which printer will be used or whether a print wheel or selectable character set is used.

If the printer has mountable print wheels, you need only list their names. If the printer has selectable character sets, you need to list their names and map each one into a name or number that uniquely identifies it in the Terminfo database. If you are using UNIX System V/386, Release 3.2 or later, you can use the following command to determine the names of the character sets listed in the Terminfo database:

TERM=*printer-type* **tput csnm 0**

printer-type is the name of the printer type in question. The name of the 0th character set (the character set obtained by default after the printer is initialized) should be printed. Repeat the command, using 1, 2, 3, and so on in place of the 0, to see the names of the other character sets. In general, the Terminfo names should closely match the names used in the user documentation for the printer. However, because not all manufacturers use the same names, the Terminfo names may differ from one printer type to the next.

NOTE

For the LP print service to be able to find the names in the Terminfo database, you must specify a printer type. See the section "Printer Type" above.

To specify a list of print wheel names when adding a printer, enter the following command:

/usr/lib/lpadmin -p *printer-name* **-S** *print-wheel-list*

print-wheel-list is a list of names separated by a comma or space. If you use spaces to separate the names, enclose the entire list (but not the **-S**) in quotes.

To specify a list of character set names and to map them into Terminfo names or numbers, enter the following command:

/usr/lib/lpadmin -p *printer-name* **-S** *character-set-list*

character-set-list is also a list of names separated by a comma or space; however, each item in the list looks like one of the following:

cs*N*=*character-set-name*
character-set-name$_1$=*character-set-name*$_2$

N in the first case is a number from 0 to 63 that identifies the number of the character set in the Terminfo database. *character-set-name*$_1$ in the second case

identifies the character set by its Terminfo name. In either case, the name to the right of the "=" sign is the name you choose as an alias of the character set.

NOTE You do not have to provide a list of aliases for the character sets if the Terminfo names are adequate. You can refer to a character set by number, by Terminfo name, or by your alias.

For example, suppose your printer has two selectable character sets (sets #1 and #2) in addition to the standard character set (set #0). The printer type is 5310. You enter the following commands to determine the names of the selectable character sets:

```
TERM=5310 tput csnm 1
english
TERM=5310 tput csnm 2
finnish
```

The words `english` and `finnish`, the output of the commands, are the names of the selectable character sets. You feel that the name `finnish` is adequate for referring to character set #2, but better names are needed for the standard set and set #1. You enter the following command to define synonyms:

/usr/lib/lpadmin -p *printer-name* **-S** `"cs0=american, english=british"`

If you do not list the print wheels or character sets that can be used with a printer, then the LP print service will assume the following: a printer that takes print wheels has only a single, fixed print wheel, and people cannot ask for a special print wheel when using the printer, and a printer that has selectable character sets can take any **cs***N* name or Terminfo name known for the printer.

Alerting to Mount a Print Wheel

If you have printers that can take changeable print wheels and you have listed the print wheels allowed on each then users will be able to submit a print request to use a particular print wheel. However, until it is mounted (see "Mounting a Form or Print Wheel" in this section), a request for a print wheel will stay queued and will not be printed. You could periodically monitor the number of print requests pending for a particular print wheel, but the

LP print service provides an easier way. You can ask to be alerted when the number of requests waiting for a print wheel has exceeded some threshold.

You can choose one of several ways to receive an alert:

- You can receive an alert via electronic mail. See the description of the **mail** command in the *User's/System Administrator's Reference Manual.*
- You can receive an alert written to whatever terminal on which you are logged in. See the description of the **write** command in the *User's/System Administrator's Reference Manual.*
- You can receive an alert through a program of your choice.
- You can receive no alerts.

NOTE

If you elect to receive no alerts, you are responsible for checking whether the proper print wheel is mounted.

In addition to the method of alerting, you can also set the number of requests that must be queued before you are alerted, and you can arrange for repeated alerts every few minutes until the print wheel is mounted. You can choose the rate of repeated alerts, or you can choose to receive only one alert per print wheel.

To arrange for alerting to the need to mount a print wheel, enter one of the following commands:

/usr/lib/lpadmin -S *print-wheel-name* **-A mail -Q** *integer* **-W** *minutes*
/usr/lib/lpadmin -S *print-wheel-name* **-A write -Q** *integer* **-W** *minutes*
/usr/lib/lpadmin -S *print-wheel-name* **-A** *'command'* **-Q** *integer* **-W** *minutes*
/usr/lib/lpadmin -S *print-wheel-name* **-A none**

The first two commands direct the LP print service to send you a mail message or write the message directly to your terminal, respectively, for each alert. The third command directs the LP print service to run *command* for each alert. The shell environment currently in effect when you enter the third command is saved and restored for the execution of *command;* this includes the environment variables, user and group IDs, and current directory. The fourth command above directs the LP print service to never send you an alert when the print wheel needs to be mounted. *integer* is the number of requests that need

to be waiting for the print wheel, and *minutes* is the number of minutes between repeated alerts.

NOTE

If you want mail sent or a message written to another person when a printer fault occurs, you'll have to use the third command listed. Use the option **-A 'mail** *user-name'* or **-A 'write** *user-name'*

Once you start receiving repeated alerts, you can direct the LP print service to stop sending you alerts for the current case by giving the following command:

/usr/lib/lpadmin -S *print-wheel-name* **-A quiet**

Once the print wheel has been mounted and unmounted again, alerts will start again if too many requests are waiting. Alerts will also start again if the number of requests waiting falls below the **-Q** threshold and then rises up to the **-Q** threshold again, as when waiting requests are canceled or if the type of alerting is changed.

If *print-wheel-name* is **all** in any of the commands above, the alerting condition will apply to all print wheels for which an alert has already been defined.

If you don't define an alert method for a print wheel, you will not receive an alert for it. If you do define a method but don't give the **-W** option, you will be alerted once for each occasion.

Forms Allowed

NOTE

For a description of forms, see the "Forms" section in this chapter.

You can limit the use of preprinted forms on any printer. You may want to do this, for instance, if a printer is not well suited for printing on a particular form because of low print quality or if the form cannot be lined up properly in the printer.

The LP print service will use the list of forms allowed or denied for a printer to warn you against mounting a denied form on the printer. However, you have the final word on this; the LP print service will not refuse such an attempt. The LP print service will, however, refuse a user's request to print a file on a printer using a form denied on that printer unless the form is already mounted.

If you try to list a form as allowed on a printer but the printer does not have sufficient capabilities to handle the form, the command will be rejected.

The method of listing the forms allowed or denied for a printer is similar to the method used to list those users allowed or denied access to the **cron** and **at** facilities. See the description of the **crontab** command in the *User's/System Administrator's Reference Manual*. Briefly, the rules are as follows:

1. An allow list is a list of forms that you are allowed to use with the printer. A deny list is a list of forms for which you are denied permission to use with the printer.
2. If the allow list is not empty, the deny list is ignored. If the allow list is empty, the deny list is used. If both lists are empty, there are no restrictions on which forms can be used.
3. Putting "any" or "all" into the allow list allows all forms; putting "any" or "all" into the deny list denies all forms.

You can add names of forms to either list using one of the following commands:

/usr/lib/lpadmin -p *printer-name* **-f allow:***form-list*
/usr/lib/lpadmin -p *printer-name* **-f deny:***form-list*

form-list is list of names of forms separated by a comma or space. If you use spaces to separate names, enclose the entire list (including the **allow:** or **deny:** but not the **-f**) in quotes. The first command adds names to the allow list and removes them from the deny list. The second command adds names to the deny list and removes them from the allow list. To make use of all forms permissible, specify **allow:all**; to deny permission for all forms, specify **deny:all**.

If you do not add forms to the allow list or deny list, the LP print service will consider that the printer denies the use of all forms. It will, however, allow you to mount any form. It will also provide a warning message if the form is not in the allow list or if you are attempting to mount a form that doesn't match the capabilities of the printer as described earlier.

Fault Alerting

The LP print service provides a framework for detecting printer faults and alerting you. Faults can range from simple problems, such as running out of paper or ribbon or needing to replace the toner, to more serious faults, such as a local power failure or printer failure. The range of fault indicators is also broad, ranging from dropping carrier (the signal that indicates that the printer is on line), to sending an XOFF, to sending a message. Only two classes of printer fault indicators are recognized by the LP print service itself: a drop in carrier and an XOFF not followed in reasonable time by an XON. However, you can add filters that can recognize any other printer fault indicators and rely on the LP print service to alert you to a fault when the filter detects it.

NOTE

For a description of how to add a filter, see the "Filter Management" section in this chapter. For a description of how a filter should let the LP print service know a fault has occurred, see the "Customizing the Print Service" section in this chapter.

You can choose one of several ways to receive an alert to a printer fault:

- You can receive an alert via electronic mail. See the description of the **mail** command in the *User's/System Administrator's Reference Manual.*
- You can receive an alert written to the terminal on which you are logged in (any terminal). See the description of the **write** command in the *User's/System Administrator's Reference Manual.*
- You can receive an alert through a program of your choice.
- You can receive no alerts.

NOTE

If you elect to receive no alerts, you will need a way of finding out about the faults and fixing them; the LP print service will not continue to use a printer that has a fault.

In addition to the method of alerting, you can also arrange for repeated alerts every few minutes until the fault is cleared. You can choose the rate of repeated alerts, or you can choose to receive only one alert per fault.

NOTE

Without a filter that provides better fault detection, the LP print service cannot automatically determine when a fault has been cleared except by trying to print another file. It will assume that a fault has been cleared when it is successfully able to print a file. Until that time, if you have asked for only one alert per fault, you will not receive another alert. If after you have fixed a fault, but before the LP print service has tried printing another file, the printer faults again, or if your attempt to fix the fault did not succeed, you will not be notified. Receiving repeated alerts per fault or requiring manual re-enabling of the printer (see the "Fault Recovery" section below) will overcome this problem.

To arrange for alerting to a printer fault, enter one of the following commands:

/usr/lib/lpadmin -p *printer-name* **-A mail -W** *minutes*
/usr/lib/lpadmin -p *printer-name* **-A write -W** *minutes*
/usr/lib/lpadmin -p *printer-name* **-A** *'command'* **-W** *minutes*
/usr/lib/lpadmin -p *printer-name* **-A none**

The first two commands direct the LP print service to send you a mail message or write the message directly to your terminal, respectively, for each alert. The third command directs the LP print service to run *command* for each alert. The shell environment currently in effect when you enter the third command is saved and restored for the execution of *command*. The environment includes environment variables, user and group IDs, and current directory. The *minutes* is the number of minutes between repeated alerts. The fourth command above directs the LP print service to not send you an alert when a fault occurs.

NOTE

If you want mail sent or a message written to another person when a printer fault occurs, use the third command. Use the option

-A 'mail *user-name'* or **-A 'write** *user-name'*

Once a fault occurs and you start receiving repeated alerts, you can direct the LP print service to stop sending you alerts for the current fault by giving the following command:

/usr/lib/lpadmin -p *printer-name* **-A quiet**

If *printer-name* is **all** in any of the commands above, the alerting condition will apply to all printers.

If you don't define an alert method, you will receive mail once for each printer fault. If you do define a method but don't give the **-W** option, you will be alerted once for each fault.

Fault Recovery

Once a printer fault has been detected and you have been alerted, you will probably fix the fault and get the printer ready for printing. When the printer is ready for printing again, the LP print service will recover in one of three ways:

- continue printing at the top of the page where printing stopped
- restart printing at the beginning of the print request that was active when the fault occurred
- wait for you to tell the LP print service to re-enable the printer

NOTE

The ability to continue printing at the top of the page where printing stopped requires the use of a filter that can wait for a printer fault to be cleared before resuming properly. Such a filter probably has to have detailed knowledge of the control sequences used by the printer so it can keep track of page boundaries and know where in a file printing stopped. The default filter used by the LP print service cannot do this. If a proper filter is not being used, you will be notified in an alert if recovery cannot proceed as you want.

To specify the way the LP print service should recover after a fault has been cleared, enter one of the following commands:

/usr/lib/lpadmin -p *printer-name* **-F continue**
/usr/lib/lpadmin -p *printer-name* **-F beginning**
/usr/lib/lpadmin -p *printer-name* **-F wait**

These direct the LP print service, respectively, to continue at the top of the page, restart from the beginning, or wait for you to enter an **enable** command to re-enable the printer (see the "Enabling and Disabling Printer" section in this chapter for information on the **enable** command).

If you do not specify how the LP print service is to resume after a printer fault, it will try to continue at the top of the page where printing stopped, or failing that, at the beginning of the print request.

If the recovery is **continue** but the interface program does not stay running so that it can detect when the printer fault has been cleared, printing will be attempted every few minutes until it succeeds. You can force the LP print service to retry immediately by issuing an **enable** command.

Restricting User Access

You can limit the use of a printer to a subset of all people on your computer. You may want to do this, for instance, if a printer is being set aside for printing sensitive information and only a subset of the people can print sensitive information or if use of a high quality printer incurs expenses not all people are allowed to incur.

The LP print service will use the list of users allowed or denied for a printer to restrict use of the printer. The LP print service will refuse a user's request to print a file on a printer he or she is not allowed to use.

The method of listing the users allowed or denied for a printer is similar to the method used to list users allowed or denied access to the **cron** and **at** facilities and the method described above in the "Forms Allowed" section. Briefly, the rules are as follows:

1. An allow list is a list of those users allowed to use the printer. A deny list is a list of those users denied access to the printer.
2. If the allow list is not empty, the deny list is ignored. If the allow list is empty, the deny list is used. If both lists are empty, there are no restrictions on who can use the printer.

3. Putting "any" or "all" into the allow list allows everybody to use the printer; putting "any" or "all" into the deny list denies everybody, except the user **lp** and the superuser **root**.

You can add names of users to either list using one of the following commands:

/usr/lib/lpadmin -p *printer-name* **-u allow:***user-list*
/usr/lib/lpadmin -p *printer-name* **-u deny:***user-list*

user-list is a list of names of users separated by a comma or space. If you use spaces to separate the names, enclose the entire list (including **allow:** or **deny:** but not the **-u**) in quotes. The first command adds the names to the allow list and removes them from the deny list. The second command adds the names to the deny list and removes them from the allow list. Using **allow:all** will allow everybody; using **deny:all** will deny everybody.

If you do not add user names to the allow or deny lists, the LP print service will assume that everybody can use the printer.

Banner Necessary

Usually you will want to have each print request preceded with a banner page. The banner page shows you who requested the printing, the request ID, and when it was printed, and allows for an optional title that the requester can use to better identify the printout. The banner page greatly eases the task of separating a sequence of print requests so that each can be given to the correct user or placed in separate bins.

Sometimes a user needs to avoid printing a banner page. The likely occasions are when the printer has forms mounted that should not be wasted, such as payroll checks or accounts payable checks. Printing a banner page in such occasions may cause problems.

Enter the following command to allow a user to skip the banner page:

/usr/lib/lpadmin -p *printer-name* **-o nobanner**

If you later change your mind, you can remove this choice by entering the following command.

/usr/lib/lpadmin -p *printer-name* **-o banner**

If you do not allow a user to skip the banner page, the LP print service will reject all attempts to avoid a banner page when printing on the printer. This is the default action.

Description

You can add a description of a printer that can give people using the LP print service helpful information about the printer. This description can contain any message you'd like, including a room number where the printer is found, who to call with printer problems, and so forth.

You can see the message when you use the **lpstat -D -p** *printer-name* command.

To add a description when adding a printer, enter the following command:

/usr/lib/lpadmin -p *printer-name* **-D** *'text'*

text is the message. You'll need to include the quotes if the message contains blanks or other characters that the shell might interpret if the quotes are left out.

Unless you give a printer description, none will be presented to people who ask about it.

Default Printing Attributes

When a user submits a request to print a file, the page size, character pitch, and line pitch (i.e., print spacing) are normally determined from the form that will be printed on. If the user doesn't require a form, he or she can give the page size and print spacing to use. However, if he or she gives neither a form to use nor the page size and print spacing, defaults will be used.

You can set the defaults for each printer. This can also serve to make submitting a print request easier, by designating different printers as having different default page sizes or print spacing. Users then simply route their file to the appropriate printer to get the style output they want. For example, you can have one printer dedicated to printing wide (132 column) output, another printing normal (80 column by 66 lines) output, yet another printing letter quality (12 characters per inch, 8 lines per inch).

You can independently specify four default settings: page width, page length, character pitch, and line pitch. You can scale these to fit your needs. The first two can be given in columns and lines, inches, or centimeters. The last two can be given as characters and lines per inch or per centimeter. In addition, the character pitch can be specified as `pica` for 10 characters per inch (cpi), `elite` for 12 cpi, or `compressed` for the maximum cpi the printer can provide (up to a limit of 30 cpi).

Set the defaults using one or more of the following commands:

/usr/lib/lpadmin -p *printer-name* **-o width=***scaled-number*
/usr/lib/lpadmin -p *printer-name* **-o length=***scaled-number*
/usr/lib/lpadmin -p *printer-name* **-o cpi=***scaled-number*
/usr/lib/lpadmin -p *printer-name* **-o lpi=***scaled-number*

Add the letter `"i"` to *scaled-number* to indicate inches, or the letter `"c"` to indicate centimeters. The letter `"i"` for character pitch (cpi) or line pitch (lpi) is redundant. You can also give `pica`, `elite`, or `compressed` instead of a number for the character pitch.

If you don't provide defaults, the page size and print spacing will be those available when the printer is initialized. You can find out what the defaults will be by first defining the printer configuration without providing your own defaults, then using the **lpstat** program to display the printer configuration. The command

lpstat -p *printer-name* **-l**

will report the default page size and print spacing. If you have not provided the defaults, the reported defaults will be calculated from the Terminfo database entry for the printer. Obviously, this requires you to have provided a printer type in the printer configuration.

Adding a Printer to a Class

It is occasionally convenient to treat a collection of printers as a single class. The benefit is that a person can submit a file for printing by a member of a class, and the LP print service will pick the first printer in the class that it finds free. This allows faster turn-around, as printers are kept as busy as possible.

Classes aren't needed if the only purpose is to allow a user to submit a print request by type of printer. The **lp -T** *type* command allows a user to submit a file and specify its type. The first available printer that can handle the type of file will be used to print the file. The LP print service will avoid using a filter, if possible, by choosing a printer that can print the file directly over one that would need it filtered first.

NOTE

See the "Filter Management" section of this chapter for more information about filters.

Classes do have uses, however. One use is to put into a class a series of printers that should be used in a particular order. If you have a high-speed printer and a low-speed printer, for instance, you probably want the high-speed printer to handle as many print requests as possible, with the low-speed printer reserved for use when the other is busy. Because the LP print service always checks for an available printer in the order the printers were added to a class, you could add the high-speed printer to the class before the low-speed printer and let the LP print service route print requests in the order you wanted.

Add a printer to a class using the following command:

/usr/lib/lpadmin -p *printer-name* **-c** *class-name*

If the class *class-name* doesn't exist yet, it will be created.

NOTE

Class names and printer names must be unique. This allows a user to specify the destination for a print request without having to know whether it's a class of printers or a single printer. Thus, you can't have a class and printer with the same name.

Until you add a printer to a class, it won't belong to any.

Setting the System Default Destination

You can define the printer or class to be used to print a file when the user has not explicitly asked for a particular destination and has not set the **LPDEST** shell variable. The printer or class must already exist first.

Make a printer or class the default destination by entering the following command:

/usr/lib/lpadmin -d *printer-or-class-name*

If you later decide that there should be no default destination, enter a null *printer-or-class-name* as in the following command:

/usr/lib/lpadmin -d

If you don't set a default destination, there will be none. Users will have to explicitly name a printer or class in each print request, or will have to set the **LPDEST** shell variable with the name of a destination.

Mounting a Form or Print Wheel

NOTE

See the "Forms" section in this chapter for information about pre-printed forms.

Before the LP print service will start printing files that need a pre-printed form or print wheel, you will have to mount it on a printer. If alerting has been set on the form or print wheel, you will be alerted when enough print requests are queued waiting for it to be mounted.

When you mount a form, you may wish to see if it is lined up properly. If an alignment pattern has been registered with the form, you can ask that this be repeatedly printed after you've mounted the form, until you have adjusted the printer so that the alignment pattern looks correct.

Mounting a form or print wheel involves first loading it onto the printer and then telling the LP print service that it is mounted. Because it is difficult to do this on a printer that's currently printing and because the LP print service will continue to print files not needing the form on the printer, you will probably have to disable the printer first. Thus, the proper procedure is to follow these three steps:

1. Disable the printer using the **disable** command.
2. Mount the new form or print wheel as described below.
3. Re-enable the printer using the **enable** command. (The **disable** and **enable** commands are described in the "Enabling and Disabling a Printer" section of this chapter.)

When you've loaded the new form or print wheel into the printer, enter the following command to tell the LP print service to mount it. (This command is shown on two lines for readability; it must be entered as one line.)

/usr/lib/lpadmin -p *printer-name* **-M -S** *print-wheel-name* **-f** *form-name* **-a -o filebreak**

Leave out **-S** *print-wheel-name* if you are mounting just a form, or leave out the **-f** *form-name* **-a -o filebreak** if you are mounting just a print wheel.

If you are mounting a form, you will be asked to press the return key before each copy of the alignment pattern is printed. After the pattern is printed, you can adjust the printer and press the return key again. If no alignment pattern has been registered, you won't be asked to press the key. You can drop the **-a** and **-o filebreak** options if you don't want to bother with the alignment pattern.

The **-o filebreak** option tells the LP print service to add a "formfeed" after each copy of the alignment pattern. The actual control sequence used for the "formfeed" depends on the printer involved and is obtained from the Terminfo database. If the alignment pattern already includes a formfeed, leave out the **-o filebreak** option.

If you want to unmount a form or print wheel, use the following command:

/usr/lib/lpadmin -p *printer_name* **-M -S none -f none**

Leave out **-S none** if you just want to unmount a form; likewise, leave out **-f none** if you just want to unmount a print wheel.

Until you've mounted a form on a printer, only print requests that don't require a form will be sent to it. Likewise, until you've mounted a print wheel on a printer, only print requests that don't require a particular print wheel will be sent to it.

Removing a Printer or Class

You can remove a printer or class if it has no pending print requests. If there are pending requests, you have to first move them to another printer or class using the **lpmove** command, or remove them using the **cancel** command.

Removing the last remaining printer of a class automatically removes the class as well. However, the removal of a class does not cause the removal of printers that were members of the class. If the printer or class removed is also the system default destination, the system will no longer have a default destination.

To remove a printer or class, enter the following command:

/usr/lib/lpadmin -x *printer-or-class-name*

If all you want to do is remove a printer from a class but not delete the printer, enter the following command:

/usr/lib/lpadmin -p *printer-name* **-r** *class-name*

Putting it All Together

When adding a new printer, you can do it in separate steps, entering the commands described above. However, you may find it easier to enter one or two commands that combines all the necessary arguments. Below are some examples.

Example 1

Add a new printer called `lp1` on printer port */dev/tty00*. It should use the standard interface program, with the default page size of 90 columns by 71 lines, and linefeeds should not be mapped into carriage return/linefeed pairs.

```
/usr/lib/lpadmin -p lp1 -v /dev/tty00 -T 455 -o
"width=90 length=71 stty=-onlcr"
```

(The preceding line is split into two lines for readability in this document.

Example 2

Add a new printer called *laser* on printer port */dev/tty01*. It should use a customized interface program, it can handle three file types—*i10, i300,* and *impress*—and only the users *doceng* and *docpub* may use it.

```
/usr/lib/lpadmin -p laser -v /dev/tty01 -i /usr/doceng
      /laser_intface -I "i10,i300,impress" -u
                "allow:doceng,docpub"
```

(The preceding line is split into three lines for readability in this document.)

Example 3

When adding the `lp1` printer in the first example, we forgot to set the alerting. We can do this now. We'll have the LP print service alert us every ten minutes after a fault until we fix the problem.

```
/usr/lib/lpadmin -p lp1 -A write -W 10
```

Accepting Print Requests for a New Printer

Initially, the LP print service will not consider a new printer eligible for printing files. This will give you time to make sure you've defined the printer configuration the way you want. When you are ready to make the printer available to others, you'll have to tell the LP print service.

There are two steps in making a printer ready for use after you've defined the printer configuration. First, the LP print service will have to be told to accept print requests for the new printer. Second, the new printer will have to be enabled to print. These are separate tasks because you may have occasion to want to do one but not the other.

Telling the LP print service to accept print requests for the new printer is done with the **accept** command. You will read more about this command in a later section, "Managing the Printing Load." For now, all you need to know is that you should enter the following command to let this printer be used:

/usr/lib/accept *printer-or-class-name*

As you can see, this command is needed to let the LP print service start accepting print requests for a class, too.

Enabling and Disabling a Printer

When a printer is ready for use and the LP print service is accepting print requests for it, you will have to enable it before anything will print. You will use the **enable** command to do this. Having the LP print service wait for you instead of automatically starting to print files lets you make sure that the correct form is loaded in the printer, that the correct print wheel or font cartridge is in place, and that the printer is on-line.

When all is ready, you should enter the following command to enable printing on a printer.

enable *printer-name*

Only printers are enabled for printing—not classes. If you want to enable several printers at one time, list the printers separated by spaces on the same line as the **enable** command. Don't enclose the list in quotes.

At some point you may have to disable a printer. This should be done before you change the form or print wheel, or whenever you wish to stop what's currently printing. Disabling a printer stops further print requests from being printed, but it will not stop the LP print service from accepting new print requests for the printer. Normally, disabling a printer will also stop the request that's currently printing, placing it back in the queue so it can be printed later. However, you can have the LP print service wait until the current request finishes or even cancel the request outright.

Enter one of the following commands to disable a printer:

disable -r "*reason*" *printer-name*
disable -W -r "*reason*" *printer-name*
disable -c -r "*reason*" *printer-name*

The first command disables the printer, stopping the currently printing request and saving it for printing later. The other commands also disable the printer, but the second will have the LP print service wait for the current request to finish, while the third will cancel the current request. The *reason* will be stored and displayed whenever anyone asks the status of the printer. You can leave it (and the **-r**) out if you don't want to give a reason.

Several printers can be disabled at once by listing their names in the same line as the **disable** command.

Allowing Users to Enable and Disable a Printer

You may want to make the **enable** and **disable** commands available for use by other people. This is useful, for instance, if you have a small organization where anyone who spots a problem with the printer should disable it and fix the problem. This is not a good idea if you want to keep others from interfering with the proper operation of the print services.

If you want to allow others access to the **enable** and **disable** commands, you make use of a standard UNIX feature called the "setuid bit." By having these commands owned by the user *lp* (they should be already) and by setting the setuid bit, anyone will be allowed to use the files. Clearing the bit again removes this privilege.

To allow everybody to use these commands, enter the following two commands:

```
chown lp /usr/bin/enable /usr/bin/disable
chmod u+s /usr/bin/enable /usr/bin/disable
```

The first command makes the user *lp* the owner of the commands; this should be redundant, but it is safer to run the command anyway. The second command turns on the setuid bit.

To prevent others from using these commands, enter the following command:

```
chmod u-s /usr/bin/enable /usr/bin/disable
```

Examining a Printer Configuration

Once you've defined a printer configuration, you probably want to review it to see if it is correct. If after examining the configuration you find you've made a mistake, just re-enter the command that applies to the part that's wrong.

You'll use the **lpstat** command to examine both the configuration and the current status of a printer. A short form of this command gives just the status; you can use it to see if the printer exists and if it is busy, idle, or disabled. A long form of the command adds the complete configuration.

Enter one of the following commands to examine a printer:

lpstat -p *printer-name*
lpstat -p *printer-name* **-l**

The second command is the long form. With either command you should see one of the following lines of output:

```
printer printer-name now printing request-id. enabled
since date.

printer printer-name is idle. enabled since date.

printer printer-name disabled since date.
        reason

printer printer-name waiting for auto-retry.
        reason
```

The `waiting for auto-retry` output shows that the LP print service failed in trying to use the printer (because of the *reason* shown) and that the print service will try again later.

With the long form of the command, you should also see the following output:

```
Form mounted: form-name
Content types: content-type-list
Printer type: printer-type
Description: comment
Connection: connection-info
Interface: path-name
On fault: alert-method
After fault: fault-recovery
Users allowed:
        user-list
Forms allowed:
        form-list
Banner required
Character sets:
        character-set-list
Default pitch: integer CPI, integer LPI
Default page size: scaled-decimal-number wide,
scaled-decimal-number long
Default port settings: stty-option-list
```

Trouble Shooting

If you are having difficulty getting your printer to work, here are a few suggestions for what to do.

No Output - Nothing Prints

The printer is sitting idle; nothing happens. First, check the documentation that came with the printer to see if there is a self-test feature you can invoke; make sure the printer is working before continuing.

Is the Printer Connected to the Computer?

Check to make sure that the printer is connected to the printer. Refer to your printer's owners manual for installation instructions.

Is the Printer Enabled?

The printer must be "enabled" in two ways. First, the printer must be turned on and ready to receive data from the computer. Second, the LP print service must be ready to use the printer. Set up the printer as described in the "Printer Management" section of this chapter. If you receive error messages when doing this, follow the "fixes" suggested in the messages. When you've finished setting up the printer, issue the commands

/usr/lib/accept *printer-name*
enable *printer-name*

where *printer-name* is the name you assigned to the printer for the LP print service. Now submit a sample file for printing:

lp -d *printer-name* **-T** *printer-type file-name*

If you didn't give a printer type for the printer, leave out the **-T** *printer-type* option.

Is the Baud Rate Correct?

If the baud rate (the rate at which the computer sends data to the printer) is not matched with the printer, sometimes nothing will print. See below.

Illegible Output

The printer tries printing, but it is not what you expected and certainly isn't readable.

Is the Baud Rate Correct?

Usually when the baud rate isn't matched with the printer, you'll get some output but it will not look at all like what you submitted for printing. Random characters will appear with an unusual mix of special characters and unlikely spacing.

Read the documentation that came with the printer to find out what its baud rate is. It should probably be set at 9600 baud for optimum performance, but that doesn't matter for now. If it isn't set to 9600 baud, you can have the LP print service use the correct baud rate (by default it uses 9600). If the printer is connected via a parallel port, the baud rate doesn't matter.

To set a different baud rate for the LP print service to use, enter the following command:

/usr/lib/lpadmin -p *printer-name* **-o stty=***baud-rate*

Now submit a sample file for printing (explained earlier in this section).

Is the Parity Setting Correct?

Some printers use a "parity bit" to ensure that the data they receive for printing has not been garbled in transmission. The parity bit can be encoded in a few different ways, and both the computer and the printer must agree on which to use. If they do not agree, some characters will not be printed or will have another character substituted. Generally, though, the output will look approximately correct, with the spacing of "words" typical for your document and many letters in their correct place.

Check the documentation for the printer to see what it expects. If your printer is directly connected to the computer with a fairly short wire (50 feet or so), it doesn't have to use the parity bit, but it doesn't matter for now. The LP print service will not expect to set the parity bit by default. You can change this, however, by entering one of the following commands:

/usr/lib/lpadmin -p *printer-name* **-o stty=oddp**
/usr/lib/lpadmin -p *printer-name* **-o stty=evenp**
/usr/lib/lpadmin -p *printer-name* **-o stty=-parity**

The first command sets odd parity generation, the second sets even parity. The last command sets the default, no parity. Select the command that matches what your printer needs.

If you are also setting a baud rate other than 9600, combine the baud rate setting with the parity settings, as in the sample command below:

/usr/lib/lpadmin -p *printer-name* **-o "stty='evenp 1200'"**

Both double and single quotes are needed.

Tabs Set Correctly?

If the printer doesn't expect to receive tab characters, the output may be there but all of it is jammed up against the right margin. See below.

Correct Printer Type?

See below under "Wrong Character Set or Font."

Legible Printing, but Wrong Spacing

The output is all there, it's readable, but is double spaced, has no left margin, is run together, or zig-zags down the page. These problems can be fixed by adjusting the printer settings (if possible) or having the LP print service use matching settings.

Double Spaced

Either the **-onlcr** or **-tabs** option is needed.

/usr/lib/lpadmin -p *printer-name* **-o stty=-onlcr**
/usr/lib/lpadmin -p *printer-name* **-o stty=-tabs**

No Left Margin/Runs Together/Jammed Up

The **-tabs** option is needed.

/usr/lib/lpadmin -p *printer-name* **-o stty=-tabs**

Zig Zags Down the Page

The **onlcr** option is needed. This is set by default, but you may have cleared it accidentally.

/usr/lib/lpadmin -p *printer-name* **-o stty=onlcr**

A Combination of Problems

If several of these options must be combined to take care of multiple problems, include them in one list as in the sample command below. Include any baud rate or parity settings, too.

/usr/lib/lpadmin -p *printer-name* **-o "stty='-onlcr**
-tabs 2400'"

Both double and single quotes are needed.

Correct Printer Type?
See below.

Wrong Character Set or Font

If the wrong printer type was selected when you set up the printer with the LP print service, the wrong "control characters" can be sent to the printer. The results are unpredictable and may cause output to disappear or be illegible, making it look like a problem described above. A simpler problem is that it sets the wrong character set or font.

If you don't know what printer type to give, try the following to examine the available printer types. First, if you think the printer type has a certain name, try the following command:

TERM=*printer-type* **tput longname**

(This may not work on early versions of UNIX System V.) The output of this command will appear on your terminal and is a short description of the printer identified by the *printer-type*. Try the names you think might be right until you find one that identifies your printer.

If you don't know what names to try, you can examine the */usr/lib/terminfo* directory to see what names are available. Warning: There are probably many names in that directory. Enter the following command to examine the directory:

```
ls -R /usr/lib/terminfo/*
```

Pick names from the list that match one word or number identifying your printer. For example, the name `495` would identify the AT&T 495 Printer. Try each of the names in the other command above.

When you have the name of a printer type you think is correct, set it in the LP print service by entering the following command:

/usr/lib/lpadmin -p *printer-name* **-T** *printer-type*

Dial Out Failures

The LP print service uses the Basic Networking Utilities to handle dial out printers. If a dialing failure occurs and you are receiving printer fault alerts, the LP print service reports the same error reported by the Basic Networking software for similar problems. (If you haven't arranged to receive fault alerts, by default they are mailed to the user **lp**.)

Idle Printers

There are several reasons why you may find a printer idle and enabled but with print requests still queued:

- The print requests need to be filtered. Slow filters run one at a time to avoid overloading the system. Until a print request has been filtered (if it needs slow filtering), it will not print. Use the following command to see if the first waiting request is being filtered:

  ```
  lpstat -o -l
  ```

- The printer has a fault. Automatic continuation of printing after a fault has been detected does not happen immediately. The LP print service will wait about five minutes before trying again and will keep trying until a request prints successfully. You can force a retry immediately by enabling the printer:

  enable *printer-name*

- A dial-out printer was busy or didn't answer, or all dial-out ports are busy. As with automatic continuation after a fault, the LP print service waits five minutes before trying to reach a dial-out printer again. If the dial-out printer can't be reached for an hour or two (depending on the reason), the LP print service will finally alert to a possible problem. You can force a retry immediately by enabling the printer:

  enable *printer-name*

- Lost "child process." If the UNIX process controlling the printer is killed (by the UNIX System during periods of extremely heavy load or by an administrator), the LP print service may not realize it for a few minutes. Disabling the printer and then re-enabling it again will force the LP print service to check for the controlling process and restart one. Make sure the printer is really idle, though, because disabling a printer stops it in the middle of printing a request. Though the request will not be lost, it will have to be reprinted in its entirety.

  disable *printer-name*
  enable *printer-name*

  If the process that is lost is one controlling a slow filter, don't try re-enabling the printer; instead, put the print request (the one at the head of the queue for the printer) on hold and then resume it, as shown

below:

lpstat -o -l
lp -i *request-id* **-H hold**
lp -i *request-id* **-H resume**

Use the first command to list the requests queued.

Managing the Printing Load

Occasionally, you may need to stop accepting print requests for a printer or move print requests from one printer to another. There are various reasons why you might want to do this, such as the following:

- The printer needs periodic maintenance.
- The printer is broken.
- The printer has been removed.
- You've changed the configuration so that the printer is to be used differently.
- Too many large print requests are queued for one printer and should be spread around.

If you are going to make a big change in the way a printer is to be used, such as stopping its ability to handle a certain form, changing the print wheels available for it, or disallowing some people from using it, print requests that are currently queued for printing on it will have to be moved or canceled. The LP print service will attempt to find alternate printers but only if the user doesn't care which printer is to be used. Such requests won't be automatically moved; if you don't move them first, the LP print service will cancel them.

If you decide that a printer is to be taken out of service, its configuration is to be changed, or it is too heavily loaded, you may want to move print requests off it and reject additional requests for it for awhile. You'll use the **lpmove** and **reject** commands for this. If you do reject requests for a printer, you can later accept requests using the **accept** command.

Rejecting Requests for a Printer or Class

To stop accepting any new requests for a printer or class of printers, enter the following command:

/usr/lib/reject -r "*reason*" *printer-or-class-name*

You can reject requests for several printers or classes in one command by listing their names on the same line, separating the names with spaces. The *reason* will be displayed whenever anyone tries to print a file on the printer. You can drop it (and the **-r**) if you don't want to give a reason.

Although the **reject** command stops any new print requests from being accepted, it will not move or cancel any requests currently queued for the printer. These will continue to print as long as the printer is enabled.

Accepting Requests for a Printer or Class

After the condition that led to denying requests has been corrected or changed, enter the following command to start accepting new requests:

/usr/lib/accept *printer-or-class-name*

Again, you can accept requests for several printers or classes in one command by listing their names on the same line.

You will always have to use the **accept** command for a new printer or class after you have added it because the LP print service does not initially accept requests for new printers or classes.

Moving Requests to Another Printer

If you have to move requests from one printer or class to another, enter one of the following commands

/usr/lib/lpmove *request-id printer-name*
/usr/lib/lpmove *printer-name*$_1$ *printer-name*$_2$

You can give more than one request ID before the printer name in the first command.

The first command above moves the listed requests to the printer named. The latter command moves *all* requests currently queued for the first printer to the second printer. When the latter command is used, the LP print service will also no longer accept requests for the first printer (this is the same effect as the **reject** command).

Examples

Here are some examples of how you might use these three commands:

Example 1

You've decided it is time to change the ribbon on printer `lp1` and perform some preventive maintenance. You'll want to move all the requests for printer `lp1` to printer `lp2`. After the requests are moved, the LP print service will no longer accept requests for `lp1` (this is the same effect as a **reject lp1** command issued after the **lpmove** command).

```
/usr/lib/lpmove lp1 lp2
```

(At this point you may disable the printer and start working on it.)

Example 2

You've finished changing the ribbon and the other work on `lp1`, so now you want to bring it back into service.

```
/usr/lib/accept lp1
```

(At this point, if you had disabled the printer you should re-enable it. See the "Enabling and Disabling a Printer" section under "Printer Management" in this chapter.)

Example 3

You notice that someone has queued several large files for printing on the printer `laser1`. Meanwhile `laser2` is currently idle because no one had queued requests for it. You'll move the two biggest requests, `laser1-23` and `laser1-46` to `laser2`, and reject any new requests for `laser1` for the time being.

```
/usr/lib/lpmove laser1-23 laser1-46 laser2
/usr/lib/reject -r "too busy--will reopen later"
laser1
```

Managing Queue Priorities

The LP print service provides a simple priority mechanism that people can use to adjust the position of a print request in the queue. Each print request can be given a priority level by the person who submits it; this is a number from 0 to 39, with *smaller* numbers indicating *higher* levels of priority. Requests with higher priority (smaller numbers) are placed ahead of requests with lower priority (larger numbers).

In this way, if you decide that your print request is of too low priority, you can set a higher priority (lower value) when you submit the file for printing. If you decide that your print request is of too high priority, you can set a lower priority (higher value) when she submits the file for printing.

A priority scheme this simple would not work if there were no controls on how high one can set the priority. You can define the following characteristics of this scheme:

- Each user can be assigned a priority limit. One cannot submit a print request with a priority higher than his or her limit, although one can submit a request with a lower priority.
- A default priority limit can be assigned for the balance of users not assigned a personal limit.
- A default priority can be set. This is the priority given print requests to which the user does not assign a priority.

By setting the characteristics according to your needs, you can prevent lower priority printing tasks (such as regular printing by most staff members) from interfering with higher priority printing tasks (such as payroll check printing by the accounting staff).

You may find that you want a critical print request to print ahead of any others, perhaps even if it has to pre-empt the currently printing request. You can have the LP print service give "immediate" handling to a print request and can have it put on "hold" another print request. This will let the first print request print and will delay that latter print request until you have it "resumed."

The **lpusers** command lets you assign both priority limits for users and priority defaults. In addition, you can use the **lp -i** *request-id* **-H hold** and **lp -i** *request-id* **-H immediate** commands to put a request on hold or move it up for immediate printing, respectively. These commands are discussed in detail below.

Setting Priority Limits

To set a user's priority limit, enter the following command:

/usr/lib/lpusers -q *priority-level* **-u** *user-name*

You can set the limit for a group of users by listing their names after the **-u** option. Separate multiple names with a comma or space (enclose the list in quotes if you use a space, though). *priority-level* is a number from 0 to 39. As mentioned before, the lower the number, the higher the priority or in this case the priority limit.

If you want to set the priority limit for all other users, enter the following command:

/usr/lib/lpusers -q *priority-level*

This sets the default limit; the default applies to those people who have not been given a personal limit using the earlier **lpusers** command.

If you later decide that someone should have a different priority limit, just re-enter the first command above with a new limit. If you decide that someone with a personal limit should have just whatever the default limit is, enter the following command:

/usr/lib/lpusers -u *user-name*

Again, you can do this for more than one person at a time by giving a list of names. Using the **lpusers** command with just the **-u** option puts the users in the "default limit" category.

If you do not set a default limit, people without personal limits will be limited to priorities in the range of 20 to 39.

Setting a Default Priority

You can set the default priority that should be assigned to those print requests submitted without a priority. Use the following command:

/usr/lib/lpusers -d *priority-level*

Don't confuse this default with the "default limit." This default is applied when a user doesn't give a priority; the "default limit" is applied if you haven't assigned a limit for a user—it is used to limit the user from giving too high a priority.

NOTE If the default priority is greater than the limit for a user, the limit is used instead.

If you do not set a default priority, the LP print service will use the default of 20.

Examining the Priority Limits and Defaults

You can examine all the settings you have assigned for priority limits and defaults by entering the following command:

```
/usr/lib/lpusers -l
```

Moving a Request Around in the Queue

Once a user has submitted a print request, you can move it around in the queue to some degree:

- You can adjust the priority to any level regardless of the limit for the user
- You can put it on hold and let other requests print ahead of it
- You can put it at the head of the queue for immediate printing.

You'll use the regular **lp** user command to do each of these.

Changing the Priority for a Request

Print requests that are still waiting to print can be reassigned a new priority. This will reposition it in the queue to put it ahead of lower priority requests, behind any others at the same or higher priority. The priority limit assigned to the user (or the default priority limit) has no effect because you will override this limit as the administrator.

Enter the following command to change the priority of a request:

lp -i *requestid* **-q** *new-priority-level*

You can change only one request at a time with this command.

If a request is already printing, you cannot change its priority.

Putting a Request on Hold

Any request that has not finished printing can be put on hold. You can stop its printing, if it currently is printing, and keep it from printing until you resume it. A user can also put his or her own request on hold and then resume it, but cannot resume a print request you have put on hold.

Enter the following command to place a request on hold:

lp -i *request-id* **-H hold**

Enter the following command to resume the request:

lp -i *request-id* **-H resume**

Once resumed, a request will continue to move up the queue and will eventually print. If it had been printing when you held it, it will be the next request to print. Normally it will start printing from the beginning, with page one, but you can have it start printing at a later page. Enter the following command to resume the request at a different page:

lp -i *request-id* **-H resume -P** *starting-page-*

The final dash is needed to specify the starting page and all subsequent pages.

NOTE

The ability to print a subset of pages requires the presence of a filter that can handle this. The default filter used by the LP print service does not. An attempt to resume a request on a later page will be rejected if an appropriate filter is not being used.

Moving a Request to the Head of the Queue

You can move a print request to the head of the queue where it will be the next one eligible for printing. If it must start printing immediately but another request is currently printing, you can hold the other request as described above.

Enter the following command to move a print request to the head of the queue:

lp -i *request-id* **-H immediate**

Only you can move a request like this; regular users cannot use the **-H immediate** option.

NOTE

If you set more than one request for immediate printing, they will print in the reverse order set; that is, the request moved to the head of the queue most recently will print first.

Forms

This section tells you how you can manage the use of preprinted forms with the LP print service. You will see how you can

- define a new form
- change an old form
- remove a form
- examine a form
- restrict user access to a form
- arrange alerting to the need to mount a form
- mount a form

But before getting into the details, let's see what a form means in the context of the LP print service.

What is a Form?

A preprinted form is a paper image of a blank form that you can load into your printer. An application will typically generate a file that, when printed on the blank form, will fill out the form. Common examples of forms are

- blank checks
- vouchers
- receipts
- labels
- company letterhead
- special paper stock

Typically, several copies of the blank form are loaded into the printer either as a tray of single sheets or as a box of fan-folded paper.

The LP print service helps you manage the use of preprinted forms but does not provide your application any help in filling out a form. This is solely your application's responsibility. The LP print service, however, will keep track of which print requests need special forms mounted and which forms are currently mounted, and can alert you to the need to mount a new form.

Of course, if you don't use special forms for printing, you can skip this section.

Defining a Form

The first thing you have to do to add a new form is define its characteristics. This is a short list that helps the LP print service remind you how to deal with the form and tells the LP print service how to initialize the printer to print properly on the form. You'll need to know the following about the form:

Page length
: The length of the form or of each page in a multi-page form. This can be expressed as the number of lines, or the size in inches or centimeters.

Page width
: The width of the form expressed in columns, inches, or centimeters.

Number of pages
: The number of pages in a multi-page form.

  The LP print service uses this number with a filter (if available) to restrict the alignment pattern to be a single form long. (See the description of alignment patterns below.) If no filter is available to truncate the alignment pattern, the LP print service will skip that step.

Line pitch
: How close together separate lines are on the form. This is expressed in either lines per inch or lines per centimeter.

Character pitch
: How close together separate characters are on the form. Similarly, this is expressed in either characters per inch or characters per centimeter.

Character set choice
: The character set, print wheel, or font cartridge that should be used when this form is used. A user can choose a different character set for his or her own print request using this form, or you can insist that only one character set be used.

Ribbon color — If the form should always be printed using a certain color ribbon, then the LP print service can remind you which color to use when you mount the form.

Comment — Any comment you wish to make about the form. This comment is available for people to see so they can understand what the form is, when it should be used, and so on.

Alignment pattern — A sample file that the LP print service will use to fill one blank form. When mounting the form, you can examine this sample to see if the printing is lined up properly on the form. If it isn't, you can adjust the printer to get it lined up.

NOTE

The LP print service will not try to mask sensitive information in an alignment pattern. If you do not want sensitive information printed on sample forms – very likely the case when you align checks, for instance – then you should mask the appropriate data. The LP print service will keep the alignment pattern stored in a safe place, where only you (i.e., the user **lp** and the superuser **root**) can read it.

When you've gathered this information about the form, you'll enter it as input to the **lpforms** command. You may want to first record this information in your own file to make it easier to edit the information as you enter it. You can then give the file as input instead. However you enter it, you should present the information in the following way:

```
Page length: scaled-number
Page width: scaled-number
Number of pages: integer
Line pitch: scaled-number
Character pitch: scaled-number
Character set choice: character-set-name,mandatory
Ribbon color: ribbon-color
Comment: comment
Alignment pattern: alignment-pattern
```

Except for the `Alignment pattern`, the information can appear in any order (the *comment* must follow the `Comment:` line, though). The `Alignment pattern` must be the last information given. If the *comment* has to contain a line beginning with any of the key phrases (`Page length`,

`Page width`, and so on), you should precede it with a ">" character to hide the key phrase. This means, though, that any initial ">" will be stripped from the comment when it is displayed.

Not all of the information has to be given. Missing information will be assigned the following defaults:

Item	Default
Page length	66 lines
Page width	80 columns
Number of pages	1
Line pitch	6
Character pitch	10
Character set choice	any
Ribbon color	any
Comment	(no default)
Alignment pattern	(no default)

Use one of the following commands to define the form.

/usr/lib/lpforms -f *form-name* **-F** *file-name*
/usr/lib/lpforms -f *form-name* **-**

The first command gets the form definition from a file; the second command gets the form definition from you through the standard input. *form-name* can be anything you choose as long as it contains fourteen or fewer letters, digits, and underscores.

If you need to change a form, just re-enter one of the same commands. You need only give the changed information; information you leave out will stay the same.

Removing a Form

The LP print service has no fixed limit to the number of forms you can define. However, it is a good idea to remove forms no longer appropriate to avoid confusing the users, who would otherwise see a long list of obsolete forms when they are trying to choose the correct form, and to avoid extra processing by the LP print service, which must occasionally look through all the forms to perform certain tasks.

Enter the following command to remove a form:

/usr/lib/lpforms -f *form-name* **-x**

Restricting User Access

You can limit the use of a form to a subset of people on your computer. You may want to do this, for instance, with sensitive forms such as checks, which only people in the payroll department or accounts payable department can use.

The LP print service will use the list of users allowed or denied for a form to restrict use of the form. The LP print service will refuse a user's request to print a file with a form he or she is not allowed to use.

The method of listing the users allowed or denied access to a form is similar to the method used to list users allowed or denied access to the **cron** and **at** facilities. See the description of the **crontab** command in the *User's/System Administrator's Reference Manual*. Briefly, the rules are as follows:

1. An allow list is a list of those users allowed to use the form. A deny list is a list of those users denied access to the form.
2. If the allow list is not empty, the deny list is ignored. If the allow list is empty, the deny list is used. If both lists are empty, there are no restrictions on who can use the form.
3. Putting **any** or **all** into the allow list allows everybody to use the form; putting **any** or **all** into the deny list denies everybody, except the user *lp* and the superuser *root*.

You can add names of users to either list using one of the following commands:

/usr/lib/lpforms -f *form-name* **-u allow:***user-list*
/usr/lib/lpforms -f *form-name* **-u deny:***user-list*

The *user-list* is a list of names of users separated by a comma or space. If you use spaces to separate the names, enclose the entire list (including the **allow:** or **deny:** but not the **-u**) in quotes. The first command adds the names to the allow list and removes them from the deny list. The second command adds the names to the deny list and removes them from the allow list. Using **allow:all** will allow everybody; using **deny:all** will deny everybody.

If you do not add user names to the allow or deny lists, the LP print service will assume that everybody can use the form.

Alerting to Mount a Form

If you define more forms than printers, you will obviously not be able to print files on all the forms simultaneously. This means that some print requests may be held in the queue until you mount the forms they need. You could periodically monitor the number of print requests pending for a particular form, but the LP print service provides an easier way: You can ask to be alerted when the number of requests waiting for a form has exceeded some threshold.

You can choose one of several ways to receive an alert:

- You can receive an alert via electronic mail. See the description of the **mail** command in the *User's/System Administrator's Reference Manual.*
- You can receive an alert written to whatever terminal on which you are logged in. See the description of the **write** command in the *User's/System Administrator's Reference Manual.*
- You can receive an alert through a program of your choice.
- You can receive no alerts.

NOTE

If you elect to receive no alerts, you are responsible for checking to see if any print requests haven't printed because the proper form isn't mounted.

In addition to the method of alerting, you can also set the number of requests that must be queued before you are alerted, and you can arrange for repeated alerts every few minutes until the form is mounted. You can choose the rate of repeated alerts, or can choose to receive only one alert per form.

To arrange for alerting to the need to mount a form, enter one of the following commands:

/usr/lib/lpforms -f *form-name* **-A mail -Q** *integer*
-W *minutes*
/usr/lib/lpforms -f *form-name* **-A write -Q** *integer*
-W *minutes*
/usr/lib/lpforms -f *form-name* **-A** *'command'* **-Q** *integer*
-W *minutes*
/usr/lib/lpforms -f *form-name* **-A none**

The first two commands direct the LP print service to send you a mail message or write the message directly to your terminal, respectively, for each alert. The third command directs the LP print service to run *command* for each alert. The shell environment currently in effect when you enter the third command is saved and restored for the execution of *command*; this includes the environment variables, user and group IDs, and current directory. The fourth command above directs the LP print service not to send you an alert when the form needs to be mounted. *integer* is the number of requests that need to be waiting for the form, and *minutes* is the number of minutes between repeated alerts.

NOTE

If you want mail sent or a message written to another person when a printer fault occurs, you'll have to use the third command listed. Use the option **-A 'mail** *user-name'* or **-A 'write** *user-name'*

Once you start receiving repeated alerts, you can direct the LP print service to stop sending you alerts for the current case only by giving the following command:

/usr/lib/lpforms -f *form-name* **-A quiet**

Once the form has been mounted and unmounted again, alerts will start again if too many requests are waiting. Alerts will also start again if the number of requests waiting falls below the **-Q** threshold and then rises up to the **-Q** threshold again, as when waiting requests are canceled or if the type of alerting is changed.

If *form-name* is `all` in any of the commands above, the alerting condition will apply to all forms.

If you don't define an alert method for a form, you will not receive an alert for it. If you do define a method but don't give the **-W** option, you will be alerted once for each occasion.

Mounting a Form

Refer to the "Mounting a Form or Print Wheel" section under "Defining the Configuration of a Printer" in this chapter.

Examining a Form

You can examine a form definition once you've added it to the LP print service. There are two commands to use, depending on the information you want to examine. The **lpforms** command will display the definition of the form. The display will be suitable as input again, so that you can save the output in a file for future reference. The **lpstat** command will display the current status of the form.

Enter one of the following commands to examine a defined form:

/usr/lib/lpforms -f *form-name* **-l**
/usr/lib/lpforms -f *form-name* **-l** >*file-name*
lpstat -f *form-name*
lpstat -f *form-name* **-l**

The first two commands present the definition of the form; the second command captures this definition in a file, which can later be used to redefine the form if you inadvertently remove the form from the LP print service. The last two commands present the status of the form, with the second of the two giving a long form of output similar to the output of **lpforms -l**; their output looks like the following:

```
Page length: scaled-number
Page width: scaled-number
Number of pages: integer
Line pitch: scaled-number
Character pitch: scaled-number
Character set choice: character-set,mandatory
Ribbon color: ribbon-color
Comment: comment
Alignment pattern: content-type content
```

The `Alignment pattern` is not shown if the **lpstat** command is used to protect the potentially sensitive content.

Filter Management

This section tells you how you can manage the use of filters with the LP print service. You will see how you can

- define a new filter
- change a filter
- remove a filter
- examine a filter

The "Customizing the Print Service" section at the end of this chapter will describe how you can write a filter. First, let's see what a filter is and how the LP print service can use one.

What is a Filter?

A filter plays three related roles:

- It converts a user's file into a data stream that will print properly on a given printer.
- It handles the various modes of printing that people may request with the **-y** option to the **lp** command, such as two-sided printing, landscape printing, draft or letter quality printing, etc.
- It detects printer faults and informs the LP print service so that the latter can alert you.

Not every filter will perform all three roles. However, given the printer-specific nature of these three roles, the LP print service has been designed so that these roles are separated out so that you, a printer manufacturer, or another source can provide these filters without having to change the LP print service.

A default "filter" is provided with the LP print service to provide simple printer fault detection; it does not convert files nor handle any of the special modes. This may be adequate for your needs.

Let's examine the three roles more closely.

Converting Files

The LP print service allows you to "type" each printer you add to the system and allows a user to "type" each file he or she submits for printing. This type information is used to match a file with the printer that will reproduce the file the best. Since many applications can generate data for various printers, this is often sufficient. However, not all of the applications you will use may generate output that will work on your printers.

By defining and creating a filter that will convert such output into a type that your printers can handle, you can begin to support more applications in the LP print service. The Terminal Filters Utilities provide a small set of simple filters that convert output from applications like **nroff** (from the Documenter's Workbench Utilities) to data streams that print properly on some printers.

Each filter that is added to the system is "typed" as well, with the input type it can accept and the output type it can produce. Now the LP print service can be more sophisticated in its attempt to match a user's file with a printer. If it cannot find a direct match, it will consult the table of filters to find one that will convert the file's type into the printer's type. Below are some examples.

Example 1

The user Chris has run a spreadsheet program and generated a file copy of a spreadsheet. Chris now wants to print this file using the LP print service. You have only AT&T model 455 printers on your system. Fortunately, the spreadsheet application understands how to generate output for several printers, and Chris knew to ask it to generate the file for the AT&T 455. When Chris submits the file for printing, the LP print service will queue it for one of the printers; no filter is needed.

Example 2

The user Marty has run the **nroff** word processing program to produce a copy of a large document. The **nroff** program also understands how to generate output for several printers, but Marty forgot and had it generate the default output type (let's call it type **nroff35**) which won't reproduce well on the AT&T 455. However, you had foreseen this situation and added the `450` filter to the filter table, marked it as taking standard nroff output (i.e., **nroff35**) and marked it as producing output for the AT&T 455 (let's call it type `455`). Since you added the printer as a type `455`, the LP print service recognizes that it can use the `450` filter to convert Marty's output before printing it.

Handling Special Modes

Another important role that filters can provide is the handling of the various printing modes that may be encountered. Each filter you add to the filter table can be registered as handling several aspects of printing. These are listed in the table below:

Input type
Output type
Printer type
Character pitch
Line pitch
Page length
Page width
Pages to print
Character set
Form name
Number of copies
Modes

A filter is not required to handle most of these, only the modes. The LP print service provides a default handling for all the rest. However, it may be more efficient to have a filter handle these, or it may be that a filter has to know several of these aspects if it is to fulfill its other roles properly. A filter may need to know, for example, the page size and the print spacing if it is going to break up the pages in a file to fit on the printed pages. As another example, some printers can handle multiple copies more efficiently than the LP print service will, so a filter that is controlling the printer can use the number of copies information to skip the LP print service's default handling of this.

We'll see below how you register the printing modes and other aspects of printing with each filter.

Detecting Printer Faults

Just as converting a file and handling special printing modes is a printer-specific role, so is the detecting of printer faults. The LP print service attempts to do this in general, and for most printers it will properly detect a fault. However, it is limited to checking for "hang-ups" (loss of carrier or the signal that indicates the printer is on-line) and excessive delays in printing (i.e., receipt of an XOFF flow-control character to shut off the data flow with no matching XON to turn the flow back on). It also can't determine the cause of the fault, so it can't tell you what to look for.

A properly designed filter can provide better fault coverage. Some printers are able to send a message to the host describing the reason for a fault. Others indicate a fault by other than dropping carrier or shutting off data flow. A filter can serve you by giving more information about a fault and detecting more of them.

Another benefit a filter can give is to wait for a printer fault to clear and to resume printing. This allows for more efficient printing when a fault occurs because the print request that was interrupted does not have to be reprinted in its entirety. Only a real filter, which will have knowledge of the control sequences used by a printer, can know where a file breaks into pages; thus, only the filter will know how far back to go in the file to restart properly.

The LP print service has a simple interface that lets the filter get the fault information to you and restart if it can. The alerting mechanism (see the "Fault Alerting" section under "Printer Management" in this chapter) is handled by the LP print service; the interface program that manages the filter takes all error messages from the filter and places them into an alert message that can be sent to you. Thus, you'll see any fault descriptions that the filter puts out. If you've set the printer configuration so that printing should automatically resume after a fault is cleared, the interface program will keep the filter active so that it can pick right up where it left off.

Will Any Program Make a Good Filter?

It is tempting to use a program like **troff**, **nroff**, or a similar word-processing program as a filter. However, the **troff** and **nroff** programs have a feature that allows people to reference additional files in the source document; these are called "include files." The LP Spooler does not know about these files and will not enqueue them with the source document. The **troff** or **nroff** program may fail because it cannot access these additional files. Other programs may have similar features that limit their use as filters.

Here are guidelines that will help you choose a good filter:

1. Examine the kinds of files people will submit for printing that will have to be processed by the filter. If they stand alone, that is, if they do not reference other files that the filter will need, the filter is probably okay. Check also to see if the filter expects any other files except those submitted by a user for printing.

2. If there can be referenced files inside the files submitted for printing or if the filter will need files other than those submitted by a user, then the filter is likely to fail because it will not be able to access the additional files. We suggest you don't use the program as a filter but have each user run the program before submitting the files for printing.

Referenced files that are always given with full path names *may* be okay but only if the filter is used for local print requests. When used on requests submitted from a remote machine for printing on your machine, the filter may still fail if the referenced files are only on the remote machine.

Defining a Filter

There are several aspects of a filter that you have to define for the LP print service. These are listed below:

Input types — This is the list of file types that the filter can process. Most filters can take only one input type, but the LP print service doesn't restrict them to one. Several file types may be similar enough for the filter that it can deal with them. You can use whatever names you like here, subject to a limit of 14 letters, digits, and dashes (no underscore). Because the LP print service will use these names to match a filter with a file type, you should be consistent in the naming convention. For example, if more than one filter can accept the same input type, use the same name.

These names should be advertised to your users so they know how to name their file's type when they submit the file for printing.

Output types — This is the list of "file" types that the filter can produce as output. For each file, the filter will produce a single output type but it may be able to vary that type on demand. The names of the output types are also restricted to 14 letters, digits, and dashes.

These names should either match the types of printers you have on your system or should match the input types handled by other filters. The LP print service will gang filters together in a shell pipeline to produce a new filter if it finds that

several passes by different filters are needed to convert a file. It's unlikely that you will need this level of sophistication, but the LP print service allows it. Try to find a set of filters that take as input types all the different files your users may want printed and that convert those files directly into types your printers can handle.

Printer types

This is a list of printer types into which the filter can convert files. While for most filters this list will be identical to the output types, it can be different.

For example, you may have a printer that is given a single type for purposes of initialization (see the "Printer" section under "Printer Management" in this chapter) but which can recognize several different types of files. In essence, these printers have an internal filter that converts the various types into one with which they can deal. Thus, a filter may produce one of several output types that match the "file types" that the printer can handle. The filter should be marked as working with that printer type.

As another example, you may have two different models of printers that are listed as both accepting the same types of files. However, due to slight differences in manufacture, one printer deviates in the results it produces. You label the printers as being of different printer types, say A and B, where B is the one that deviates. You create a filter that adjusts files to account for the deviation produced by printers of type B. Since this filter is only needed for those printer types, you would list it as working only on type B printers.

For most printers and filters, you can leave this part of the filter definition blank.

Printers

You may have some printers that, although they're of the correct type for a filter, are in other ways not adequate for the output that the filter will produce. For instance, you may want to dedicate one printer for fast turn-around; only files that the printer can handle without filtering will be sent to that printer. Other printers, of identical type, you'll allow to

be used for files that may need extensive filtering before they can be printed. You'll label the filter as working with only the latter printers.

For most cases, the filter should be able to work with all printers that accept the output that the filter produces, so you can leave this part of the filter definition blank.

Filter type — The LP print service recognizes "fast" filters and "slow" filters. Fast filters are labeled "fast" either because they incur little overhead in preparing a file for printing or because they must have access to the printer when they run. A filter that is to detect printer faults has to be a fast filter. Slow filters are the opposite. Filters that incur a lot of overhead in preparing a file and that don't have to have access to the printer should be labeled "slow." The LP print service will run slow filters in the background without tying up a printer. This allows files that need at most fast filtering (or no filtering) to move ahead; printers will not be left idle while a slow filter works on a file if other files can be printed.

Command — This is the full path name of the program to run; this is the filter. If there are any fixed options that the program will always need, you can include them here.

Options — Options that the filter program will need, depending on the various modes and other aspects of printing, can be registered with the filter. This is discussed in more detail below.

When you've gathered this information about the filter, you'll enter it as input to the **lpfilter** command. You may want to first record this information in your own file to make it easier to edit the information as you enter it. You can then give the file as input instead. However you enter it, you should present the information in the following way:

```
Input types: input-type-list
Output types: output-type-list
Printer types: printer-type-list
Printers: printer-list
Filter type: fast or slow
Command: simple-command
Options: template-list
```

The information can appear in any order. Not all the information has to be given. Missing information will be assigned the following defaults:

Item	Default
Input types	any
Output types	any
Printer types	any
Printers	any
Filter type	slow
Command	(no default)
Options	(none)

As you can see, the defaults would define a very flexible filter, so you probably have to supply at least the input and output type(s). When you enter a list, separate the items in the list with blanks or commas.

Templates

All the information has been explained except the *templates-list*. Here's how the modes and printing aspects are registered.

The *templates-list* is a list of templates separated by commas and has the following form:

keyword pattern = replacement

keyword must be one of those listed in the following table; it labels the template as registering a particular characteristic of the printing. *pattern* is either a value of the characteristic or an asterisk (*) that stands as a place-holder for any value.

Characteristic	*keyword*	Possible *patterns*
Content type (input)	`INPUT`	*content-type*
Content type (output)	`OUTPUT`	*content-type*
Printer type	`TERM`	*printer-type*
Character pitch	`CPI`	*integer*

Line pitch	`LPI`	*integer*
Page length	`LENGTH`	*integer*
Page width	`WIDTH`	*integer*
Pages to print	`PAGES`	*page-list*
Character set	`CHARSET`	*character-set*
Form name	`FORM`	*form-name*
Number of copies	`COPIES`	*integer*
Modes	`MODES`	*mode*

The source of the values for these templates are as follows:

- The values of the `INPUT` and `OUTPUT` templates come from the file type that needs to be converted by the filter and the output type that has to be produced, respectively. They'll each be a type registered with the filter.
- The value for the `TERM` template is the printer type.
- The values for the `CPI`, `LPI`, `LENGTH`, and `WIDTH` templates come from the user's request, the form being used, or the defaults for the printer.
- The value for the `PAGES` template is a list of pages that should be printed. Typically it is a list of page ranges, either a pair of numbers or a single number, each range separated by a comma (e.g., 1-5,6,8,10 for pages 1 through 5, 6, 8, and 10). However, whatever value was given in the **-P** option to a print request is passed unchanged.
- The value for the `CHARSET` template is the name of the character set to be used.
- The value for the `FORM` template is the name of the form being printed on, if any.
- The value of the `COPIES` template is the number of copies of the file that should be made. If the filter uses this template, the LP print service will reduce the number of copies of the filtered file it will print to 1 since this "single copy" will really be the multiple copies produced by the filter.
- The value of the `MODES` template comes from the **-y** option of the **lp** command, the command a person uses to submit a print request. Since a user can give several **-y** options, there may be several values for the

MODES template. The values will be applied in the left-to-right order given by the user.

The replacement shows how the value of a template should be given to the filter program. It is typically a literal option, sometimes with the placeholder * included to show where the value goes. A few examples should show how this works.

Example 1

The filter program is called */usr/bin/npf*. It takes two input types, **nroff37** and **X**, produces an output type called **TX**, and will work with any printer of type **TX**. The program accepts three options:

-Xb	only for the input type **X**
-l *integer*	for the length of the output page
-w *integer*	for the width of the output page

The filter definition would look like this:

```
Input types: X,nroff37
Output types: TX
Printer types: TX
Command: /usr/bin/npf
Options: INPUT X = -Xb, LENGTH * = -l*,
WIDTH * = -w*
```

If a user submits a file of type `nroff37` and asks that it be printed by a printer named `lp1` which is of type `TX`, and requests a page length of `72`,

```
lp -T nroff37 -d lp1 -o length=72
```

then this filter will be called upon by the LP print service to convert the file. The filter will be invoked as

```
/usr/bin/npf -172
```

Example 2

Another user submits a file of type `X` that is to be printed on the same printer, with default length and width. The filter will be invoked as

```
/usr/bin/npf -Xb
```

Example 3

The filter program is called */usr/bin/x9700*. It takes one input type, `troff`, produces an output type called `9700`, and will work with any printer of type `9700`. The program has one fixed option, **-ib**, and accepts three other options:

-l *integer* for the length of the output page

-s *name* for the character set

-o portrait or

-o landscape for portrait or landscape orientation of the paper

You've decided that your users need give just the abbreviations **port** and **land** when they ask for the paper orientation. Since these are not options intrinsic to the LP print service, users will specify them using the **-y** option to the **lp** command.

The filter definition would look like this:

```
Input types: troff
Output types: 9700
Printer types: 9700
Command: /usr/bin/x9700 -ib
Options: LENGTH * = -l *, CHARSET * = -s *,
         MODES port = -o portrait, MODES land
         = -o landscape
```

(The last line is split into three lines for readability in this document. It would be entered as a single line.)

If a user submits a file of type `troff` for printing on a printer of type `9700` and requests landscape orientation using the `gothic` character set,

```
lp -T troff -S gothic -y land
```

then this filter will be invoked by the LP print service to convert the file as follows:

```
/usr/bin/x9700 -ib -s gothic -o landscape
```

NOTE

If a pattern or replacement must include a comma or equals sign (=), escape its special meaning by preceding it with a backslash. A backslash in front of these two characters will be removed when the pattern or replacement is used. (All other backslashes are left alone.)

Command to Enter

Once you have a filter definition complete, enter one of the following commands to add the filter to the system:

/usr/lib/lpfilter -f *filter-name* **-F** *file-name*
/usr/lib/lpfilter -f *filter-name* **-**

The first command gets the filter definition from a file, and the second command gets the filter definition from you through the standard input. *filter-name* can be anything you choose, as long as it contains 14 or less letters, digits, and underscores.

If you need to change a filter, just re-enter one of the same commands. You need only give the changed information; information you leave out will stay the same.

Removing a Filter

The LP print service has no fixed limit to the number of filters you can define. However, it is a good idea to remove filters no longer applicable to avoid extra processing by the LP print service which must examine all filters to find one that works in a given situation.

Enter the following command to remove a filter:

/usr/lib/lpfilter -f *filter-name* **-x**

Examining a Filter

You can examine a filter definition once you've added it to the LP print service. The **lpfilter** command will display the definition of the filter in a form suitable as input again so that you can save the output in a file for future reference.

Enter one of the following commands to examine a defined filter:

/usr/lib/lpfilter -f *filter-name* **-l**
/usr/lib/lpfilter -f *filter-name* **-l >***file-name*

The first command presents the definition of the filter on your screen; the second command captures this definition in a file, which can later be used to redefine the filter if you inadvertently remove the filter from the LP print service.

A Word of Caution

Adding, changing, or deleting filters can cause print requests still queued to be canceled. This is because the LP print service evaluates each print request still queued to see which are affected by the filter change. Requests that are no longer printable, because a filter has been removed or changed, are canceled (with notifications sent to the people who submitted them). There can also be a delay in the response to new or changed print requests when filters are changed, due to the many characteristics that must be evaluated for each print request still queued. This delay can become noticeable if there are a large number of requests needing filtering.

Because of this possible impact, you may want to make changes to filters during periods when the LP print service is not being used much.

Directories and Files

This section lists the directories and files used by the LP print service. You can use this list to see if any files are missing or if the ownership or access permissions have changed. Normal operation of the LP print service should not cause any problems. However, if you do notice any discrepancies, it could be a cause for a security breach on your system.

At the end of this section is a description of the script used to clean out the request log periodically. You may want to change this script to have the file cleaned out on a different period or to condense the information into a report. See "Cleaning Out the Request Log" below.

All directories and files are found under the parent directory */usr/spool/lp*. This directory should have the following access permissions and ownership:

Permissions	Owner	Group	Directory or File
drwxrwxr-x	lp	bin	/usr/spool/lp

You can check this by entering the following command.

```
ls -ld /usr/spool/lp
```

Under this directory you should see only the directories and files shown in the following table. Those marked with an asterisk (*) may be missing, depending on the state of the print service or its configuration.

You can generate a similar table for comparison by entering this command:

```
ls -lR /usr/spool/lp
```

Permissions	Owner	Group	Directory or File
-rw-rw-r--	lp	bin	* SCHEDLOCK
drwxrwxr-x	lp	bin	admins
drwxrwxr-x	lp	bin	bin
-rw-r--r--	lp†	bin†	* default
drwxrwxr-x	lp	bin	fifos
drwxrwxr-x	lp	bin	logs

Permissions	Owner	Group	Directory or File
drwxrwxr-x	lp	bin	model
drwxrwxr-x	lp	bin	requests
drwxrwxr-x	lp	bin	system
drwxrwxr-x	lp	bin	temp
-rw-r--r--	lp†	bin†	* users
/usr/spool/lp/admins:			
drwxrwxr-x	lp	bin	lp
/usr/spool/lp/admins/lp:			
drwxrwxr-x	lp	bin	classes
-rw-rw-r--	lp†	bin†	* filter.table
-rw-rw-r--	lp	bin	* filter.table.i
drwxrwxr-x	lp	bin	forms
drwxrwxr-x	lp	bin	interfaces
drwxrwxr-x	lp	bin	logs
drwxrwxr-x	lp	bin	printers
drwxrwxr-x	lp	bin	pwheels
/usr/spool/lp/admins/lp/classes:			
-rw-rw-r--	lp†	bin†	* *class1*
-rw-rw-r--	lp†	bin†	* *class2*
.			
.			
.			
-rw-rw-r--	lp†	bin†	* *classN*
/usr/spool/lp/admins/lp/forms:			
drwxrwxr-x	lp†	bin†	* *form1*
drwxrwxr-x	lp†	bin†	* *form2*
.			
.			
.			
drwxrwxr-x	lp†	bin†	* *formN*
/usr/spool/lp/admins/lp/forms/*formK*:			
-rwxrwx---	lp†	bin†	* alert.sh
-rw-rw----	lp†	bin†	* alert.vars
-rw-rw----	lp†	bin†	* align_ptrn

Permissions	Owner	Group	Directory or File
-rw-rw-r--	lpt	bint	* allow
-rw-rw-r--	lpt	bint	* comment
-rw-rw-r--	lpt	bint	* deny
-rw-rw-r--	lpt	bint	* describe
/usr/spool/lp/admins/lp/interfaces:			
-rwxrwxr-x	lpt	bint	* *printer1*
-rwxrwxr-x	lpt	bint	* *printer2*
.			
.			
.			
-rwxrwxr-x	lpt	bint	* *printerN*
/usr/spool/lp/admins/lp/printers:			
drwxrwxr-x	lpt	bint	* *printer1*
drwxrwxr-x	lpt	bint	* *printer2*
.			
.			
.			
drwxrwxr-x	lpt	bint	* *printerN*
/usr/spool/lp/admins/lp/printers/*printerK*:			
-rwxrwx---	lpt	bint	* alert.sh
-rw-rw----	lpt	bint	* alert.vars
-rw-rw-r--	lpt	bint	* comment
-rw-rw-r--	lpt	bint	* configuration
-rw-rw-r--	lpt	bint	* forms.allow
-rw-rw-r--	lpt	bint	* forms.deny
-rw-rw-r--	lpt	bint	* users.allow
-rw-rw-r--	lpt	bint	* users.deny
/usr/spool/lp/admins/lp/pwheels:			
drwxrwxr-x	lpt	bint	* *printwheel1*
drwxrwxr-x	lpt	bint	* *printwheel2*
.			
.			
.			
drwxrwxr-x	lpt	bint	* *printwheelN*

Permissions	Owner	Group	Directory or File
/usr/spool/lp/admins/lp/pwheels/*printwheelK*			
-rwxrwx---	lp†	bin†	* alert.sh
-rw-rw----	lp†	bin†	* alert.vars
/usr/spool/lp/bin:			
-r--r--r--	lp	bin	alert.proto
-rwxrwxr-x	lp	bin	drain.output
-rwxrwxr-x	lp	bin	lp.cat
-rwxrwxr-x	lp	bin	lp.page
-rwxrwxr-x	lp	bin	lp.set
-rwxrwxr-x	lp	bin	lp.tell
-rwxrwxr-x	lp	bin	lpsched.jr
-rwxrwxr-x	lp	bin	slow.filter
/usr/spool/lp/fifos:			
p-w--w--w-	root	other	* FIFO
drwxrwx--x	lp	sys	private
drwxrwx-wx	lp	sys	public
/usr/spool/lp/fifos/private:			
pr--------	*user*	*group*	* *machPID*
.			
.			
.			
/usr/spool/lp/fifos/public:			
pr--------	*user*	*group*	* *machPID*
.			
.			
.			
/usr/spool/lp/logs:			
-rw-rw----	lp	bin	* lpsched
-rw-rw----	lp	bin	* requests
-rw-rw----	lp	bin	* *requests1*
-rw-rw----	lp	bin	* *requests2*
.			
.			
.			

Permissions	Owner	Group	Directory or File
-rw-rw----	lp	bin	* *requestsN*
/usr/spool/lp/model:			
-rwxrwxr-x	bin	bin	1640
-rwxrwxr-x	bin	bin	5310
-rwxrwxr-x	bin	bin	dqp10
-rwxrwxr-x	bin	bin	dumb
-rwxrwxr-x	bin	bin	f450
-rwxrwxr-x	bin	bin	hp
-rwxrwxr-x	bin	bin	lqp40
-rwxrwxr-x	bin	bin	pprx
-rwxrwxr-x	bin	bin	prx
-rwxrwxr-x	bin	bin	standard
/usr/spool/lp/requests:			
-rw-rw----	lp	bin	* *id1*-0
-rw-rw----	lp	bin	* *id2*-0
.			
.			
.			
-rw-rw----	lp	bin	* *idN*-0
/usr/spool/lp/system:			
-rw-rw-r--	lp	bin	* cstatus
-rw-rw-r--	lp	bin	* pstatus
/usr/spool/lp/temp:			
-rw-------	lp	bin	* *idN*-0
-rw-------	lp	bin	* *idN*-1
-rw-------	lp	bin	* *idN*-2
.			
.			
.			
-rw-------	lp	bin	* *idN-M*
-rw-------	lp	bin	* F*idN*-1
-rw-------	lp	bin	* F*idN*-2
.			
.			
.			

Permissions	Owner	Group	Directory or File
-rw-------	lp	bin	* F*idN-M*
-rw-------	lp	bin	* *idN*
-rw-------	lp	bin	* A-*K*
-rw-------	lp	bin	* F-*K*
-rw-------	lp	bin	* P-*K*

The italicized names, *printerN*, *formN*, *classN*, *printwheelN*, and *idN* are placeholders for a single printer, form, class, print wheel, and request ID, respectively. (*idN* is just the numeric part of the request ID.) There will be one set of these directories and files for each active printer, form, class, print wheel, and request on your system. The italicized letter *K* is a placeholder for an internal number; the A-*K*, F-*K*, and P-*K*, files are used to store alert messages.

The ownership and permissions of the *idN-M* request files under the */usr/spool/lp/temp* directory will change during the life of a print request, alternating between the user who submitted the request and the **lp** ID.

The directories under the *usr/spool/lp/fifos* directory contain named pipes used to communicate between the LP print service and commands such as **lpadmin**, **lpstat**, **lp**, and so on. These two directories must have the permission flags and ownership shown if the communication with the LP print service is to work. Every entry below these directories is given a unique name formed by combining the name of the system (the node name) and the process ID of the command. The uniqueness of the entry names prevents two or more people from accidentally sharing the same communications path.

Cleaning Out the Request Log

The directories */usr/spool/lp/temp* and *usr/spool/lp/requests* contain files that describe each request that has been submitted to the LP print service. Each request has two files, one in each directory, that contain information about the request. The information is split to put more sensitive information in the */usr/spool/lp/requests* directory where it can be kept secure. The

request file in the */usr/spool/lp/temp* is safe from all except the user who submitted the request, while the file in */usr/spool/lp/requests* is safe from even the submitting user.

These files remain in their directories only as long as the request is on the queue. Once the request is finished, the information in the files is combined and appended to the file */usr/spool/lp/logs/requests*. This file is not removed by the LP print service but can be cleaned out periodically, using, for instance, the **cron** facility. See the description of the **crontab** command in the *User's/System Administrator's Reference Manual*.

The default **crontab** entry suggested with the LP print service system is shown below:

```
13 3 * * * cd /usr/spool/lp/logs; if [ -f
requests ]; then /bin/mv requests xyzzy; /bin/cp
xyzzy requests; >xyzzy; /usr/lbin/agefile -c2
requests; /bin/mv xyzzy requests; fi
```

(This is one line in **crontab** but is split into several lines here for readability.) What this entry does, briefly, is "age" the file, changing the name to *requests-1* and moving the previous day's copy to *requests*-2. The number 2 in the **-c** option to the **agefile** program will keep the log files from the previous two days, discarding older log files. By changing this number, you can change the amount of information saved. On the other hand, if you want the information saved more often or want to clean out the file more often than once a day, change the time when the **crontab** entry is run by changing the first two numbers. The current values, 13 and 3, cause the cleanup to occur at 3:13 AM each day.

The default **crontab** entry supplied is sufficient to keep the old print request records from accumulating in the spooling filesystem. You may want to condense information in the request log to produce a report on the use of the LP print service or to aid in generating accounting information. You can produce a different script that examines the file and extracts information just before the cleanup procedure.

The request log has a simple structure that makes it easy to extract data using common UNIX shell commands. The requests are listed in the order in which they were printed and are separated by lines that give the request ID. Each line below the separator line is marked with a single letter that identifies the kind of information contained in the line. Each letter is separated from the data by a single space. See the following table for details.

Letter	Content of line
=	This is the separator line, containing the request ID, the user and group IDs of the user, the total number of bytes in the original (unfiltered) files, and the time when the request was queued. These items are separated by commas and are in the order just named. The user ID, group ID, and sizes are preceded by the word `uid`, `gid`, or `size`, respectively.
C	The number of copies printed.
D	The printer or class destination or the word `any`.
F	The name of the file printed. This line is repeated for each file printed, and files are printed in the order given.
f	The name of the form used.
H	The type of special handling used, spelled out (`resume`, `hold`, `immediate`). The only useful value found in this line will be `immediate`.
N	The type of alert used when the print request successfully completed. The type is the letter `M` if the user was notified by mail, or `W` if the user was notified by a message to his or her terminal.
O	The **-o** options.
P	The priority of the print request.
p	The list of pages printed.
r	This single letter line is present if the user asked for "raw" processing of the files (the **-r** option of the **lp** command.)
S	The character set or print wheel used.

Letter	Content of line
s	The outcome of the request as a combination of individual bits expressed in hexadecimal form. While several bits are used internally by the Spooler, the most important bits are listed below: 0x0004 Slow filtering finished successfully. 0x0010 Printing finished successfully. 0x0040 The request was canceled. 0x0100 The request failed filtering or printing.
T	The title placed on the banner page.
t	The type of content found in the file(s).
U	The name of the user who submitted the print request.
x	The slow filter used for the request.
Y	The list of special modes to give to the filters used to print the request.
y	The fast filter used for the request.
z	The printer used for the request. This will differ from the destination (the D line) if the request was queued for any printer or a class of printers or if the request was moved to another destination by the LP print service administrator.

Customizing the Print Service

Although the LP print service tries to be flexible enough to handle most printers and printing needs, it can't be complete. You may buy a printer that doesn't quite fit into the way the LP print service handles printers or may have a printing need that the standard features of the LP print service won't accommodate.

You can customize the LP print service in a few ways. This section tells you how you can

- adjust the printer port characteristics
- adjust the Terminfo database
- write an interface program
- write a filter

The diagram in Figure 7-4 gives an overview of the processing of a print request.

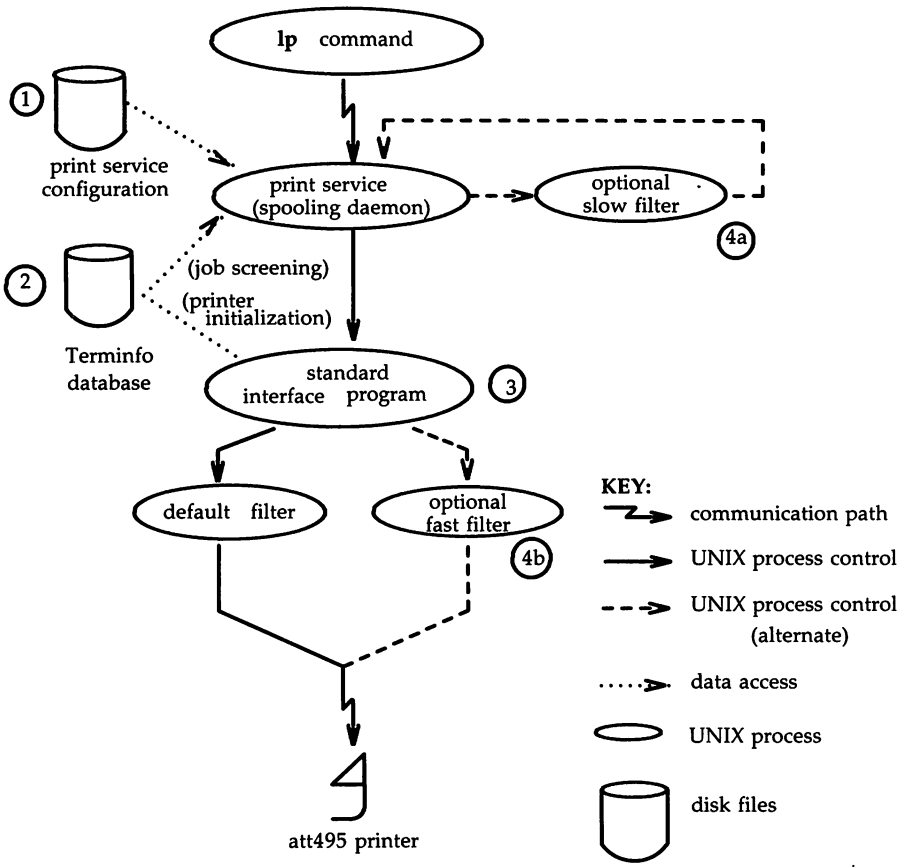


Figure 7-4: How LP processes print request lp –d att495 file

Each print request is sent to a *spooling daemon* that keeps track of all the requests. The daemon is created when you start the LP print service. This UNIX System process is also responsible for keeping track of the status of the printers and slow filters; when a printer finishes printing a user's file, the daemon will start it printing another request, if one is queued.

You are able to customize the print service by adjusting or replacing some of the pieces shown in Figure 7-4. (The numbers are keyed to the diagram.)

1. For most printers, you need only change the printer configuration stored on disk. The earlier sections of this chapter have explained how to do this. Some of the more printer-dependent configuration data are the printer port characteristics: baud rate, parity, and so on.

2. For printers that are not represented in the Terminfo database, you can add a new entry that describes the capabilities of the printer. This database is used in two parallel capacities: screening print requests to ensure that those accepted can be handled by the desired printer and setting the printer in a state where it is ready to print the request.

   For instance, if the Terminfo database does not show a printer capable of setting a page length requested by a user, the spooling daemon will reject the request. On the other hand, if it does show it capable, then the same information will be used by the interface program to initialize the printer.

3. For particularly difficult printers or if you want to add features not provided by the delivered LP print service, you can change the standard interface program. This program is responsible for managing the printer: it prints the banner page, initializes the printer, and invokes a filter to send copies of the user's files to the printer.

4a. and 4b.
   To provide a link between the applications used on your system and the printers, you can add slow and fast filters. Each type of filter can convert a file into another form, mapping one set of escape sequences into another, for instance, and can provide special setup by interpreting print modes requested by a user. Slow filters are run separately by the daemon to avoid tying up a printer. Fast filters are run so their output goes directly to the printer; thus, they can exert control over the printer.

Adjusting the Printer Port Characteristics

You should make sure that the printer port characteristics set by the LP print service match the printer communication settings. The standard printer port settings were designed to work with typical UNIX files and many printers, but they won't work with all files and printers. This isn't really a customizing step, since a standard feature of the LP print service is to allow you to specify the port settings for each printer. However, it's an important step in getting your printer to work with the LP print service, so it's described in more detail here.

When you add a new printer, read the documentation that comes with it so that you understand what it expects from the host (the LP print service). Then read the manual page for the *stty*(1) command in the *User's/System Administrator's Reference Manual*. It summarizes the various characteristics that can be set on a terminal or printer port.

Only some of the characteristics listed in the *stty*(1) manual page are important for printers. The ones likely to be of interest to you are listed below (but you should still consult the *stty*(1) manual page for others.)

stty Option	Meaning
evenp	Sends even parity in the 8th bit
oddp	Sends odd parity in the 8th bit
-parity	Doesn't generate parity, sends all 8 bits unchanged
110 - 38400	Sets the communications speed to this baud rate
ixon	Enables XON/XOFF (also known as START/STOP or DC1/DC3) flow control
-ixon	Turns off XON/XOFF flow control
-opost	Doesn't do any "output post-processing"

opost	Does "output post-processing" according to the settings listed below
onlcr	Sends a carriage return before every linefeed
-onlcr	Doesn't send a carriage return before every linefeed
ocrnl	Changes carriage returns into linefeeds
-ocrnl	Doesn't change carriage returns into linefeeds
-tabs	Changes tabs into an equivalent number of spaces
tabs	Doesn't change tabs into spaces

When you have a set of printer port characteristics you think should apply, adjust the printer configuration as described in the "Printer Port Characteristics" section under "Printer Management" in this chapter. You may find that the default settings are sufficient for your printer.

Adjusting the Terminfo Database

The LP print service relies on a standard interface and the Terminfo database to initialize each printer and set up a selected page size, character pitch, line pitch, and character set. Thus, it is usually sufficient to have the correct entry in the Terminfo database to add a new printer to the LP print service. Several entries for AT&T printers and other popular printers are delivered in Terminfo database entries with the LP print service package.

Each printer is identified in the Terminfo database with a short name; this kind of name is identical to the kind of name used to set the **TERM** shell variable. For instance, the AT&T model 455 printer is identified by the name **455**. The "Acceptable Terminal Names" section of "Appendix F" in the *User's Guide* gives a description of how to determine a correct **TERM** variable for a user terminal and can be used as a guide for picking a known name for your printer.

If you cannot find a Terminfo entry for your printer, you should add one. If you don't, you may still be able to use the printer with the LP print service, but you won't be able to get automatic selection of page size, pitch, and character sets, and you may have trouble keeping the printer set in the correct

modes for each print request. Another option to follow instead of updating the Terminfo entry is to customize the interface program used with the printer. See the next section for details on how to do this.

There are hundreds of items that can be defined for each terminal or printer in the Terminfo database. However, the LP print service uses less than fifty of these, and most printers need even less than that. The following table lists the items that need to be defined (as appropriate for the printer) to add a new printer to the LP print service.

Terminfo item	Meaning
Booleans:	
`daisy`	Printer needs operator to change character set
Numbers:	
`bufsz`	Number of bytes buffered before printing
* `cols`	Number of columns in a line
* `it`	Tabs initially every this many spaces
* `lines`	Number of lines on a page
`orc`	Horizontal resolution in units per character
`orhi`	Horizontal resolution in units per inch
`orl`	Vertical resolution in units per line
`orvi`	Vertical resolution in units per inch
`cps`	Average print rate in characters per second
Strings:	
* `cr`	Carriage return
`cpi`	Change number of characters per inch
`lpi`	Change number of lines per inch
`chr`	Change horizontal resolution
`cvr`	Change vertical resolution
`csnm`	List of character set names
`mgc`	Clear all margins (top, bottom and sides)
* `hpa`	Horizontal position absolute

	Terminfo item	Meaning
*	`cud1`	Down one line
*	`cuf1`	Carriage right
	`swidm`	Enable double wide printing
	`rwidm`	Disable double wide printing
*	`ff`	Page eject
*	`is1`	Printer initialization string
*	`is2`	Printer initialization string
*	`is3`	Printer initialization string
*	`if`	Name of initialization file
*	`iprog`	Path name of initializing program
*	`cud`	Move carriage down # lines
*	`cuf`	Move carriage right # columns
*	`rep`	Repeat a character # times
*	`vpa`	Vertical position absolute
	`scs`	Select character set
	`smgb`	Set bottom margin at current line
	`smgbp`	Set bottom margin
*	`smgl`	Set left margin at current column
	`smglp`	Set left margin
*	`smgr`	Set right margin at current column
	`smgrp`	Set right margin
	`smgt`	Set top margin at current line
	`smgtp`	Set top margin
	`scsd`	Start definition of a character set
*	`ht`	Tab to next 8 space tab stop

The items marked with a leading asterisk (*) are available on all releases of UNIX System V. The rest can be added only if you are using UNIX System V Release 3.2 or later.

NOTE

If you are running the LP print service on a UNIX System V Release 3.1, only the Terminfo items marked in the table are available. They are sufficient for initializing the printer but not for setting page sizes and pitches or selecting character sets.

Consult the manual page for the *terminfo*(4) file structure in the *Programmer's Reference Manual* for details on how to construct a Terminfo database entry for a new printer.

Once you've made the new entry, you need to compile it into the database using the **tic** program. Just enter the following command:

tic *filename*

filename is the name of the file containing the Terminfo entry you have crafted for the new printer. (This program is available in the Terminal Information Utilities.)

NOTE

The LP print service gains much efficiency by "caching" information from the Terminfo database. If you add or delete Terminfo entries or change the values that govern pitch settings, page width and length, or character sets, you should stop and restart the LP print service so it can read the new information.

How to Write an Interface Program

NOTE

If you have an interface program that you have used with the LP Spooler Utilities before UNIX System V Release 3.2, it should still work with the LP print service. Note, though, that several **-o** options have been "standardized" and will be passed to every interface program. These may interfere with similarly named options your interface program uses.

If you have a printer that is not supported by simply adding an entry to the Terminfo database or if you have printing needs that are not supported by the

standard interface program you can furnish your own interface program. It is a good idea to start with the standard interface program and change it to fit, rather than starting from scratch. You can find a copy of it under the name

```
/usr/spool/lp/model/standard
```

What Does an Interface Program Do?

Any interface program performs the following tasks:

- Initializes the printer port, if needed. The generic interface program uses the **stty** command to do this.
- Initializes the physical printer. The generic interface program uses the Terminfo and the **TERM** shell variable to get the control sequences to do this.
- Prints a banner page, if needed.
- Prints the correct number of copies of the request content.

An interface program is not responsible for opening the printer port. This is done by the LP print service, which calls a "dial-up" printer if that's how the printer is connected. The printer port connection is given to the interface program as standard output, and the printer is set to be the "controlling terminal" for the interface program so that a "hang-up" of the port will cause a `SIGHUP` signal to be sent to the interface program.

A customized interface program must not terminate the connection to the printer or in any fashion "uninitialize" the printer. This allows the LP print service to use the interface program only for preparing the printer and printer port, while the printing of content is done elsewhere, by the LP print service, for example, for preprinted form alignment patterns.

How is an Interface Program Used?

When the LP print service routes an output request to a printer, the interface program for the printer is invoked as follows:

/usr/spool/lp/admins/lp/interface/*P id user title copies options file1 file2 ...*

(The last line is split into two lines for readability in this document.)

Arguments for the interface program are

P	printer name
id	request id returned by lp
user	logname of user who made the request
title	optional title specified by the user
copies	number of copies requested by user
options	list of options separated by blanks, specified by user or set by the LP print service
file	full path name of a file to be printed

When the interface program is invoked, its standard input comes from */dev/null*, its standard output is directed to the printer port, and its standard error output is directed to a file that will be given to the user who submitted the print request.

The standard interface recognizes the following values in the list in *options*:

nobanner This option is used to skip the printing of a banner page; without it, a banner page is printed.

nofilebreak This option is used to skip page breaks between separate data files; without it, a page break is made between each file in the content of a print request.

cpi=*decimal-number*$_1$
lpi=*decimal-number*$_2$
These options say to print with *decimal-number*$_1$ columns per inch and *decimal-number*$_2$ lines per inch, respectively. The standard interface program extracts from the Terminfo database the control sequences needed to initialize the printer to handle the character and line pitches.

The words *pica*, *elite*, and *compressed* are acceptable replacements for the *decimal-number*$_1$ and are synonyms for *10* columns per inch, *12* columns per inch, and as many columns per inch as possible.

length=*decimal-number*$_1$
width=*decimal-number*$_2$
These options specify the length and width, respectively, of the pages to be printed. The standard interface program extracts from the Terminfo database the control sequences needed to initialize the printer to handle the page length and page width.

stty='*stty-option-list*'
The *stty-option-list* is applied after a default *stty-option-list* as arguments to the **stty** command. The default list is used to establish a default port configuration; the additional list given to the interface program is used to change the configuration as needed.

The above options are either specified by the user when issuing a print request or by the LP print service from defaults given by the administrator for the printer (**cpi, lpi, length, width, stty**) or for the preprinted form used in the request (**cpi, lpi, length, width**).

Additional printer configuration information is passed to the interface program in shell variables:

TERM=*printer-type*
This shell variable specifies the type of printer. The value is used as a key for getting printer capability information from the extended Terminfo database.

FILTER='*pipeline*'
This shell variable specifies the filter to use to send the request content to the printer; the filter is given control of the printer.

CHARSET=*character-set*
This shell variable specifies the character set to be used when printing the content of a print request. The standard interface program extracts from the Terminfo database the control sequences needed to select the character set.

A customized interface program should either ignore these options and shell variables or should recognize them and treat them in a consistent manner.

Customizing the Interface Program

You want to make sure that the custom interface program sets the proper **stty** modes (terminal characteristics such as baud rate or output options). The standard interface program does this, and you can follow suit. Look for the section that begins with the shell comment

```
## Initialize the printer port
```

Follow the code used in the standard interface program. It sets both the default modes and the adjusted modes given by the LP print service or the user with a line like the following:

stty *mode options* **0<&1**

This command line takes the standard input for the **stty** command from the printer port. An example of an **stty** command line that sets the baud rate at 1200 and sets some of the option modes is shown below:

```
stty -parenb -parodd 1200 cs8 cread clocal ixon 0<&1
```

One printer port characteristic not set by the standard interface program is hardware flow control. The way that this is set will vary depending on your computer hardware. The code for the standard interface program suggests where this and other printer port characteristics can be set. Look for the section that begins with the shell comment

```
# Here you may want to add other port
initialization code.
```

The above line is split into two lines for readability.

Because different printers have different numbers of columns, make sure the header and trailer for your interface program correspond to your printer. The standard interface program prints a banner that fits on an 80-column page (except for the user's title which may be longer). Look for the section in the code for the standard interface program that begins with the shell comment

```
## Print the banner page
```

The custom interface program should print all user-related error messages on the standard output or on the standard error. The messages sent to the standard error will be mailed to the user; the messages printed on the standard output will end up on the printed page where they can be read by the user when he or she picks up the output.

When printing is complete, your interface program should exit with a code that tells the status of the print job. Exit codes are interpreted by the LP print service as follows:

Code	Meaning to the LP print service
0	The print request has completed successfully. If a printer fault has occurred, it has been cleared.
1 to 127	A problem was encountered in printing this particular request (for example, too many non-printable characters or the request exceeds the printer capabilities). This problem will not affect future print requests. The LP print service notifies the person who submitted the request that there was an error in printing it. If a printer fault has occurred, it has been cleared.
128	Reserved for internal use by the LP print service. Interface programs must not exit with this code.
129	A printer fault was encountered in printing the request. This problem will affect future print requests. If the fault recovery for the printer directs the LP print service to wait for the administrator to fix the problem, it will disable the printer. If the fault recovery is to continue printing, the LP print service will not disable the printer but will try printing again in a few minutes.
greater than 129	These codes are reserved for internal use by the LP print service. Interface programs must not exit with codes in this range.

As the table shows, one way of alerting the administrator to a printer fault is to exit with a code of 129. Unfortunately, if the interface program exits, the LP print service has no choice but to reprint the request from the beginning when the fault has been cleared. Another way of getting an alert to the

administrator but without requiring reprinting the entire request, is to have the interface program send a fault message to the LP print service but wait for the fault to clear. When the fault clears, the interface program can resume printing the user's file. When done printing, it can give a zero exit code just as if the fault never occurred. An added advantage is that the interface program can detect when the fault is cleared automatically so that the administrator doesn't have to enable the printer.

Fault messages can be sent to the LP print service using the **lp.tell** program. This is referenced using the **$LPTELL** shell variable in the standard interface code. The program takes its standard input and sends it to the LP print service where it is put into the message that alerts the administrator to the printer fault. If its standard input is empty, **lp.tell** does not initiate an alert. Examine the standard interface code immediately after these comments for an example of how the **lp.tell** (**$LPTELL**) program is used:

```
# Here's where we set up the $LPTELL program
to capture
# fault messages.

# Here's where we print the file.
```

With the special exit code 129 or the **lp.tell** program, there is no longer the need for the interface program to disable the printer itself. Your interface program can disable the printer directly, but doing so will override the fault alerting mechanism. Alerts are sent only if the LP print service detects the printer has faulted and the special exit code and the **lp.tell** program are its main detection tools.

If the LP print service has to interrupt the printing of a file at any time, it will "kill" the interface program with a signal 15 (see *kill*(1) in the *User's/System Administrator's Reference Manual* and *signal*(2) in the *Programmer's Reference Manual*). If the interface program dies from receipt of any other signal, the LP print service assumes that future print requests won't be affected and will continue to use the printer. The LP print service will notify the person who submitted the request that it did not finish successfully.

The signals `SIGHUP`, `SIGINT`, `SIGQUIT`, and `SIGPIPE` (trap numbers 1, 2, 3, and 13) start out being ignored when the interface is invoked. The standard interface changes this to trap these signals at appropriate times. The standard interface will consider receipt of these signals as meaning the printer has a problem and will issue a fault.

How to Write a Filter

A filter is used by the LP print service each time it has to print a type of file that isn't acceptable by a printer. While a filter can be as simple or as complex as needed, there are only a few external requirements:

- The filter should get the content of a user's file from its standard input and send the converted file to the standard output.
- A "slow" filter can send messages about errors in the file to standard error. A "fast" filter should not, as described below. Error messages from a "slow" filter will be collected and sent to the user who submitted the file for printing.
- If a "slow" filter dies because of receiving a signal, the print request is finished and the user who submitted the request is notified. Likewise, if a "slow" filter exits with a non-zero exit code, the print request is finished and the user is notified. The exit codes from "fast" filters are treated differently, as described below.
- A filter should not depend on other files that would not normally be accessible to a regular user; if the filter would fail if the user ran it directly, it will fail when the LP print service runs it.

The "Filter Management" section describes how to add a filter to the LP print service.

There are a few more requirements if the filter is also to detect printer faults:

- If it can, it should wait for a fault to clear before exiting. Additionally, it should continue printing at the top of the page where printing stopped after the fault clears. If this is not the administrator's intentions, the LP print service will stop the filter before alerting the administrator.
- It should send printer fault messages to its standard error as soon as the fault is recognized. It does not have to exit but can wait as described above.
- It should not send messages about errors in the file to standard error. Any messages on the standard error will eventually generate a pointer fault. These should be included in the standard output stream, where they can be read by the user.

- It should exit with a zero exit code if the user's file is finished (even if errors in the file prevented it from printing correctly).
- It should exit with a non-zero exit code only if a printer fault kept it from finishing a file.
- When added to the filter table, it must be added as a "fast" filter. See the "Defining a Filter" section under "Filter Management" in this chapter.

Chapter 8: Basic Networking Administration

Introduction to Basic Networking Administration

This chapter describes the administration of the Basic Networking Utilities (BNU). It allows you to communicate with other UNIX System computers using either dial-up or hard-wired communication lines. The BNU comes with the UNIX System Foundation Software Set and is specifically part of the Base System Package. The BNU files are located on a separate floppy along with all modem scripts and a file containing a list of supported modems, such as AT&T 2212C, AT&T 2224B, AT&T 4000 Model 1A01, AT&T 4000 Model 1A02, AT&T 4024, and AT&T 4112.

The installation script for the Base System is normally used to select the type(s) of modems that you will be using. However, if you have not made a selection, refer to "Setting Up an RS-232 Connection" in Chapter 4, "System Administration." The basic instructions for setting up your computer to communicate with other computers is provided there. This will include connecting your computer to a modem so you can send electronic mail. Once a particular modem has been selected, refer to Appendix B, "Adding Basic Networking," for additional details.

Also, you can refer to the *User's Guide*, Chapter 9, "Communication Tutorial," for information on using Basic Networking.

Basic Networking is complex; the documentation included in this chapter will cover only the information most important for you.

Terms You Need to Know

The following list contains some terms used in Basic Networking that you may not be familiar with and a brief description of each item:

local machine	Refers to the machine on the "near" end of a communication link, normally your computer.
remote machine	Refers to a machine on the "far" end of a communication link, normally a machine that your computer talks to.
active machine	A machine that has Basic Networking and the hardware required to establish communication links (i.e., Auto Dial Modem).
passive machine	A machine that has Basic Networking but does not have the hardware required to establish communication links.
network	A group of machines set up to exchange information and resources.
node	A terminating point (machine) on a network.
UUCP	Indicates a group of programs and files that allow systems to send or copy information from one system to another. UUCP means "UNIX System-to-UNIX System copy." In general, it refers to Basic Networking with the exception of the **cu** and **ct** programs. If "uucp" (lowercase) is used in text with bold type (**uucp**), it refers specifically to the **uucp** program or login ID.

Overview of Basic Networking

Basic Networking allows machines using the UNIX Operating System to communicate with one another. In general, Basic Networking allows you to do the following:

- Transfer files and send electronic mail to other UNIX System machines as background processes
- Interactively communicate with UNIX System machines and, in some cases, non-UNIX System machines
- Execute commands (restrictive) on a remote machine without logging in
- Call a remote terminal and allow the user of that terminal to log in on your system

The latter part of this chapter discusses the various ways information is transferred from one machine to another. It also discusses how commands are executed remotely and how your computer can call a remote terminal. But first, you should become familiar with the hardware and software associated with Basic Networking.

What Kind of Hardware Is Needed

Before your computer can communicate with a remote machine, a communication link must be established to the remote machine. There are two types of hardware used to establish a communication link to another machine.

The first is a direct link from a serial port on the computer to a serial port on the other machine. This type of connection is useful when two machines communicate with each other on a regular basis. Even though the RS-232 standard recommends that direct links be limited to 50 feet or less, two machines may be separated by several hundred feet provided that noise on the direct link does not become a problem. If noise becomes a problem or greater distance is needed between the two machines, the transfer rate may need to be decreased or limited distance modems placed at each end of the connection.

The second type of communication link uses the telephone network. In this type of link, the machine that establishes the connection (local machine) must have an Automatic Call Unit (ACU). The ACU dials the specified telephone number upon request from Basic Networking. The called (remote) machine must have a telephone modem capable of answering incoming calls so that other machines can contact it through the telephone network. The computer supports a number of automatic dial modems as ACUs. See Appendix B, "Adding Basic Networking," for details.

The Basic Networking Software

Basic Networking is composed of software programs, daemons (background routines), and a supporting data base. The supporting data base contains support files that store information such as telephone numbers, location of the devices (hardware) used to establish links, and security restrictions. The software programs and a skeleton data base are supplied in Basic Networking.

The Directories and Their Purpose

There are several directories that contain the programs and support files of Basic Networking. Some of these directories are unique to Basic Networking, while others are common to the UNIX Operating System and the computer. The directories used by Basic Networking are as follows:

/usr/bin	This directory is used by the UNIX Operating System and by Basic Networking to store executable programs.
/usr/lib/uucp	This directory is the "HOME" directory for the **uucp** administrative login. It contains the files of the supporting data base and some executable programs.

/usr/spool/locks	This directory contains the lock (LCK) files for the Basic Networking hardware devices. Lock files prevent duplicate conversations and multiple attempts to use the same device.
/usr/spool/uucp	This directory is the "spool directory" for "work" that is to be processed by Basic Networking. It contains a tree-like structure of subdirectories associated with remote machines that your computer wishes to communicate with or has communicated with recently. These subdirectories are also used for administrative purposes such as storing log and status information.
/usr/spool/uucppublic	This directory is the "public" directory for UUCP transfers. The public directory is used to store files that have been sent to your computer. Some remote machines may be restricted to placing files in this directory, while others may have permission to place files elsewhere.

The Software Programs and Their Purpose

There are several types of programs associated with Basic Networking. Some of these programs are used by regular users to transfer data and obtain status information, while others are used for administration purposes or are executed internally. The following paragraphs contain a brief description of the programs and their purpose.

User Programs

cu: Connects your computer to a remote machine and allows you to be logged in on both machines at the same time. This allows you to transfer files or execute commands on either machine without dropping the link.

ct: Connects your computer to a remote terminal and allows you to log in to that terminal. The user of the remote terminal may call into the computer and request that the computer call the remote terminal back. In this case, the computer drops the initial link so that the modem will be available when it is called back.

uucp: Performs all of the preliminary work to allow you to send files to remote machines. It creates "work" files that contain the instructions for transferring the queued file(s). Depending on the options specified, it may make a copy of the file to be transferred in the spool directory. These files are called "data" files. Once the "work" and "data" files have been created, **uucp** calls the **uucico** daemon that attempts to contact the remote machine to deliver the files.

uuto: This program works very similarly to the **uucp** program. In fact, it calls the **uucp** program to create "work" and "data" files. The main difference between **uuto** and **uucp** is the way the transferred files are placed on the remote machine. With **uucp**, you can specify a pathname on the remote machine where you want the files to be placed. With **uuto**, all transferred files are placed in the *uucppublic* directory under */usr/spool/uucppublic/receive*. See the *uuto*(1C) manual page in the *User's/System Administration Reference Manual* for additional information.

uupick: When files are transferred to a machine using **uuto**, **uupick** can be used to retrieve the files placed under */usr/spool/uucppublic/receive*.

uux: This program creates "work" files, "data" files, and "execute" files for executing commands on a remote machine. The "work" file contains the same information as files created by **uucp** and **uuto**. The "execute" files contain the command string to be executed on the remote machine and a list of the "data" files. The "data" files are those files required for the command execution.

uustat: This program displays status information for requested transfers (**uucp**, **uuto**, or **uux**). It also provides you with a means of controlling queued transfers.

Administrative Programs

uulog: This program displays the contents of a specified machine's log file. Individual log files are created for each remote machine that your computer communicates with using the **uucp**, **uuto**, and **uux** programs.

uucleanup: This program has several functions associated with the cleanup of the spool directory. It is normally executed out of a shell script called **uudemon.cleanu** that is started by **cron**. See the *cron*(1M) manual page in the *User's/System Administrator's Reference Manual* for additional information.

Uutry: This program is a shell script used to test call processing capabilities with a moderate amount of debugging. It invokes the **uucico** daemon to establish the communication link between your computer and the specified machine.

uucheck: This program checks for the presence of Basic Networking directories, programs, and support files. It is also capable of checking certain parts of the *Permissions* file.

Internal Programs

uugetty: This program is very similar to the **getty** program except it permits a line (port) to be used in both directions. The **uugetty** program allows users to log in on your computer; if the line is not in use, it will allow **uucico**, **cu**, or **ct** to use it for dialing out. If one of these programs attempts to dial out when the line is busy, **uugetty** will deny the requester permission and echo a message indicating that the device is unavailable. The **uugetty** is executed as a function of the **init** program.

The UUCP Daemons and Their Purpose

There are three daemons that are part of Basic Networking. These daemons are routines that run as background processes to handle file transfers and command executions.

uucico: This daemon is referred to as the transport program for UUCP requests. It selects the device used for the link, establishes the link to the remote machine, performs the required login sequence, and performs permission checks. It also transfers "data" and "execute" files, logs results, and notifies specified users of transfer completions via **mail**. When the local **uucico** daemon calls a remote machine, it "talks" to the **uucico** daemon on the remote machine during the session. The **uucico** daemon is executed by several methods. It is started by the **uucp**, **uuto**, and **uux** programs to contact the remote machine after all the required "data," "work," and/or "execute" files have been created. It is also started by the **uusched** and **Uutry** programs.

uuxqt: This daemon is the execution program for remote execution requests. It searches the spool directory for "execute" files (X.) that have been sent from a remote machine. When an X. file is found, **uuxqt** opens it to get the list of data files required for the execution. It then checks to see if the required data files are available and accessible. If the files are present and can be accessed, **uuxqt** will check the *Permissions* file to verify that it has permission to execute the requested command. The **uuxqt** daemon is executed out of the **uudemon.hour** shell script that is started by **cron**.

uusched: This daemon schedules the queued work in the spool directory. Before starting the **uucico** daemon, **uusched** randomizes the order in which remote machines will be called. The **uusched** is executed out of a shell script called **uudemon.hour** that is started by **cron**.

The Supporting Data Base Files and Their Purpose

As mentioned earlier, several of the Basic Networking programs require information contained in support files. These support files are located in the */usr/lib/uucp* directory. The **cu**, **ct**, **uucico**, and **uuxqt** programs require supporting information from the following files:

Devices	This file contains information concerning the location and line speed of the automatic call unit, direct links, and possibly network devices.

Dialers	This file contains character strings required to negotiate with network devices (automatic calling devices) in the establishment of connections to remote computers (non-801-type dialers).
Systems	This file contains information needed by the **uucico** daemon (and possibly the **cu** program) to establish a link to a remote machine. It contains information such as the name of the remote machine, the name of the connecting device associated with the remote machine, when the machine can be reached, the telephone number, the login ID, the password, etc.
Dialcodes	This file contains dial-code abbreviations that may be used in the phone number field of *Systems* file entries.
Permissions	This file defines the level of access granted to machines when they attempt to transfer files or remotely execute commands on your computer.

There are several other files that may be considered part of the supporting data base, but these files are not directly related to the process of establishing a link and transferring files. For this reason, discussion of these files is reserved for the "Administration" section in this chapter.

How Basic Networking Operates

The operation of Basic Networking is briefly described here. There are five programs that allow your computer to communicate with remote machines. The following paragraphs briefly describe what happens when you execute these programs.

ct—Connect a Terminal

The **ct** program instructs your computer to initiate a call to a remote terminal and issue a **getty** to that remote terminal. The **ct** command line must contain the telephone number of the remote terminal. Of course, the remote terminal must be attached to a modem that will automatically answer the call.

When the **ct** command line is issued, the **ct** program will search for an automatic dialer in the *Devices* file with a transfer rate that matches what was specified in the command line. If no transfer rate was specified, it will default

to 1200 bps. When **ct** finds the dialer to be used, it attempts to dial the telephone number specified in the command line. If no dialer is available, **ct** asks if it should wait for an available dialer and, if so, how many minutes it should wait. An option is available to override this dialogue. When the modem at the remote terminal answers the call from your computer, it is issued a **getty** (login) process. At this point, the user at the remote terminal may attempt to log in.

The user at a remote terminal can call your computer, log in, and request that the computer call the remote terminal back using the **ct** command. If this scenario is used, the remote user will issue a **ct** command, and the link from the remote terminal is dropped. After **ct** finds an available dialer in the *Devices* file, it will call the remote terminal back.

cu—Call a UNIX System

The **cu** command enables you to call another machine and log in as a remote user. The telephone number or node name of the remote machine is required in the command line. If the telephone number is specified, it is passed on to the automatic dial modem. If a system name is specified, the telephone number is obtained from the associated *Systems* file entry. If an automatic dial modem is not used to establish the connection, the line (port) associated with the direct link to the remote machine can be specified in the command line.

If an automatic dial modem is used, the **cu** program will search for an automatic dialer in the *Devices* file with a transfer rate that matches what was specified in the command line. If no speed is specified, the first dialer listed (if available) will be used regardless of its transfer rate. After the link has been established and you have successfully completed the login process, you will be logged in on both computers. This will allow you to execute commands on either computer and/or transfer ASCII coded files from one computer to another. After you have terminated the connection, you will still be logged in on your computer (calling computer). This command can only be executed by an active computer.

uucp—UNIX System-to-UNIX System Copy

The **uucp** command will allow you to transfer file(s) to a remote computer without knowing any details of the connection. All that you are required to know is the name of the remote computer and possibly the login ID of the remote user to whom the file(s) is being sent. The details of the connection are kept in the *Systems* file.

When you enter a **uucp** command, the **uucp** program creates a "work" file and possibly a "data" file for the requested transfer. The work file contains information required for transferring the file(s). The data file is simply a copy of the specified source file. After these files have been created in the spool directory, the **uucico** daemon will start.

The **uucico** daemon attempts to establish a connection to the remote machine that is to receive the file(s). It first gathers the information required for establishing a link to the remote machine from the *Systems* file. This is how **uucico** knows what type of device to use in establishing the link. Then, **uucico** searches the *Devices* file looking for the devices that match the requirements listed in the *Systems* file. After **uucico** has found an available device, it will attempt to establish the link and log in on the remote machine.

When **uucico** logs in on the remote machine, it starts the **uucico** daemon on the remote machine. The two **uucico** daemons then negotiate the line protocol to be used in the file transfer(s). The local **uucico** daemon then transfers the file(s) to the remote machine, and the remote **uucico** places the file in the specified pathname(s) on the remote machine. After your computer completes the transfer(s), the remote machine may send files that are queued for your computer. The remote machine can be denied permission to transfer these files with an entry in the *Permissions* file. If this is done, the remote machine must establish a link to your computer to perform the transfers. If the remote machine or the device selected to make the connection to the remote machine is unavailable, the request will remain queued in the spool directory. Each hour, **cron** starts **uudemon.hour**, which in turn starts the **uusched** daemon. When the **uusched** daemon starts, it searches the spool directory for the remaining work files, generates the random order in which these requests are to be processed, and then starts the transfer process (**uucico**) described in the previous paragraphs.

The transfer process described generally applies to an active machine. An active machine (one with calling hardware and Basic Networking software) can be set up to "poll" a passive machine. A passive machine can queue file

transfers (because it has Basic Networking software), but it cannot call the remote machine because it does not have the required hardware. The *Poll* file (*/usr/lib/uucp/Poll*) contains a list of machines that are to be polled in this manner. For additional information, refer to the discussion on the *Poll* file and **uudemon.poll** in the "Administration" section of this chapter.

uuto—Public UNIX System-to-UNIX System Copy

The **uuto** program uses the **uucp** program to build work files and data files in the spool directory for requested transfers. The difference is that the **uuto** command will not allow you to specify a pathname as a destination for the file. The **uuto** command automatically puts the file in a directory under */usr/spool/uucppublic/receive*. Once the transfer is complete, mail is sent to the appropriate user indicating that a file has arrived and was placed in the public area. That user can then use the **uupick** command to retrieve that file. The **uupick** command will search the public area for files destined to the user and allow the user to interactively delete, print, or move the file to a named directory.

uux—UNIX System-to-UNIX System Execution

The **uux** command allows commands to be executed on a remote machine. It gathers files from various computers, executes the specified command on these files, and sends the standard output to a file on the specified computer. This can be useful when some of the required resources (commands and/or files) are not present on your computer. Remote mail is implemented using the **uux** program, but its execution is embedded in the standard **mail** command. For security, many machines will limit the list of commands that can be executed via **uux** to the default (receipt of mail).

When the **uux** command is issued, the **uux** program creates an "execute" (X.) file that contains the names of the files required for execution, your login name, the destination of the standard output, and the command to be executed. **Uux** also creates "work" (C.) files that are used to gather the files required for execution. These files are then sent to the remote machine, along with the "execute" file, by the **uucico** daemon and placed in the remote spool directory.

Periodically, the **uuxqt** daemon on the remote machine is started to search for X. files in the spool directory. Upon finding an X. file, the **uuxqt** daemon checks to see if all the required data files are available and accessible. It then checks the *Permissions* file to verify that the command(s) listed can be performed. After execution, **uuxqt** sends the standard output to a file on the specified computer.

Administration

The files and tasks associated with the operation of Basic Networking is discussed here. The amount of effort required to administer Basic Networking depends on the amount of "traffic" that enters or leaves your computer. For an average computer, little—if any—intervention with the automatic cleanup functions is required. A computer with a large amount of traffic may require more attention as problems arise.

By now, you've probably realized that the "UUCP facilities" make up the bulk of Basic Networking. The UUCP facilities could generally be defined as all of the programs and support files in Basic Networking with the exception of the **ct** and **cu** programs.

Administrative Files

TM—temporary data file

This data file is created under the spool directory (i.e., */usr/spool/uucp/XXXX)* when receiving a file from another machine. The directory "XXXX" has the same name as the remote machine that is sending the file. The temporary data filename has the following format:

TM.*pid.ddd*

where:

pid	is a process ID
ddd	is a sequential 3-digit number starting at zero

After the entire file is received, the *TM.* file is moved to the pathname specified in the command line. If the file was sent via the **uuto** program, the file will be automatically moved to the public area. If processing is abnormally terminated, the *TM.* file may remain in the "XXXX" directory. This file should be periodically removed.

LCK—lock file

The lock file is created in the */usr/spool/locks* directory for each device in use. A lock file prevents duplicate conversations and multiple attempts to use the same calling device. The filename has the following format:

LCK..*str*

where *str* is either a device or computer name. The file may be left in the spool directory if runs abort (usually on computer crashes). The lock file will be ignored (reused) after the parent process is no longer active.

Work (C.) file

The work file is created in a spool directory when work (transfers or remote command executions) has been queued for a remote computer. The name has the following format:

C.*sysnxxx*

where *sys* is the name of the remote computer, *n* is the ASCII character representing the grade (priority) of the work, and *xxxx* is the 4-character job sequence number assigned by UUCP. A work file contains the following information:

- Full pathname of the file to be sent or requested
- Full pathname of the destination or ~user/filename

NOTE: The ~ is shorthand for */usr/spool/uucppublic* and must be included if the full pathname is not used.

- User login name
- List of options
- Name of associated data file in the spool directory (If the **-c** or **-p** option was specified, a dummy name [*D*.0] will be used.)
- Mode bits of the source file
- Remote user's login name to be notified upon completion of the transfer

Data (D.) file

The data file is created when it is specified in the command line to copy the source file to the spool directory. The filename has the following format:

D.*sysnxxxx*

where *sys* is the name of the remote computer, *n* is the character representing the grade (priority) of the work, and *xxxx* is the 4-character job sequence number assigned by **uucp**. The 4-character job sequence number may be followed by a subjob number that is used when there are several *D.* files created for a work (*C.*) file.

Execute (X.) file

The execute file is created in the spool directory prior to remote command executions. The filename has the following format:

X.*sysnxxxx*

where *sys* is the name of the remote computer, *n* is the character representing the grade (priority) of the work, and *xxxx* is the 4-character sequence number assigned by UUCP.

The execute file contains the following information:

- Requester's login and computer name
- Name of file(s) required for execution
- Input to be used as the standard input to the command string
- Computer and filename to receive standard output from the command execution
- Command string
- Option lines for return status requests

Machine Log File

The log file is created for each remote machine with which your computer communicates. Each machine may have four log files, one for **uucico**, **uuxqt**, **uux**, and/or **uucp** requests, depending on the type of communication that has taken place. The log files are kept in the directory */usr/spool/uucp/.Log*. Each day, these log files are combined and stored in the directory */usr/spool/uucp/.Old* when **uudemon.cleanu** is executed. The combined files are kept three days before they're removed. If space is a problem, the administrator may consider reducing the number of days the files are kept by modifying the **uudemon.cleanu** shell file.

Supporting Data Base

The data base that supports Basic Networking is composed of several support files. These support files contain information required by the **uucico** and **uuxqt** daemons during file transfers or remote command executions. All of the support files are located in the */usr/lib/uucp* directory.

Devices File

The *Devices* file (*/usr/lib/uucp/Devices*) contains the information for all of the devices that may be used to establish a link to a remote machine. It contains information for both automatic call units, direct links, and network connections. Although provisions are made for several types of devices, only Modems and Direct Links are supported by AT&T.

This file works very closely with the *Dialers*, *Systems*, and *Dialcodes* files. It may be beneficial to become familiar with these files before attempting to gain an understanding of the *Devices* file.

Each entry in the *Devices* file has the following format:

Type Line Line2 Class Dialer-Token-Pairs (DTP)

where each field (separated by a space) is defined in the following paragraphs.

`Type`: This field may contain one of five keywords:

`Direct`	This keyword indicates a Direct Link to another computer (for **cu** connections only).
`ACU`	This keyword indicates that the link to a remote computer is made through an automatic call unit (Automatic Dial Modem). This modem may be connected either directly to the computer or indirectly through a Local Area Network (LAN) switch.
`Network`	This keyword indicates that the link is established through a LAN switch where `Network` is replaced with either `micom` or `develcon`. These two LAN switches are the only ones that contain caller scripts in the *Dialers* file. Other switches may be used if caller scripts are constructed and placed in the *Dialers* file.
`Modem Control`	This keyword causes the device to be opened with O_NDELAY set (so the open does not hang waiting for carrier). After the open, O_NDELAY is cleared.

System-Name This keyword indicates a direct link to a particular machine where *System-Name* is replaced by the name of the particular computer. This naming scheme is used to convey the fact that the line associated with this *Devices* entry is for a particular machine.

The keyword used in the `Type` field is matched against the third field of *Systems* file entries as follows:

Devices: ACU tty1,M - 1200 penril

Systems: eagle Any ACU 1200 3-2-5-1 ogin: nuucp ssword: Oakgrass

`Line`: This field contains the device name of the line (port) associated with the *Devices* entry. For instance, if the Automatic Dial Modem for a particular entry was attached to the */dev/tty1* line, the device name would be tty1. The **,M** indicates that modem control is being used.

`Line2`: If the ACU keyword was used in the Type field and the ACU is an 801-type dialer, this field would contain the device name of the 801 dialer. It should be noted that 801-type ACUs do not contain a modem. Therefore, a separate modem is required and would be connected to a different line (defined in the Line field). This means that one line would be allocated to the modem and another to the dialer. Since the computer will not normally use this type of configuration, this field is ignored but must contain a pseudo entry as a placeholder (use a `"-"` as a placeholder).

`Class`: If an ACU keyword is used, this may be just the speed of the device. It may contain a letter and speed (e.g., C1200, D1200, etc.) to differentiate between classes of dialers (centrex or DIMENSION PBX). This is necessary because many larger offices may have more than one type of telephone network. One network may be dedicated to serving only internal office communications while the other handles the external communications. Therefore, it is necessary to distinguish which line(s) should be used for internal communications and which should be used for external communications. The same distinction must be made in the *Systems* file because a match is made against the fourth field of *Systems* file entries as follows:

Devices: ACU tty1,M - **D1200** penril

Systems: eagle Any ACU **D1200** 3-2-5-1 ogin: nuucp ssword: Oakgrass

Some devices can be used at any speed, so the keyword `Any` may be used in the Class field. If `Any` is used, the line will match any speed requested in a *Systems* entry. If this field is `Any` and the *Systems* Class field is `Any`, the speed will default to 1200 bps.

`Dialer-Token-Pairs`: This field contains pairs of dialers and tokens. The "dialer" portion may be an automatic dial modem or "direct" for Direct Link devices. The "token" portion may be supplied immediately following the dialer; or if not present, it can be taken from the *Systems* file. This field has the following format:

```
dialer-token dialer-token
```

where the last pair may or may not be present, depending on the associated device (dialer). In most cases, the last pair will contain only a dialer, and the token is retrieved from the Phone field of the *Systems* entry. The DTP field may be structured four different ways, depending on the device associated with the entry:

1. If a direct link is established to a particular computer, the DTP field of the associated entry will contain the keyword "direct." This is true for both types of direct link entries, Direct and System-Name (refer to discussion on the Type field).

2. If an automatic dialing modem is connected directly to a computer port, the DTP field of the associated *Devices* entry will only have one pair. This pair would normally be the name of the modem. This name is used to match the particular *Devices* entry with an entry in the *Dialers* file. Therefore, this dialer must match the first field of a *Dialers* file entry as follows:

   Devices: ACU tty1,M - 1200 **ventel**

   Dialers: **ventel** =&-% "" \M\r\p\r\c $ <K\T%%\r>\c ONLINE!\m

   Notice that only the dialer (**ventel**) is present in the DTP field of the *Devices* entry. This means that the token to be passed on to the dialer (in this case the telephone number) is taken from the Phone field of a *Systems* file entry.

3. If an automatic dialing modem is connected to a local area network (LAN), the computer must first access the switch and the switch will make the connection to the automatic dialing modem. This type of

entry would have two pairs. The dialer portion of each pair (fifth and seventh fields of entry) is used to match entries in the *Dialers* file as follows:

Devices: ACU tty1 - 1200 **develcon** vent **ventel**

Dialers: **ventel** =&-% "" \M\r\p\r\c $ <K\T%%\r>\c ONLINE!\m
Dialers: **develcon** "" "" \pr\ps\c est:\077 \E\D\e \007

In the first pair, develcon is the dialer and vent is the token that is passed to the Develcon switch to tell it which device (ventel modem) to connect to the computer. This token would be unique for each LAN switch since each switch may be set up differently. Once the ventel modem has been connected, the second pair is accessed where ventel is the dialer and the token is retrieved from the *Systems* file.

4. If a machine that you want to communicate with is on the same local network switch as your computer, your computer must first access the switch and then the switch can make the connection to the other machine. In this type of entry, there is only one pair. The dialer portion is used to match a *Dialers* entry as follows:

   Devices: develcon tty1 - 1200 **develcon** \D

   Dialers: **develcon** "" "" \pr\ps\c est:\007 \E\D\e \007

   As shown, the "token" is left blank. This indicates that it's retrieved from the *Systems* file. The *Systems* file entry for this particular machine will contain the token in the Phone field that is normally reserved for the telephone number of the machine (refer to "Systems File" Phone field). This type of DTP contains an escape character (\D) which ensures that the contents of the Phone field will not be interpreted as a valid entry in the *Dialcodes* file.

There are two escape characters that may appear at the end of a DTP field:

\T — Indicates that the Phone (token) field should be translated using the *Dialcodes* file. This escape character is normally placed in the *Dialers* file for each caller script associated with an automatic dial modem (penril, ventel, etc.). Therefore, the translation will not take place until the caller script is accessed.

`\D` Indicates that the Phone (token) field should not be translated using the *Dialcodes* file. If no escape character is specified at the end of a `Devices` entry, the `\D` is assumed (default). A `\D` is also used in the *Dialers* file with entries associated with network switches (develcon and micom).

Dialers File

The *Dialers* file (*/usr/lib/uucp/Dialers*) is used to specify the initial handshaking that must take place on a line before it can be made available for transferring data. This initial handshaking is usually a sequence of ASCII strings that are transmitted and expected and is often used to dial a telephone number using an ASCII dialer (such as the AT&T 2212C Modem). As shown in the above examples, the fifth field in a *Devices* file entry is used as an index into the *Dialers* file. Here an attempt is made to match the *Devices* field with the first field of each *Dialers* entry. In addition, each odd numbered *Devices* field starting with the seventh position is used as an index into the *Dialers* file. Changes must be made using one of the editors (**ed** or **vi**).

If the match succeeds, the *Dialers* entry is interpreted to perform the dialer negotiations. The first field matches the fifth and additional odd numbered fields in the *Devices* file. The second field is used as a translate string (the first of each pair of characters is mapped to the second character in the pair). This is usually used to translate "=" and "-" into whatever the dialer requires for "wait for dial tone" and "pause." The remaining fields are "expect-send" strings. The following *Dialers* file entries are typical examples:

```
att4000  =,-,      "" \M\dat\r\c OK\r \EATDT\T\r\c CONNECT \m\c
penril   =W-P      "" \d > s\p9\c )-W\p\r\ds\p9\c-) y\c : \E\TP > 9\c OK
ventel   =&-%      "" \M\r\p\r\c $ <K\T%%\r>\c ONLINE!\m
hayes    =,-,      "" \M\dAT\r\c OK\r \EATDT\T\r\c CONNECT\m\c
rixon    =&-%      "" \d\r\r\c $ s9\c )-W\r\ds9\c-) s\c : \T\r\c $ 9\c LINE
vadic    =K-K      "" \005\p *-\005\p-*\005\p-* D\p BER? \E\T\e \r\c LINE
develcon ""        "" \pr\ps\c est:\007 \E\D\e \007
micom    ""        ""\s\c NAME? \D\r\c GO
direct
```

The meaning of some of the escape characters (those beginning with "\") used in the *Dialers* file are shown in the following list:

`\p`	pauses (approximately ¼ to ½ second)
`\d`	delays (approximately 2 seconds)
`\D`	phone number or token without `Dialcodes` translation
`\M`	sets no modem control
`\T`	phone number or token with `Dialcodes` translation
`\K`	inserts a BREAK
`\E`	enables echo checking (for slow devices)
`\e`	disables echo checking
`\r`	carriage return
`\c`	no new-line
`\m`	restores modem control
`\n`	sends new-line
`\nnn`	sends octal number

Additional escape characters that may be used are listed in the section discussing the *Systems* file. The penril entry in the *Dialers* file is executed as follows. First, the telephone number argument is translated, replacing any `"="` with a `"W"` (wait for dialtone) and replacing any `"-"` with a `"P"` (pause). The handshake given by the remainder of the line works as follows:

`""`	Waits for nothing.
`\d`	Delays for 2 seconds.
`>`	Waits for a `">"`.
`s\p9\c`	Sends an `"s,"` pauses for ½ second, sends a `"9,"` sends no terminating new-line.
`)-W\p\r\ds\p9\c-)`	Waits for a `")"`. If it is not received, processes the string between the `"-"` characters as follows. Sends a `"W,"` pauses, sends a carriage return, delays, sends an `"s,"` pauses, sends a `"9"` without a new-line and then waits for the `")"`.

`y\c`	Sends a "y" without a new-line.
`:`	Waits for a ":".
`\M`	Sets no modem control (CLOCAL).
`\m`	Restores modem control. Typically, CLOCAL is set for the duration of the dialer chat, then cleared (so **uucico**, **cu**, or **ct** will detect dropped lines) once connected to the remote system.
`\E\TP`	Enables echo checking. (From this point on, whenever a character is transmitted, it will wait for the character to be received before doing anything else.) Then, sends the telephone number followed by a pause character (P). The `\T` means take the telephone number passed as an argument and apply the *Dialcodes* translation and the modem function translation specified by field number 2 of this entry.
`>`	Waits for a ">".
`9\c`	Sends a "9" without a new-line.
`OK`	Waits for the string "OK."

Systems File

The *Systems* file (*/usr/lib/uucp/Systems*) contains the information needed by the **uucico** daemon to establish a communication link to a remote machine. Each entry in the file represents a machine that can be called by the computer. Furthermore, only those machines listed in the *Systems* file will be permitted to communicate with your computer via Basic Networking (UUCP) unless the execute permissions for **remote.unknown** are changed to permit communications with other machines (refer to "**remote.unknown**"). More than one entry may be present for a particular machine. The additional entries represent alternate communication paths that will be tried in sequential order.

Each entry in the *Systems* file has the following format:

System-Name `Time Type Class Phone Login`

where each field is defined in the following paragraphs.

System-name: This field contains the node name of the remote machine.

`Time`: This field is a string that indicates the day of week and time of day when the remote machine can be called. The day portion may be a list containing some of the following:

Su Mo Tu We Th Fr Sa

`Wk:` For any weekday.

`Any:` For any day.

`Never:` For a passive arrangement with the remote machine. In this case, the computer will never initiate a call to the remote machine. The call must be initiated by the remote machine. The computer is in a passive mode in respect to the remote machine. (See discussion of *Permissions* file.)

The time should be a range of times such as 0800-1230. If no time portion is specified, any time of day is assumed to be allowed for the call. Note that a time range that spans 0000 is permitted. For example, 0800-0600 means all times are allowed other than times between 6 a.m. and 8 a.m. An optional subfield is available to specify the minimum time (in minutes) before a retry following a failed attempt. The subfield separator is a semicolon (;). For example, **"Any ;9"** is interpreted as call any time, but wait at least 9 minutes before retrying if a failure occurs.

`Type`: This field contains the device type that should be used to establish the communication link to the remote machine. The *Devices* file is searched for the device type listed and the device found is used to establish the connection (if available). The following keywords may appear in this field:

`ACU` This keyword indicates that the link to a remote computer is made through an automatic call unit (Automatic Dial Modem). This modem may be connected either directly to the computer or indirectly through a Local Area Network (LAN) switch.

`Network` — This keyword indicates that the link is established through a LAN switch where `Network` is replaced with either `micom` or `develcon`. These two LAN switches are the only ones that contain caller scripts in the *Dialers* file. Other switches may be used if caller scripts are constructed and placed in the *Dialers* file.

System-Name — This keyword indicates a direct link to a particular machine where *System-Name* is replaced by the name of the particular computer (should be same as field one).

The keyword used in this field is matched against the first field of *Devices* file entries as follows:

Systems: eagle Any **ACU** D1200 3-2-5-1 ogin: nuucp ssword: Oakgrass

Devices: **ACU** tty1 - D1200 penril

`Class`: This field is used to indicate the transfer speed of the device used in establishing the communication link. It may contain a letter and speed (e.g., C1200, D1200, etc.) to differentiate between classes of dialers (refer to the discussion on the "Devices File," Class field). Some devices can be used at any speed, so the keyword "Any" may be used. This field must match the Class field in the associated *Devices* entry as follows:

Systems: eagle Any ACU **D1200** 3-2-5-1 ogin: nuucp ssword: Oakgrass

Devices: ACU tty1 - **D1200** penril

`Phone`: This field is used to provide the telephone number (token) of the remote machine for automatic dialers (LAN switches). The telephone number is made up of an optional alphabetic abbreviation and a numeric part. The abbreviation must be one that is listed in the *Dialcodes* file. In this string, an equals sign (=) tells the ACU to wait for a secondary dial tone before dialing the remaining digits. A dash in the string (-) instructs the ACU to pause 4 seconds before dialing the next digit.

If your computer is connected to a LAN switch, you may access other machines that are connected to that switch. The **Systems** entries for these machines will not have a telephone number in the Phone field. Instead, this

field will contain the "token" that must be passed on to the switch so it will know which machine the computer wishes to communicate with. The associated **Devices** entry should have a \D at the end of the entry to ensure that this field is not translated using the *Dialcodes* file. For direct connections, the telephone field is ignored. A "-" should be used as a place holder.

`Login`: This field contains the login information given as a series of fields and subfields of the following format:

[*expect send*] ...

where *expect* is the string that is received and *send* is the string that is sent when the *expect* string is received. The expect field may be made up of subfields of the following form:

expect[*-send-expect*]...

where the *send* is sent if the prior *expect* is not successfully read and the *expect* following the *send* is the next expected string. For example, with "login--login," UUCP will expect "login." If UUCP gets "login," it will go on to the next field. If it does not get login, it will send nothing followed by a new-line, then look for login again. If no characters are initially expected from the remote machine, the characters "" (null string) should be used in the first expect field. Note that all send fields will be sent followed by a new-line unless the send string is terminated with a `\c`.

There are several escape characters that cause specific actions when they're a part of a string sent during the login sequence. The following escape characters are useful in UUCP communications:

`\N`	Sends a null character.
`\b`	Sends a backspace character.
`\c`	If at the end of a string, suppresses the new-line that is normally sent. Ignored otherwise.
`\d`	Delays 2 seconds before sending or reading more characters.
`\p`	Pauses for approximately ¼ to ½ second.
`\n`	Sends a new-line character.
`\r`	Sends a carriage return.
`\s`	Sends a space character.
`\t`	Sends a tab character.

`\\`	Sends a \ character.
`EOT`	Sends an EOT character (actually EOT new line is sent twice).
`BREAK`	Sends a break character.
`\ddd`	Collapses the octal digits (ddd) into a single character and sends that character.

Dialcodes File

The *Dialcodes* file (*/usr/lib/uucp/Dialcodes*) contains the dial-code abbreviations used in the Phone field of the *Systems* file. Each entry has the following format:

abb *dial-seq*

where **abb** is the abbreviation used in the *Systems* file (Phone field), and *dial-seq* is the dial sequence that is passed to the dialer when that particular *Systems* entry is accessed.

The entry

```
jt 9=847-
```

would be set up to work with a Phone field in the *Systems* file such as jt7867. When the entry containing jt7867 is encountered, the sequence 9=847-7867 would be sent to the dialer.

Permissions File

The *Permissions* file (*/usr/lib/uucp/Permissions*) is used to specify the permissions that remote machines have with respect to login, file access, and command execution. Options are provided for restricting the ability to request files and the ability to receive files queued by the local site. In addition, an option is available to specify the commands that a remote site can execute on the local machine. Changes must be made using one of the editors (**vi** or **ed**).

How Entries Are Structured

Each entry is a logical line with physical lines terminated with a \ to indicate continuation. Entries are made up of options delimited by white space. Each option is a name/value pair. These are constructed by an option name followed by an "=" and the value. Note that no white space is allowed within an option assignment.

Comment lines begin with a "#," and they occupy the entire line up to a new-line character. Blank lines are ignored (even within multi-line entries). There are two types of *Permissions* entries:

LOGNAME	Specifies permissions that take effect when a remote machine logs in on (calls) your computer.
MACHINE	Specifies permissions that take effect when your computer logs in on (calls) a remote machine.

LOGNAME entries will contain a *LOGNAME* option, and *MACHINE* entries will contain a *MACHINE* option.

Considerations

The following items should be considered when using the *Permissions* file to restrict the level of access granted to remote machines:

1. All login IDs used by remote machines to log in for UUCP-type communications must appear in one and only one *LOGNAME* entry.
2. Any site that is called whose name does not appear in a *MACHINE* entry will have the following default permissions/restrictions:
   - Local send and receive requests will be executed.
   - The remote machine can send files to your computer */usr/spool/uucppublic* directory.
   - The commands sent by remote machine for execution on your computer must be one of the default commands, usually **rmail**.

Options

This section provides the details of each option, specifying how they're used and their default values.

Request. When a remote machine calls your computer and requests to receive a file, this request can be granted or denied. The *REQUEST* option specifies whether or not the remote machine can request to set up file transfers from your computer. The string

```
REQUEST=yes
```

specifies that the remote machine can request to transfer files from your computer. The string

```
REQUEST=no
```

specifies that the remote machine cannot request to receive files from your computer. The "no" string is the default value. It will be used if the *REQUEST* option is not specified. The *REQUEST* option can appear in either a *LOGNAME* (remote calls you) entry or a *MACHINE* (you call remote) entry.

Sendfiles. When a remote machine calls your computer and completes its work, it may attempt to take work that your computer has queued for it. The *SENDFILES* option specifies whether or not your computer can send the work queued for the remote machine. The string

```
SENDFILES=yes
```

specifies that the computer may send the work that is queued for the remote machine as long as it logged in as one of the names in the *LOGNAME* option. This string is mandatory if the computer is in a "passive mode" with respect to the remote machine. The string

```
SENDFILES=call
```

specifies that files queued in your computer will only be sent when the computer calls the remote machine. The call value is the default for the *SENDFILE* option. This option is only significant in *LOGNAME* entries since *MACHINE* entries apply when calls are made out to remote machines. If the option is used with a *MACHINE* entry, it will be ignored.

Read and Write. These options specify the various parts of the file system that **uucico** can read from or write to. The *READ* and *WRITE* options can be used with either *MACHINE* or *LOGNAME* entries.

The default for both the *READ* and *WRITE* options is the *uucppublic* directory, as shown in the following strings:

```
READ=/usr/spool/uucppublic WRITE=/usr/spool/uucppublic
```

The strings

```
READ=/ WRITE=/
```

specify permission to access any file that can be accessed by a local user with "other" permissions.

The value of these entries is a list of pathnames separated by colons. The *READ* option is for requesting files, and the *WRITE* option is for depositing files. One of the values must be the prefix of any full pathname of a file coming in or going out. To grant permission to deposit files in */usr/news* as well as the public directory, the following values should be used with the *WRITE* option:

```
WRITE=/usr/spool/uucppublic:/usr/news
```

It should be pointed out that if the *READ* and *WRITE* options are used, all pathnames must be specified because the pathnames are not added to the default list. For instance, if the */usr/news* pathname was the only one specified in a *WRITE* option, permission to deposit files in the public directory would be denied.

Noread and Nowrite. The *NOREAD* and *NOWRITE* options specify exceptions to the *READ* and *WRITE* options or defaults. The strings

```
READ=/ NOREAD=/etc WRITE=/usr/spool/uucppublic
```

would permit reading any file except those in the */etc* directory (and its subdirectories - remember, these are prefixes) and writing only to the default */usr/spool/uucppublic* directory. *NOWRITE* works in the same manner as the *NOREAD* option. *NOREAD* and *NOWRITE* can be used in both *LOGNAME* and *MACHINE* entries.

Callback. The *CALLBACK* option is used in *LOGNAME* entries to specify that no transaction will take place until the calling system is called back. The string

```
CALLBACK=yes
```

specifies that your computer must call the remote machine back before any file transfers will take place.

The default for the *CALLBACK* option is

```
CALLBACK=no
```

The *CALLBACK* option is very rarely used. Note that if two sites have this option set to "yes" for each other, a conversation will never get started.

Commands.

The COMMANDS option can be hazardous to the security of your system. Use it with extreme care.

The **uux** program will generate remote execution requests and queue them to be transferred to the remote machine. Files and a command are sent to the target machine for remote execution. The *COMMANDS* option can be used in *MACHINE* entries to specify the commands that a remote machine can execute on your computer. The string

```
COMMANDS=rmail
```

indicates the default commands that a remote machine can execute on your computer. If a command string is used in a *MACHINE* entry, the default commands will be overridden. For instance, the entry

```
MACHINE=owl:raven:hawk:dove \
COMMANDS=rmail:rnews:lp
```

overrides the *COMMAND* default such that the command list for machines `owl`, `raven`, `hawk`, and `dove` now consists of **rmail**, **rnews**, and **lp**. In addition to the names as specified above, there can be full pathnames of commands. For example

```
COMMANDS=rmail:/usr/lbin/rnews:/usr/local/lp
```

specifies that command **rmail** uses the default path. The default paths for the computer are */bin*, */usr/bin*, and */usr/lbin*. When the remote machine specifies **rnews** or **/usr/lbin/rnews** for the command to be executed, **/usr/lbin/rnews** will be executed regardless of the default path. Likewise, **/usr/local/lp** is the **lp** command that will be executed.

Including the *ALL* value in the list means that any command from the remote machine(s) specified in the entry will be executed. If you use this value, you give the remote machine full access to your computer.

The string

```
COMMANDS=/usr/lbin/rnews:ALL:/usr/local/lp
```

illustrates two points. The *ALL* value can appear anywhere in the string, and the pathnames specified for **rnews** and **lp** will be used (instead of the default) if the requested command does not contain the full pathnames for **rnews** or **lp**.

The *VALIDATE* option should be used with the *COMMANDS* option whenever potentially dangerous commands like **cat** and **uucp** are specified with the *COMMANDS* option. Any command that reads or writes files is potentially dangerous to local security when executed by the UUCP remote execution daemon (**uuxqt**).

Validate. The *VALIDATE* option is used with the *COMMANDS* option when specifying potentially dangerous commands. It is used to provide a certain degree of verification of the caller's identity. The use of the *VALIDATE* option requires that privileged machines have a unique login/password for UUCP transactions. An important aspect of this validation is that the login/password associated with this entry be protected. If an outsider gets that information, that particular *VALIDATE* option can no longer be considered secure.

A great deal of consideration should be given to providing a remote machine with a privileged login and password for UUCP transactions. Giving a remote machine a special login and password with file access and remote execution capability is like giving anyone on that machine a normal login and password on your computer. Therefore, if you cannot trust someone on the remote machine, do not provide that machine with a privileged login and password.

The *LOGNAME* entry

```
LOGNAME=uucpfriend VALIDATE=eagle:owl:hawk
```

specifies that if one of the remote machines that claims to be `eagle`, `owl`, or `hawk` logs in on your computer, it must have used the login **uucpfriend**. As can be seen, if an outsider gets the **uucpfriend** login/password, masquerading is trivial. But what does this have to do with the *COMMANDS* option that only appears in *MACHINE* entries? It links the *MACHINE* entry

(and *COMMANDS* option) with a *LOGNAME* entry associated with a privileged login. This link is needed because the execution daemon is not running while the remote machine is logged in. In fact, it is an asynchronous process with no knowledge of what machine sent the execution request. Therefore, the real question is how does your computer know where the execution files came from?

Each remote machine has its own "spool" directory on your computer. These spool directories have write permission given only to the UUCP programs. The execution files from the remote machine are put in its spool directory after being transferred to your computer. When the **uuxqt** daemon runs, it can use the spool directory name to find the *MACHINE* entry in the *Permissions* file and get the *COMMANDS* list, or if the machine name does not appear in the *Permissions* file, the default list will be used.

The following example shows the relationship between the *MACHINE* and *LOGNAME* entries:

```
MACHINE=eagle:owl:hawk REQUEST=yes \
COMMANDS=ALL \
READ=/  WRITE=/

LOGNAME=uucpz VALIDATE=eagle:owl:hawk \
REQUEST=yes SENDFILES=yes \
READ=/  WRITE=/
```

These entries provide unlimited read, write, and command execution for the remote machines `eagle`, `owl`, and `hawk`. The *ALL* value in the *COMMANDS* option means that any command can be executed by either of these machines. Using the *ALL* value gives the remote machine unlimited access to your computer. In fact, files that are only readable or writable by user "uucp" (like *Systems* or *Devices*) can be accessed using commands like **ed**. This means a user on one of the privileged machines can write in the *Systems* file as well as read it.

In the first entry, you must make the assumption that when you want to call one of the machines listed, you're really calling either `eagle`, `owl`, or `hawk`. Therefore, any files put into one of the `eagle`, `owl`, or `hawk` spool directories are put there by one of those machines. If a remote machine logs in and says that it is one of these three machines, its execution files will also be put in the privileged spool directory. You, therefore, have to validate that the machine has the privileged login **uucpz**.

MACHINE Entry for "Other" Systems

You may want to specify different option values for the machines your computer calls that are not mentioned in specific *MACHINE* entries. This may occur when there are many machines calling in, and the command set changes from time to time. The name *OTHER* for the machine name is used for this entry as follows:

```
MACHINE=OTHER \
COMMANDS=rmail:rnews:/usr/lbin/Photo:/usr/lbin/xp
```

All other options available for the *MACHINE* entry may also be set for the machines that are not mentioned in other *MACHINE* entries.

Combining MACHINE and LOGNAME Entries

It is possible to combine *MACHINE* and *LOGNAME* entries into a single entry where the common options are the same. For example, the two entries:

```
MACHINE=eagle:owl:hawk REQUEST=yes \
 READ=/  WRITE=/

LOGNAME=uucpz REQUEST=yes SENDFILES=yes \
 READ=/  WRITE=/
```

share the same *REQUEST*, *READ*, and *WRITE* options. These two entries can be merged into one entry as follows:

```
MACHINE=eagle:owl:hawk REQUEST=yes \
LOGNAME=uucpz SENDFILES=yes \
  READ=/  WRITE=/
```

Sample Permissions Files

Example 1. This first example represents the most restrictive access to your computer.

```
LOGNAME=nuucp
```

It states that login **nuucp** has all the default permissions/restrictions:

- The remote machine can only send files to *uucppublic*.

- The remote machine cannot request to receive files (*REQUEST* option).
- No files that are queued for the remote machine will be transferred during the current session (*SENDFILES* option).
- The only commands that can be executed are the defaults.

This entry alone is sufficient to start communications with remote machines, permitting files to be transferred only to the */usr/spool/uucppublic* directory.

Example 2. The next example is for remote machines that log in but have fewer restrictions. The login and password corresponding to this entry should not be distributed to the general public; it is usually reserved for closely coupled systems where the **Systems** file information can be tightly controlled.

```
LOGNAME=uucpz REQUEST=yes SENDFILES=yes \
  READ=/  WRITE=/
```

This entry places the following permissions/restrictions on a machine that logs in as **uucpz**:

- Files can be requested from your computer (*REQUEST* option).
- Files can be transferred to any directory or any file that is writable by user "other." That is a file/directory that is writable by a local user with neither owner nor group permissions (*WRITE* option).
- Any files readable by user "other" can be requested (*READ* option).
- Any requests queued for the remote machine will be executed during the current session. These are files destined for the machine that has called in (*SENDFILES* option).
- The commands sent for execution on the local machine must be in the default set.

Example 3. The two previous examples showed entries that referred to remote machines when they log in to your computer. This example is an entry used when calling remote machines:

```
MACHINE=eagle:owl:hawk:raven \
 REQUEST=yes READ=/  WRITE=/
```

When calling any of the systems given in the *MACHINE* list, the following permissions prevail:

- The remote machine can both request and send files (*REQUEST* option).
- The source or destination of the files on the local machine can be anywhere in the file system (with read/write option).
- The only commands that will be executed for the remote machine are those in the default set.

Any site that is called that does not have its name in a *MACHINE* entry will have the default permissions as stated in Example 1, with the exception that files queued for that machine will be sent. (The *SENDFILES* option is only interpreted in the *LOGNAME* entry.)

Poll File

The *Poll* file (*/usr/lib/uucp/Poll*) contains information for polling specified machines. Each entry in the *Poll* file contains the name of the remote machine to call, followed by a TAB character, and finally the hours the machine should be called. The entry

```
eagle 0 4 8 12 16 20
```

will provide polling of machine eagle every 4 hours.

NOTE

It should be understood that **uudemon.poll** does not actually perform the poll, it merely sets up a polling work (*C.*) file in the spool directory that will be seen by the scheduler, started by **uudemon.hour**. Refer to the discussion on **uudemon.poll**.

Maxuuxqts File

The *Maxuuxqts* (*/usr/lib/uucp/Maxuuxqts*) file contains an ASCII number to limit the number of simultaneous **uuxqt** programs running. This file is delivered with a default entry of 2. This may be changed to meet local needs. If there is a lot of traffic from **mail**, it may be advisable to increase the number of **uuxqt** programs that will run to reduce the time it takes for the mail to leave your system. However, keep in mind that the load on the system increases with the number of **uuxqt** programs running.

Maxuuscheds File

The *Maxuuscheds* (*/usr/lib/uucp/Maxuuscheds*) file contains an ASCII number to limit the number of simultaneous **uusched** programs running. Each **uusched** running will have one **uucico** associated with it; limiting the number will directly affect the load on the system. The limit should be less than the number of outgoing lines used by UUCP (a smaller number is often desirable). This file is delivered with a default entry of 2. Again, this may be changed to meet the needs of the local system. However, keep in mind that the load on the system increases with the number of **uusched** programs running.

remote.unknown

The *remote.unknown* program (*/usr/lib/uucp/remote.unknown*) is a shell file that is executed when a remote site that is not in the *Systems* file calls in to start a conversation. The shell script will append the name and time information to the file */usr/spool/uucp/.Admin/Foreign*. Since it is a shell, it can be easily modified. For example, it can be set up to send mail to the administrator. The contents of this file, as delivered, is as follows:

```
FOREIGN=/usr/spool/uucp/.Admin/Foreign
echo "'date': call from system $1" >>$FOREIGN
```

If you want to permit machines that are not listed in your *Systems* file to communicate via Basic Networking, remove the execute permissions from the *remote.unknown* file. For example,

```
chmod 444 /usr/lib/uucp/remote.unknown
```

When *remote.unknown* is executable, your computer will hang up if a machine that is not in your *Systems* file calls in (to UUCP) on your system.

Administrative Tasks

There is a minimum amount of maintenance that must be applied to your computer to keep the files updated, to ensure that the network is running properly, and to track down line problems. When more than one remote machine is involved, the job becomes more difficult because there are more files to update and because users are much less patient when failures occur between machines that are under local control. The **uustat** program provides you with information about the latest attempts to contact various machines

and the age and number of jobs in the queue for remote machines. The following sections describe the routine administrative tasks that must be performed by someone acting as the UUCP administrator or are automatically performed by the UUCP daemons (demons).

The biggest problem in a dialup network like UUCP is dealing with the backlog of jobs that cannot be transmitted to other machines. The following cleanup activities should be routinely performed.

Cleanup of Undeliverable Jobs

The **uustat** program should be invoked regularly to provide information about the status of connections to various machines and the size and age of the queued requests. The **uudemon.admin** shell should be started by **cron** at least once per day. This will send the administrator the current status. Of particular interest is the age (in days) of the oldest request in each queue, the number of times a failure has occurred when attempting to reach that machine, and the reason for failure. In addition, the age of the oldest execution request (*X.* file) is also given.

The **uudemon.cleanu** shell file is set up to remove any jobs that have been queued for several days and cannot be sent. Leftover data (*D.*) and work (*C.*) files are removed after 7 days, and execute (*X.*) files are removed after 2 days. It also provides feedback to the user indicating when jobs are not being accomplished and when these jobs are being deleted.

Cleanup of the Public Area

To keep the local file system from overflowing when files are sent to the public area, the **uudemon.cleanu** procedure is set up with a **find** command to remove any files that are older than 7 days and directories that are empty. This interval may need to be shortened by changing the **uudemon.cleanu** shell file if there is not sufficient space to devote to the public area.

Since the spool directory is very dynamic, it may grow large before transfers take place. Therefore, it is a good idea to reorganize its structure. The best way to do this on your computer is to use the **crontab** command to clean out the spool directory at a specified time.

First, specify the file you want to have the cleanup code in as follows:

crontab *clean.wk* Enter

The *clean.wk* file will contain the code for all files cleaned at a specified time (every Monday, for example), based on the time specified in the *crontab* file. You may already have entries in *clean.wk* which means you will also have the cleanup time specified. See *crontab*(1) in the *User's/System Administration Reference Manual* for additional information. If you wish to specify a new cleanup time, first, make a new file with the **crontab** command as above. Edit the *crontab* file to specify the time of cleanup. For example,

```
0 0 1 15 * 1
```

in the *crontab* file would indicate cleanup on the first and fifteenth of each month, as well as on every Monday. In the file you specified with the **crontab** command, enter the following code (the # sign lines are comment lines):

```
#     Clean up /usr/spool/uucp
#     Most cleanup is now done by uudemon.cleanu
#     so just copy out and back.
#
echo "UUCP SPOOL DIRECTORIES CLEANUP STARTED"
#
cd /usr/spool/uucp
mkdir ../nuucp
chown uucp ../nuucp
chgrp uucp ../nuucp
find . -print|cpio -pdml ../nuucp
cd ..
mv uucp ouucp
mv nuucp uucp
rm -rf ouucp
rm -f /usr/spool/locks/LCK*
#
#       Note:
#       Change the tty?? device to the
#       device you are using for UUCP.
#       For example change tty?? to tty01.
#
chown uucp /dev/tty??
chgrp uucp /dev/tty??
chmod 0644 /dev/tty??
chmod 0222 /dev/tty??
echo "UUCP SPOOL DIRECTORIES CLEANUP FINISHED"
```

Compaction of Log Files

This version of Basic Networking has individual log files for each machine and each program. For example, machine eagle has a log file for **uucico** requests and a log file for **uuxqt** execution requests. The **uulog** program gives the user access to the information in these files by machine name. These files are combined and stored in directory */usr/lib/uucp/.Old* whenever **uudemon.cleanu** is executed. This shell script saves files that are 2 days old. The 2 days can be easily changed by changing the appropriate line in the **uudemon.cleanu** shell. If space is a problem, the administrator might consider reducing the number of days the files are kept.

Cleanup of sulog and cron log

The */usr/adm/sulog* and */usr/lib/cron/log* files are both indirectly related to UUCP transactions. The *sulog* file contains a history of the **su** command usage. Since each **uudemon** entry in the */usr/spool/cron/crontab/root* file uses the **su** command, the *sulog* could become rather large over a period of time. The *sulog* should be purged periodically to keep the file at a reasonable size.

Similarly, a history of all processes spawned by **/etc/cron** are recorded in */usr/lib/cron/log*. The *cron/log* file will also become large over a period of time and should be purged periodically to limit its size.

UUCP and Cron

The **cron** daemon is a tool that proves to be very useful in the administration of UNIX Systems. When the computer is in run state 2 (multi-user), **cron** scans the */usr/spool/cron/crontab/root* file every minute for entries that contain "work" scheduled to be executed at that time. It is recommended that the UUCP administrator make use of **cron** to aid in the administration of Basic Networking.

As delivered, Basic Networking contains four entries in the *root crontab* file. Each one of these entries executes shell scripts that are used for various administrative purposes. These shell scripts can be easily modified to meet the needs of your system.

uudemon.admin

The **uudemon.admin** shell script mails status information to the UUCP administrative login (**uucp**) using **uustat** commands with the **-p** and **-q** options. Refer to the **uustat** manual page for interpretation of these options.

The **uudemon.admin** shell script should be executed daily by an entry in the *root crontab* file. The default *root crontab* entry for **uudemon.admin** is as follows:

```
48 11,14, ** 1-5 /bin/su uucp -c "/usr/lib/uucp/uudemon.admin >
/dev/null 2>&1"
```

uudemon.cleanu

The **uudemon.cleanu** shell script cleans up the Basic Networking log files and directories. Archived log files are updated so that no log information over 3 days old is kept. Log files for individual machines are taken from the */usr/spool/uucp/.Log* directory, merged, and placed in the */usr/spool/uucp/.Old* directory along with the older log information. Files and directories that are no longer needed in the spool directories are removed. After cleanup is performed, the UUCP administrative login (**uucp**) is mailed a summary of the status information gathered during the current day.

The **uudemon.cleanu** shell script should be executed by an entry in the *root crontab* file. It can be run daily, weekly, or whenever, depending on the amount of UUCP traffic that enters and leaves your computer. The default *root crontab* entry for **uudemon.cleanu** is as follows:

```
45 23 * * * ulimit 5000; /bin/su uucp -c
"/usr/lib/uucp/uudemon.cleanu > /dev/null 2>&1"
```

If log files get very large, the ulimit may need to be increased.

uudemon.hour

The **uudemon.hour** shell script is used to call UUCP programs on an hourly basis. The **uusched** program is called to search the spool directory for work files (*C.*) that have not been processed and schedule these files for transfer to a remote machine. The **uuxqt** daemon is called to search the spool directory for execute files (*X/C .*) that have been transferred to your computer and were not processed at the time they were transferred.

The **uudemon.hour** shell script should be executed by an entry in the *root crontab* file. If the amount of traffic leaving and entering your computer is large, it may be started once or twice an hour. If it is small, it may be started once every 4 hours or so. The default *root crontab* entry for **uudemon.hour** is as follows:

```
26,56 * * * * /bin/su uucp -c
"/usr/lib/uucp/uudemon.hour > /dev/null"
```

uudemon.poll

The **uudemon.poll** shell script is used to poll the remote machines listed in the *Poll* file (*/usr/lib/uucp/Poll*). It creates work files (C.) for machines according to the entries listed in the *Poll* file. It should be set up to run once an hour just prior to **uudemon.hour** so that the work files will be present when **uudemon.hour** is called.

The **uudemon.poll** script should be executed by an entry in the *root crontab* file. The exact times it runs is dependent on the scheduling of **uudemon.hour**. The default *root crontab* entry for **uudemon.poll** is as follows:

```
40 * * * * /bin/su uucp -c "/usr/lib/uucp/uudemon.poll >
/dev/null"
```

Notice how **uudemon.poll** is scheduled to run 11 minutes before **uudemon.hour** runs.

Inittab Entries

The */etc/inittab* file contains information for the processes to be spawned on the computer devices, including the ports. Ports that are used by Basic Networking are normally bidirectional ports. Bidirectional ports can be used to receive incoming calls, as well as place outgoing calls. The **uugetty** program is used in place of **getty** for those bidirectional ports associated with Basic Networking. After **uucp** has been set up on a line, for example tty00, the next step is to enable a **uugetty** login on that line. This can be done by editing */etc/inittab* to add the new tty and then telling **init** to re-read *inittab*. Do the following:

1. Edit */etc/inittab*, and add the following line:

   ```
   :23:respawn:/usr/lib/uucp/uugetty -r tty00 1200
   ```

2. Tell **init** to re-read the */etc/inittab* file:

   ```
   init q
   ```

UUCP Logins and Passwords

There are two login IDs associated with Basic Networking: one is the UUCP administrative login, **uucp**, and the other is an access login, **nuucp**, used by remote computers to access your computer. These logins should not be changed from their default settings of **uucp** and **nuucp**.

The **uucp** administrative login is the owner of all the UUCP object and spooled data files. The following is a sample entry in the */etc/passwd* file for the administrative login:

```
uucp:zAvLCKp:5:1:UUCP.Admin:/usr/lib/uucp:
```

The **nuucp** access login allows remote machines to log in on your computer. The following is a sample entry in the */etc/passwd* file for the access login:

```
nuucp:zaaAA:6:1:UUCP.Admin:/usr/spool/uucppublic:
/usr/lib/uucp/uucico
```

Notice that the standard shell is not given to the **nuucp** login. The shell that **nuucp** receives is the **uucico** daemon that controls the conversation when a remote machine logs in to your machine.

The assigning of passwords for the **uucp** and **nuucp** logins is left up to the administrator. The passwords should be at least six to eight characters. Only the first eight characters of the passwords are significant. If the password for the access login is changed for security reasons, make certain that the remote machines that are a part of your network are properly notified of the change.

Chapter 9: Remote File Sharing Administration

Overview of Remote File Sharing

Remote File Sharing (RFS) allows computers running UNIX System V/386 to selectively share resources (directories containing files, subdirectories, devices, and/or named pipes) across a network. As an administrator of an computer on an RFS network, you can choose directories on your system you want to share and add them to a list of available resources on the network. From this list, you can choose resources on remote computers that you would like to use on your computer.

Resource Sharing

Sharing a resource on an RFS system begins with a pathname to a UNIX system directory. If there is a directory you want to share, assign it a resource identifier and "advertise" it to other machines, using the **adv** command. The resource identifier is how other machines reference that directory. Computers that pass the security checks you have set up can then mount your resource as they would mount a file system locally. The **mount** command with the **-d** option is used for mounting remote resources.

Figure 9-1 shows how two computers can share resources. In this example, the administrator of a computer named **fie** on an RFS system wants to share all files and directories under */fs1* on its file system tree. The administrator advertises */fs1* as a resource called FSLOGS.

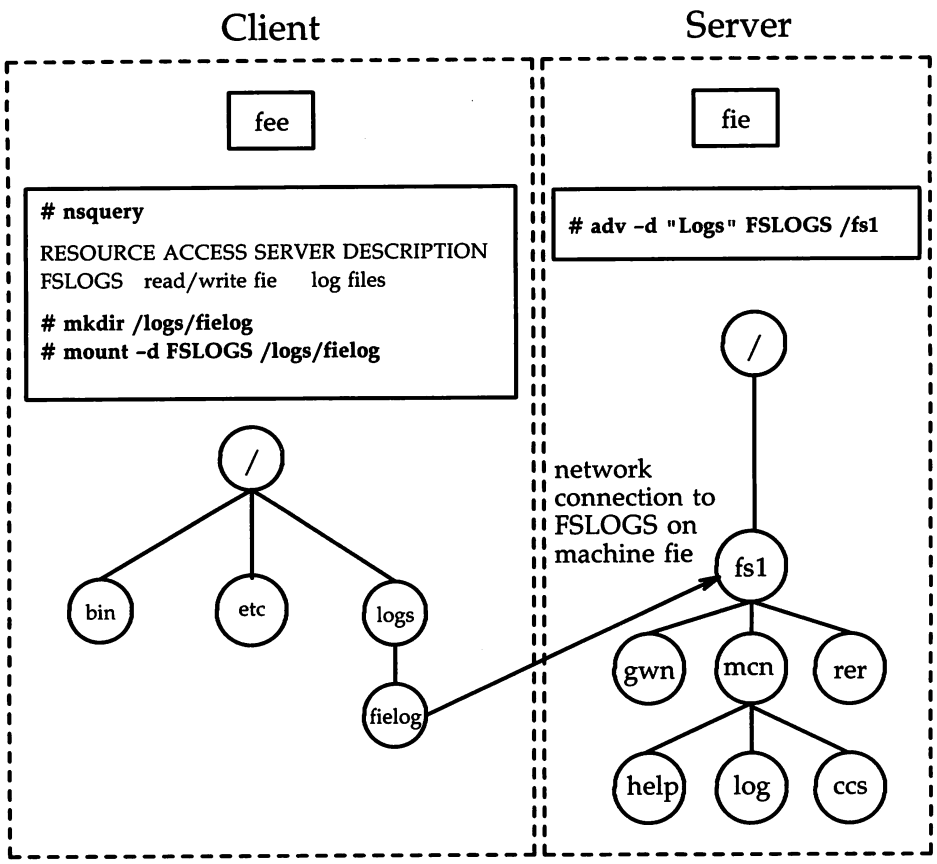


Figure 9-1: Example—Sharing Resources

Another machine in **fie**'s domain is called **fee**. The administrator from **fee** uses the **nsquery** command to see that FSLOGS is available on **fie**. **fee**'s administrator then creates a directory called */logs/fielog* on **fee** (**mkdir** command) and **mount**s FSLOGS on */logs/fielog*.

Files or subdirectories from */fs1* are now accessible to users on **fee**. Users can change to the remote directory, list the contents, and run a remote program locally. If the resource contained the */dev* directory, users could direct output to a remote device as though the device were on the local machine.

Domains

Each machine on an RFS network must be assigned to a domain. The main reasons for domains are to simplify name service and provide a focal point for security of a group of machines.

Domain names act like telephone area codes. You can address all computers and resources in your domain directly. For outside domains, simply attach the domain name to the node name or resource identifier. This becomes increasingly valuable as RFS networks expand.

Name Service

Each domain must be assigned a primary and zero or more secondary domain name server. These machines can share resources like any other computer in the domain, but they have some special responsibilities.

Primary

The main duty of the primary domain name server is to keep track of all computers and resources within the domain it serves. It ensures that all resource identifiers and machine node names are unique within the domain.

A required task of the primary is to add each computer to the Domain Member List and assign its RFS password.

A list of advertised resources are automatically stored on the primary, so any computer can see a complete list of available resources for the domain. Also, when a computer advertises a resource, it registers its network address with the primary. Therefore, when a computer tries to mount another machine's resources, the primary can tell the computer where the resource can be found on the network.

An optional function of the primary is to gather lists of each computer's users (*/etc/passwd*) and groups (*/etc/group*). Each computer in the domain can then use

these lists to specifically define the permissions each machine's users will have to its resources.

A primary can also gather names and network addresses of other domains' name servers. Once the primary knows another domain name server's address, machines in its domain have the potential to access resources from any machine in the other domain.

Secondaries

If the primary fails, domain name service functions are automatically assumed by one of the secondary domain name servers. The secondary is intended to take over temporarily, until the primary comes back up.

While the secondary will have information needed to run the domain name server, domain information should not be modified on the secondary. As soon as the primary comes back up, the secondary should be instructed to pass name server responsibility back to the primary (**rfadmin -p** command). Then the primary's administrator can change the domain member list, edit the *rfmaster* file, or gather optional user and group information again on the primary.

Transport Provider

The transport provider provides the pathway used by RFS to communicate with other machines. The term transport provider is used to refer to the physical network that connects the machines and the software needed to send messages across the network.

RFS can communicate using any transport provider that is compatible with the AT&T Transport Interface Specification. The STARLAN network is one transport provider that can be used with RFS.

Although the transport provider is not considered part of the RFS package, RFS will not work if the transport provider is not functioning properly. Also, some information needed to configure RFS varies from one transport provider to another. For example, network addresses of the primary and secondaries and the network specification to identify the transport provider to RFS are dependent on the particular transport provider used.

The following sections describe transport provider information that relates to RFS administration: network listener, network specification, and network addresses.

Network Listener

The network listener is part of the Networking Support Utilities package. Essentially, the listener's function is to wait for requests from the network. A call coming in from the network will request a particular service code. The service code will tell the listener to direct the call to a particular process.

Service code **105** is used to request RFS services. If all software installation was done as noted in the *Remote File Sharing Release Notes*, the RFS service code should be automatically configured for the listener of every transport provider you installed. Otherwise, you will need to use the **nlsadmin** command to manually configure the listener. (See the "Setting Up RFS" section of this chapter.)

Network Specification

Since you could have several transport providers on one computer, you must tell RFS which transport provider will handle RFS on your machine. The network specification is the name you will use when you initially configure RFS to indicate its transport provider. (The STARLAN network, for example, uses **starlan** as its network specification. This tells RFS that */dev/net/nav/md000* is the device representing the transport provider to use.)

Network Addresses

When RFS is started on a machine, the machine tries to contact its domain's primary name server. To do that, the machine must know the primary's network address.

The form of the network address varies according to the transport provider used. The STARLAN network convention for network addressing is to use a machine's node name and append the string **.serve** to create the network address. For example, the network address for a machine whose node name is **charlie** would be **charlie.serve** on a STARLAN network.

Security

RFS provides several mechanisms for ensuring the security of your resources. Some of these mechanisms, however, require diligence to set up and maintain. This is especially true if the machines, resources, and users are constantly changing on the network.

As a system administrator, you can maintain strict control of your resources. No files, directories, or devices in an unshared file system can be accessed by other computers. Standard UNIX system file security measures can be used in combination with special RFS facilities to protect your resources.

Direct access to your computer is controlled because local users still have to log in as they always have. As for remote accessibility, you can set up security to allow only certain remote computers to access your resources.

The major mechanisms in RFS for protecting your resources are described in the following sections: "Verify Computers," "Restrict Resources," and "Map IDs."

Verify Computers

When a remote computer tries to mount a resource from your computer and no other resources are mounted, it tries to set up a connection (virtual circuit) across the network to your machine. Once this virtual circuit is set up, the remote machine can mount any resource you have made available to it. This virtual circuit is closed when the last resource is unmounted.

Before this virtual circuit is created, you can verify that the computer is the one it claims to be by checking its RFS password. The following text describes what happens when verification is and is not used.

- No verify: any computer can connect

  If the computer is listed in the *domain/passwd* file, your machine will check its password. Otherwise, your computer will accept it as the machine it claims to be.

- Verify: some computers can connect

  If you use the RFS verification feature, you can make sure that only specific machines can use any of your resources. Those machines must be listed in the proper *domain/passwd* file and must match the password you have for them. (*domain* is the domain name of the requesting

machine.) You can tailor this file if you only want a subset of machines to be allowed to connect. (A description of how to use this feature is contained in the "Setting Up RFS" section of this chapter.)

Restrict Resources

Once a remote computer has established a connection to your computer, the resources it can mount from your machine depend on how you advertised each resource. These are your choices:

- Any machine can mount.

  You may have advertised the resource so that any machine that can connect to your machine can mount it.

- Some machines can mount.

  You restricted access to the resource to certain machines. The remote computer trying to mount it must be one of those machines.

You also may have advertised the resource as read-only. In that case, the remote computer can only mount the resource read-only instead of read/write (default).

Map IDs

Remote users' permissions can be defined to provide another layer of security for a mounted resource. Remote users and groups can be mapped into your computer's user and group list to set permissions they will have to your resources.

You can set these mapping rules on a global or per-machine basis. The global rules set user and group permissions for all remote machines that do not have explicit mapping rules.

Here are the ways you can map remote machines' users into your machine. These rules apply to both global and per-machine mapping.

- No mapping

  If you don't set any special mapping for any remote computer, all users will be mapped into your machine as a "special guest" user ID/group ID. This is the easiest approach because you don't need to keep any records for the remote machine, create rules files, or run the **idload** command.

- Default mapping

  You can set default mapping so that all remote users are mapped into one of these permissions:

  - The local user ID number that matches each remote user's ID (**default transparent**)
  - A single local ID number
  - A single local ID name
  - The local user name that matches each remote user's name (**map all**)

  Group permissions can be mapped in the same way. Users and groups are mapped independently. If there are exceptions to the default mapping, you can **exclude** certain users and groups so they only have special guest permissions (for example, **exclude 0**).

- Specific mapping

  You can map any user or group from any remote machine into a specific user or group on your machine. You can do this by user name or numeric ID.

Using these mapping techniques and standard methods for setting file permissions, you can keep strict controls over your resources, even after they are remotely mounted. (See the "Mapping Remote Users" section of this chapter for more details.)

RFS Features

Some RFS features that reflect improvements over other distributed file systems are described in the following paragraphs.

Compatibility

Once you mount a remote resource on your system, it will look to your users as though it is part of the local system. You will be able to use most standard UNIX system features on the resource. Standard commands and system calls, as well as features like File and Record Locking, work the same on remote resources as they do locally. Applications should be able to work on remote resources without modification.

Flexibility

Since you can mount a remote resource on any directory on your system, you have a lot of freedom to set up your computer's view of the world. You do not have to open up all your files to every machine on the network. Likewise, you do not have to make all files on the network available to your computer's users.

Performance (Client Caching)

The client caching feature of RFS provides substantial performance improvements over non-caching systems by reducing the number of times data must be read across the network. Client refers to the computer that is using a remote resource, while caching refers to the client's ability to store data in local buffer pools.

The first time a client process reads a block of data from a remote resource, it is placed in local buffer pools. Subsequent client processes reading a server file can avoid network access by finding the data already present in local buffers. This generally causes a large reduction in network messages, resulting in improved performance.

For client caching to work simply and reliably, the following features were built into it:

- Cache consistency. Checking mechanisms are used to ensure that the cache buffers accurately reflect the contents of the remote file the user is accessing.

- Transparency. The only difference users should see between caching and non-caching systems is improved response time. RFS-based applications do not have to be changed to run on an RFS system that caches remote data.
- Administration. By default, client caching is on. However, options are available to turn off caching for an entire system or for a particular resource. (You would probably only do this if you have an application that does its own network buffering.) There are also some tunable parameters available to fine tune your system according to the way you use RFS. (See the "Monitoring" and "Parameter Tuning" sections of this chapter for more information.)

Setting Up RFS

In most cases, you will not need the set of tasks described in this section because the basic RFS configuration and reconfiguration can be handled using the commands described earlier in this chapter. These tasks are for those who want to go deeper into the workings of RFS or are having problems with particular components.

These tasks are run from the shell. They should be run initially in the order described.

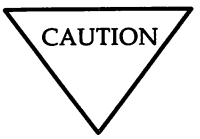
Once these tasks are completed, go to the "Starting/Stopping RFS" section for information on starting RFS.

Prerequisites

Before you begin setting up RFS, the following must be installed and running: UNIX System V Release 3.1 (or later) software, Remote File Sharing Utilities, Networking Support Utilities, and transport provider software. (See the *Remote File Sharing Release Notes* and the transport provider manuals that accompany the product for installation instructions.)

You must also log in as *root*.

Set Node Name



Changing the node name of your computer requires careful coordination with all machines that communicate with yours using Remote File Sharing or other communications packages that rely on node name.

Check to see if your computer's node name is set to the name you want (**uname -n**). If it's not, set it by typing

uname -S *nodename*

You will be asked to type in your computer's node name. A node name that is valid for RFS can consist of up to eight characters of letters (uppercase or lowercase), digits, hyphens (-), and underscores (_). Some networks, such as the STARLAN network, require that every node name in the network be

different. RFS, however, only requires that every node name in a domain be different.

Set Up Network Listener

If you have installed the Networking Support Utilities, the AT&T implementation of the STARLAN network, and RFS in the order described in Chapter 2, "Software Installation," you can skip this task. The listener will already be installed and set up to run automatically, and RFS will be listed as an available service.

If you are using another transport provider or suspect that your STARLAN network listener is set up improperly, this task will show how to manually set up the listener. In the following example, the STARLAN network is used. To set up the listener for other networks compatible with the AT&T Transport Interface, you should replace **starlan** with the name of the network (network specification) you are installing. (For more details, see the *nlsadmin*(1M) manual page in the *User's/System Administrator's Reference Manual.*)

To determine if the listener is properly installed and set up for use by RFS, type the following:

```
nlsadmin -v starlan
```

If service code 105 is listed, then the listener is configured to be used for RFS.

Run the following commands if the listener is not properly set up. If you run any of these commands and they have already been run, you will receive a message telling you so. This will not harm your listener configuration. Type

```
nlsadmin -i starlan
```

to initialize the files needed for the listener process for the network specified, in this case **starlan.**

Next, type

```
nlsadmin -a 105 -c /usr/net/servers/rfs/rfsetup -y "rfsetup" starlan
```

to add the RFS service (**rfsetup**) to the list of services available to the **starlan** listener.

Use the following command line to report the status of the **starlan** listener process installed on this machine (ACTIVE or INACTIVE):

```
nlsadmin -x
```

Next, type

```
nlsadmin -l "nodename.serve" -t "nodename" starlan
```

to register the network addresses of your machine. The listener will listen for requests for these addresses on the network. Only the **-l** address is required by RFS. The **-t** address is used only for terminal services and may not be needed on all networks.

To start the listener, type

```
nlsadmin -S starlan
```

Normally, it will be started automatically when your machine enters multi-user mode (**init 2**).

Set the Domain Name

Set the domain name by typing

dname -D *domain*

where *domain* is replaced by the domain of which your machine will be a member. The domain name must:

- contain no more than 14 characters
- consist of any combination of letters (uppercase or lowercase), digits, hyphens, and underscores
- be different from the name of any other domain used on the network if there is more than one domain on your network

You can check the current domain name by typing:

```
dname
```

Set the Transport Provider

To identify the network, you must tell RFS which network (transport provider) it should use. (In our example, this is **starlan** for the STARLAN network.)

```
dname -N starlan
```

This command indicates the device, relative to the */dev* directory, that is used for the transport provider.

Create rfmaster File

The *rfmaster* file should only be created manually on the primary. If your machine is not the primary, you should skip this task; the *rfmaster* file for your domain will automatically be placed on your machine the first time you start RFS (**rfstart -p** *primary_addr*).

If you are on the primary, you can create an *rfmaster* file in the */usr/nserve* directory using any standard file editor. The contents of this file will define the following:

- the primary name server for your domain
- secondary name servers for your domain
- network addresses for each of these machines

(See the section on "Multiple Domain Name Service" in this chapter for a description of other information you may want to put into the *rfmaster* file.)

Here is an example of an *rfmaster* file for a domain called **peanuts**, whose primary and secondary name servers' node names are **charlie**, **linus**, and **lucy**. Adding each machine's domain name (**peanuts**) to its node name, separated by a period, forms its full RFS machine name. Each line of the example translates as follows:

- For domain **peanuts**, the primary is **peanuts.charlie**.
- For domain **peanuts**, a secondary is **peanuts.linus**.
- For domain **peanuts**, another secondary is **peanuts.lucy**.

- For computer **peanuts.charlie**, the network address is **charlie.serve**.
- For computer **peanuts.linus**, the network address is **linus.serve**.
- For computer **peanuts.lucy**, the network address is **lucy.serve**.

(The addresses shown are an example of STARLAN Network addresses. These addresses should be in the form nodename.serve.)

```
peanuts          p          peanuts.charlie
peanuts          s          peanuts.linus
peanuts          s          peanuts.lucy
peanuts.charlie  a          charlie.serve
peanuts.linus    a          linus.serve
peanuts.lucy     a          lucy.serve
```

Each line in the example is an entry. The second field is the *Type* field, which indicates whether the entry defines a primary name server (`p`), secondary name server (`s`), or the network address (`a`) for one of these name servers. Here is the information needed for the first field, *Name*, and the third field, *Rdata*, for each type of entry:

`p` Primary entry. *Name* is the domain name. *Rdata* is the full RFS machine name of the domain's primary name server (*domain.nodename*).

`s` Secondary entry. *Name* is the domain name. *Rdata* is the full RFS machine name of the domain's secondary name server (*domain.nodename*).

`a` Address entry. *Name* is the full RFS machine name (*domain.nodename*) of a name server computer. *Rdata* is the network address of the computer. The manuals that come with your network should describe how to find a computer's network address.

Here are some special considerations when creating the file.

- Fields in each entry must be separated by one blank or one tab.
- An entry can extend beyond one line if you enter a backslash, then a carriage return to continue to the second line.

- This file should be write-protected from all but *root*, but all read permissions should be enabled (644 permissions).
- If you start a line with the # character in column 1, the entire line will be treated as a comment.

Add/Delete Domain Members

If your computer is the current primary name server for the domain, you must add each computer to the domain member list. (If a secondary has temporarily taken over, the secondary must pass name server responsibility back to the primary using the **rfadmin -p** command.) To add members, use the following command:

```
# rfadmin -a domain.nodename
Enter password for nodename:
Re-enter password for nodename:
```

where *nodename* is replaced by the node name of the computer you want to add to your *domain*. (The two names must be connected by a period.)

You will be prompted for an initial password, which will be stored in the */usr/nserve/auth.info/domain/passwd* file for the *domain*. When the computer you added starts RFS, the computer's administrator must enter this password. You can simply type a **<CR>** for a null password. Otherwise, the password must conform to the same criteria used with the **passwd** command. Repeat this command for each computer you want to add to the domain.

NOTE Adding a primary and secondary to the *rfmaster* file does not automatically add them to the domain. You must do this procedure for each of those machines.

You can also use the **rfadmin** command to delete members from the domain member list as follows:

rfadmin -r *domain.nodename*

Remote Computer Verification

NOTE

This procedure assumes you are starting RFS from the shell using **rfstart** with the **v** option or **init 3**.

When you start RFS, you can indicate that all remote machine passwords be verified when they try to use your computer's resources. **rfstart** is the command that is run automatically when you go into RFS state (**init 3**).

If you use **rfstart** with the **-v** option, any machine that tries to mount your resources must match a name and password you have in the *passwd* file in the */usr/nserve/auth.info/domain* directory on your machine, where *domain* is replaced by the name of the remote computer's domain. If the remote computer is not listed in this *domain/passwd* file, if it is listed and the password doesn't match, or if no *domain/passwd* file exists, the remote mount will fail. (This file is automatically on the primary, but it must be added to other machines, as described in this procedure, to use verification.)

If you do not use the **-v** option, the following validation will occur. If a *domain/passwd* exists for the remote computer's domain on your computer and the remote computer is listed, but the password does not match, a mount request will fail. If the computer is not listed in the file or if the *domain/passwd* file does not exist, the computer will be allowed to mount your resources without validation. (Of course, a remote mount could still fail if the resource was advertised to a limited subset of machines or was advertised read-only and the machine tried to mount it read/write.)

The following steps describe how verification is set up:

Step 1: Obtain *domain/passwd* file(s). The */usr/nserve/auth.info/domain* directory on the primary will contain a file called *passwd*. (*domain* is replaced by the domain name.) This file will have the name and encrypted password for each machine in the domain.

You must make the *domain/passwd* file plus the *domain/passwd* file for any outside domains containing machines you want to verify accessible to your machine in one of the following ways:

Step 1A: Place a copy of this file(s) in the same directory on your machine. The *passwd* file for each domain must be in the appropriate *domain* subdirectory.

or

Step 1B: Have the primary for each domain advertise the */usr/nserve/auth.info/domain* directory; then have it automatically mounted in the same location on your computer. This way you can automatically pick up any changes in machines or passwords. (See the description of */etc/fstab* in the "Automatic Remote Mounts" section of this chapter for information on setting up automatic mounts.)

Step 2: **rfstart -v**. You must edit the */etc/rc3.d/S21rfs* file to automatically run **rfstart** with the **-v** option. You will add the **-v** to about line 61 of this file, after the **rfstart** command, as shown in the following example:

```
'rfstart')
  trap 'rm -f /usr/tmp/rfs$$;exit' 0 1 2 3 15
  stat=1
  retries=0
  while [ ${stat} -eq 1 ]
  do
    /usr/bin/rfstart -v </dev/console >/dev/console 2>/usr/tmp/rfs$$
    stat=$?
    case ${stat} in
```

Step 3: If you want to verify only a limited subset of these computers, you must use manually edited versions of the *domain/passwd* files, removing any computers you want to prevent using your resources. (You cannot edit this file if you are a primary or secondary name server or if you have mounted the file from the primary.)

Resource Sharing With Other Domains

For computers in your domain to share resources with computers in other domains on your network, you must do the following:

Step 1: Find out

- the primary name server for each domain
- the secondary name server(s) for each domain
- the network address for each of the above name servers

Step 2: You must see that the information in Step 1 is added to your domain's */usr/nserve/rfmaster* file on the primary. See the description of the *rfmaster*(4) file in the format of the *rfmaster* file. The following example shows the information added to contact a domain called **docs**.

```
docs           p           docs.big
docs           s           docs.little
docs.big       a           big.serve
docs.little    a           little.serve
```

Step 3: Stop RFS on the primary (**rfstop** or **init 2**).

Step 4: Restart RFS on the primary (**rfstart** or **init 3**). Make sure start-up has completed before going to the next step.

Step 5: If a secondary machine took over name service when the primary was stopped, pass name service responsibilities back to the primary by typing the following from the secondary:

```
rfadmin -p
```

Step 6: Mount resources from an outside domain. Once the name server machines have picked up the new domain names, you can mount a resource from a remote domain on your own machine. You would use the same method of mounting a resource from an outside domain as you would to mount a resource from your domain, with one exception. When you specify the resource to be mounted, you must prefix the domain name to the resource identifier. For example, the command

```
mount -d docs.INFO /usr/info
```

could be used to mount a resource called INFO that is advertised in domain **docs** with read/write permissions.

Multiple Domain Name Service

Once you have defined a set of primary and secondary name servers to serve a domain, that set of machines may also be name servers for another domain on the same network. The following procedure describes how this can be configured:

Step 1: Edit *rfmaster* file. You must add the information on the new domain's name servers to the *rfmaster* file on the primary. The following is an example of two sets of name servers that serve domains called **docs** and **peanuts**:

```
docs          p          docs.big
docs          s          docs.little
docs.big      a          big.serve
docs.little   a          little.serve
peanuts       p          docs.big
peanuts       s          docs.little
```

Step 2: Stop and restart RFS. You must stop all machines served by the primary (**rfstop** or **init 2**). You must then restart the primary (**rfstart** or **init 3**). Then start each machine on the system, starting machines that previously had other machines as domain name servers with the **rfstart -p** *address*, where *address* is replaced by the network address of the new primary domain name server. This will ensure that the new information is picked up by each machine.

Complex User ID/Group ID Mapping

ID mapping lets you control the access remote users will have to files and directories that make up your shared resources. This feature lets you assign each remote user the permissions of one of your local users (listed in */etc/passwd*) or the permissions of a special "guest ID," with respect to your shared resources. The "guest ID" will never overlap with any of your local users. The same mechanism can be used to define group permissions (listed in */etc/group*).

Use this procedure as a tutorial for ID mapping and as a procedure for setting up mapping. If you have questions about particular mapping components, refer to the "Mapping Remote Users" section of this chapter.

When Not to Map

In most cases, ID mapping is not necessary. If you never set up mapping, all users will be mapped into a single special guest ID. This special guest ID is represented by an ID number that is one higher than the maximum allowed for your system. By default, the maximum number of users and groups on a system is 60000, so the special guest is ID number 60001.

No mapping, or the default mapping, provides the maximum security for your shared resources. When a remote user lists the permissions of your files (**ls -l**), all files will be owned by 60001 or 60002. The 60001 means the file was created by a remote user and, therefore, is owned by every remote user that can access your resource. 60002 means the file was created by one of your local users, and, therefore, the remote users can only access the file if the "other" permissions are set (the last 3 bits of the **rwx** permissions).

When to Map

Using mapping increases the power and flexibility of RFS. The following are some reasons you may want to use mapping:

- Special permissions.

  You may want to map some or all remote users into particular local users' permissions. For example, if you are the administrator of several machines, you may want to map all *root* logins together across the machines. Therefore, you would be able to modify any remote resources mounted on any machine you are working from.

- Transparent mapping.

  If you set up a group of computers to have the same */etc/passwd* and */etc/group* files, mapping transparently can be a very powerful technique. When a user creates a file, the user will maintain sole ownership of the file, whether or not the file resides on a remote resource.

  With transparent mapping, you could share many resources that require a consistent view of user ownership. For example, you could share your */usr/mail* directory, mount it on */usr/mail* on other computers, and have one mail directory for the entire set of machines. The basic concept is that you can avoid duplication of many files and directories while maintaining consistent user permissions.

- Mapping by machine.

  You may want to map users from one machine differently than users from another machine. For example, you may want to map all users from one machine into user ID **600**, from another machine into **700**, and from a third into **800**. In that way you could monitor which remote machine's users were creating files within your resources.

Mapping Tools and Files

The result of this procedure is "mapping translation tables." These tables will be used by your system to process requests from remote users for access to your resources that are mounted on their computers.

The command used to create the translation tables is **idload**. When **idload** is run with no options, it does the following:

- reads the rules files to determine how you want to set up the mapping
- reads the *passwd* and *group* files on your computer and copies of those files from other computers, if needed
- creates translation tables

There are two options to **idload** you also may want to use when setting up translation tables:

idload -n Before you run **idload** with no options, the **-n** option lets you do a trial run without actually changing the mapping tables. The result is a listing at your terminal of the tables you would create if you ran **idload** with no options.

idload -k After you run **idload** with no options, the **-k** option lets you read the mapping that is currently in effect on your computer.

Figure 9-2 illustrates the components described in the previous paragraphs.

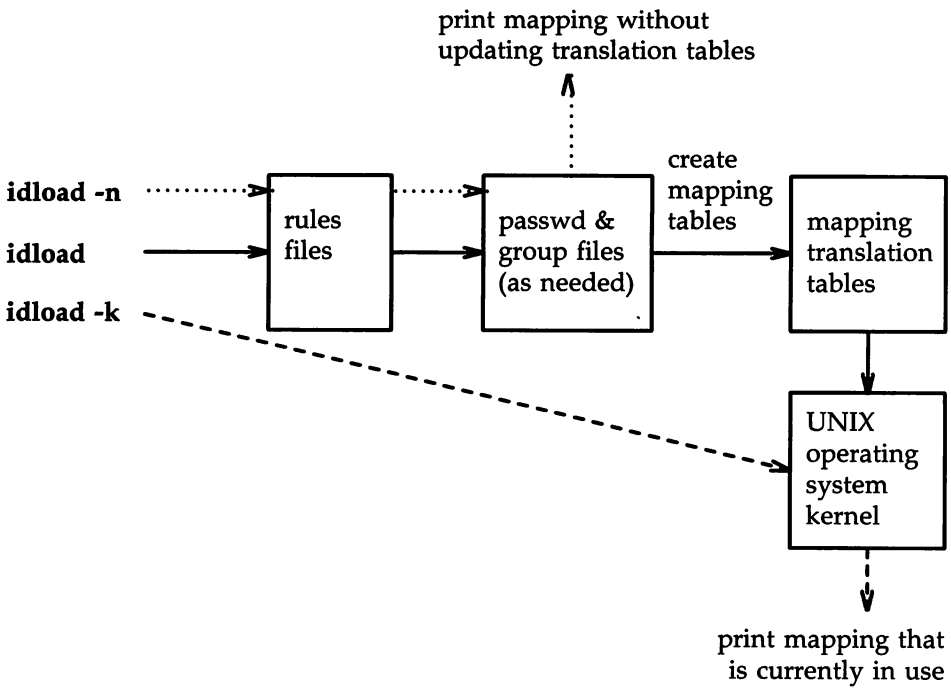


Figure 9-2: ID Mapping Components

Figure 9-3 illustrates the files that are involved in setting up ID mapping.

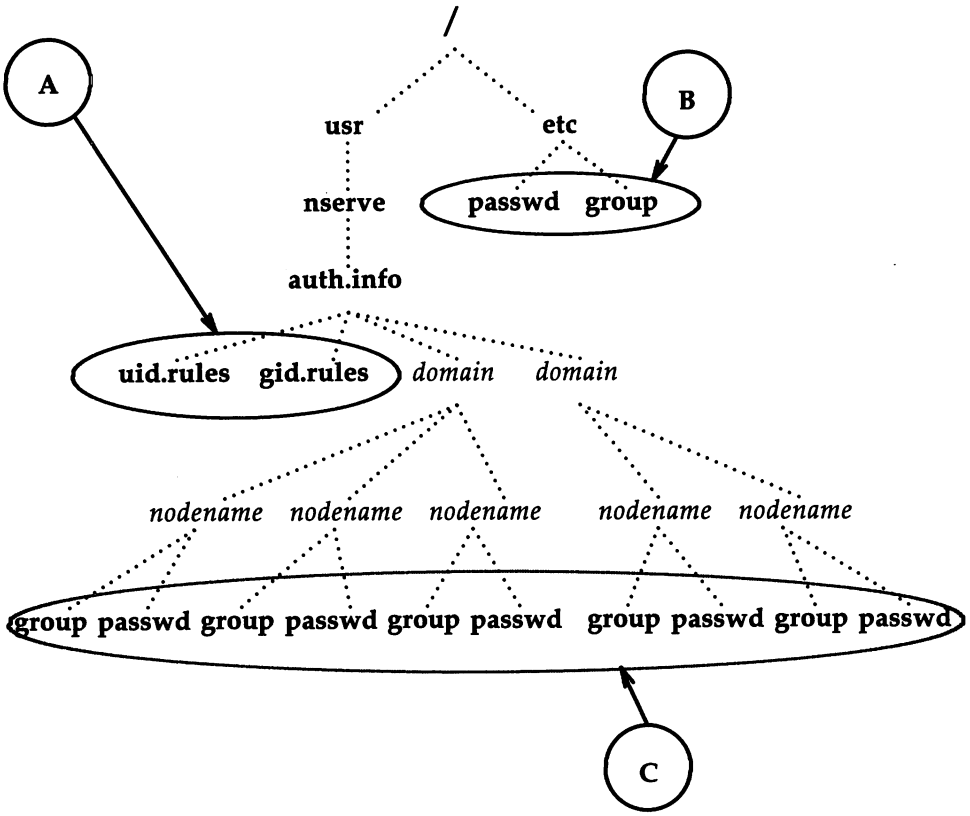


Figure 9-3: ID Mapping Files

The files used for ID mapping are divided into the following three groups, as shown in Figure 9-3:

A. Rules Files
The *uid.rules* and *gid.rules* files are located in the */usr/nserve/auth.info* directory. The information you add to these files tells the **idload** command how to create the mapping tables.

B. Local *passwd* and *group* Files
The */etc/passwd* and */etc/group* files contain lists of the local users on your system. Though you don't modify these files to do ID mapping, you will be interested in the information that is in these files. The first field in each line of your *passwd* and *group* files contains local user and group names, respectively. The third field contains the related ID number. If you map by local name in the rules files, these files are read to translate the names into numbers.

C. Remote *passwd* and *group* Files
Because mapping translation tables are sets of numbers, if you want to map a remote user by name, you must have a copy of the *passwd* and/or *group* files for the remote user's machine. These files should be placed in the */usr/nserve/auth.info/domain/nodename* directories, where *domain* and *nodename* are replaced by the remote computer's domain and node names, respectively.

Step 1: Create uid.rules File

The following steps describe how to create the rules used to map remote users.

Using any standard file editor (**ed** or **vi**, for example), create or edit the *uid.rules* file in the */usr/nserve/auth.info* directory. Steps 1A-1D will help you set up a **global** block of mapping information; Steps 1E-1H are for **host** blocks of mapping information. The **global** block defines the permissions that will apply to the users on all computers that do not have specific mapping. Note that all lines within a **global** block are optional.

Step 1A: Add the **global** line. (Add this line only if you want to define a block of global information.) The global block of information must begin with the following keyword on a line by itself:

global

Step 1B: Add a **default** line. (Add this line only if you want to define default information for a global block.) Following the **global** line, you can choose the default permissions that will apply to users from all machines that are not specifically mapped. If this line is not used, the system assumes **default 60001**. (In most cases, **default 60001** is fine.) The two types of default lines are illustrated below.

The line **default transparent** means that each user will have the permissions of the user with the same ID number on your system. (This strategy is most valuable when the */etc/passwd* files are identical on the two machines.) In the line **default** *local*, the word *local* can be replaced by a local ID number or ID name. This means that any users that are not specifically mapped will have the permissions of a particular user on your system. (Use only one **default** line in a **global** block.)

default transparent
or
default *local*

Step 1C: Add **exclude** line(s). (Add this line(s) only if you want to exclude certain users.) **exclude** lines let you exclude certain users from having the permissions defined in the default line. For example, if you used **default transparent**, you may want to use **exclude 0** to make sure that the *root* user doesn't have permission to modify the restricted files owned by *root* in your resources. The two types of exclude lines are illustrated below.

In **exclude** *remoteid, remoteid* is replaced by a remote user ID number. The remote user would then have the permissions of the guest user (UID 60001) to your resources.

The **exclude** *remoteid-remoteid* line lets you specify a range of remote IDs to exclude. For example, **exclude 0-100** could be used to exclude all administrative logins from your default mapping.

exclude *remoteid*
or
exclude *remoteid-remoteid*

Step 1D: Add **map** line(s). (Add this line only if you want to map specific users from global machines.) **map** lines let you take specific remote user IDs and map them into the permissions of one of your local users. The two types of map lines are illustrated below.

In **map** *remoteid:local, remoteid* is replaced by a remote user ID number, and *local* is replaced by a local user's name or ID number. For example, the line **map 20:root** would map the remote user with ID number 20 into your machine's *root* permissions (UID **0**). The line **map** *remoteid* says give the remote user the permissions of the user with the same ID number on the local system. For example, **map 0** would give *root* from a remote machine the same permissions as *root* on your machine.

map *remoteid:local*
or
map *remoteid*

Once **global** mapping is done, you may want to add **host** mapping information to the *uid.rules* file. A **host** block defines the permissions that will apply to the users on particular remote machines. You can have one **host** block for each remote machine you want to map specifically. Note that all lines within a **host** block are optional.

Step 1E: Add a **host** line. (Add this line only if you want to define a block of host information.) The host block of information must begin with the following keyword on a line by itself:

host *domain.nodename*

where *domain* is replaced by the remote machine's domain name and *nodename* is replaced by the machine's node name.

Step 1F: Add a **default** line. (Add this line only if you want to define default information for a host block.) Following the **host** line, you can choose the default permissions that will apply to all users on the remote machine that are not specifically mapped or excluded. If this line is not used, the system assumes **default 60001**. (In most cases, **default 60001** is fine.) The two types of default lines are illustrated below.

The line **default transparent** means that each user will have the permissions of the user with the same ID number on your system. (This strategy is most valuable when the */etc/passwd* files are identical on the two machines.) In the line **default** *local*, the word *local* can be replaced by a local ID number or ID name. This means that any users that are not specifically mapped will have the permissions of a particular user on your system.

default transparent
or
default *local*

Step 1G: Add **exclude** line(s). (Add this line(s) only if you want to exclude certain users from default permissions.) **exclude** lines let you exclude certain users from having the permissions defined in the default line. For example, if you used **default transparent**, you may want to use **exclude 0** to make sure that the *root* user doesn't have permission to modify the restricted files owned by *root* in your resources. The two types of default lines are illustrated below.

In **exclude** *remote*, *remote* is replaced by a remote user name or UID number. The *remote* user would then have the permissions of the guest user (UID 60001) to your resources.

The **exclude** *remoteid-remoteid* line lets you specify a range of remote IDs to exclude. For example, **exclude 0-100** could be used to exclude all administrative logins from your default mapping.

exclude *remote*
or
exclude *remoteid-remoteid*

Step 1H: Add **map** line(s). (Add this line(s) only if you want to map particular users.) **map** lines let you map specific remote users from specific remote machines into the permissions of one of your local users. The two types of map lines are illustrated below.

The **map all** line says to map all user names into the permissions of the users with the same names on your system. In **map** *remote:local, remote* is replaced by a remote user ID name or number and *local* is replaced by a local user's name or ID number. For example, the line **map 20:root** would map the remote user with ID number 20 into your machine's *root* permissions (UID **0**). The line **map** *remoteid* says give the remote user the permissions of the user with the same ID number on the local system. For example, **map 0** would give *root* from a remote machine the same permissions as *root* on your machine.

map all
or
map *remote:local*
or
map *remote*

Repeat steps 1E-1H for each specific computer whose users you want to map.

Figure 9-4 is an example of what your rules file may look like.

```
global
default 1000
exclude 0

host peanuts.snoopy
default transparent
exclude 0

host peanuts.linus
default 60001
map 0:100
```

Figure 9-4: Example *uid.rules* File

Step 2: Create gid.rules File

The following steps describe how to create the rules used to map remote groups.

Create the *gid.rules* file. Using any standard file editor (**ed** or **vi** for example), edit the *gid.rules* file in the */usr/nserve/auth.info* directory. The *gid.rules* file follows the same format as the *uid.rules* file. Therefore, you can use Steps 1A through 1H to set up the *gid.rules* file, replacing any references to users with references to groups.

NOTE

If you create a *uid.rules* file, you should also create a *gid.rules* file. Although **idload** will still work without the *gid.rules* file (**idload** will use defaults for mapping groups), a warning message will be produced.

Step 3: Add passwd and group Files

If, when you edited the *uid.rules* and *gid.rules* files, you referenced any remote users by name, you must have copies of the *passwd* file from the remote users' computers in the */usr/nserve/auth.info/domain/nodename* directories on your machine. The same is true of the *group* file for groups referenced by name. (Note that **map all** maps by name.)

The best way to obtain these files is as follows:

Step 3A: Obtain files. Have each machine whose users you want to map by name send you its */etc/passwd* and */etc/group* files using any standard file transfer method (such as **uucp**). (The information in the password field can be removed from each entry, if you prefer. The password is made up of the characters between the first and second colon in each entry.)

Step 3B: Create directories. You must create a separate directory on your machine for each computer whose users and groups you map by name. Each directory must be created using the path */usr/nserve/auth.info/domain/nodename*, where *domain* is replaced by the remote machine's domain name and *nodename* is replaced by the remote machine's node name. For example, you create the following directory for a machine called **linus** in domain **peanuts**:

/usr/nserve/auth.info/peanuts/linus

Step 3C: Place files. Place the remote machines' *passwd* and *group* files in the directory you created in the previous step.

Step 4: Run idload

Step 4A: Run **idload -n**. This command will print a listing of the mapping rules you set up without creating translation tables. Figure 9-5 is the output from **idload -n** using the *uid.rules* file shown after Step 1H and a *gid.rules* file with simply **default 60001** in the global block.

```
TYPE  MACHINE          REM_ID     REM_NAME   LOC_ID          LOC_NAME

USR   GLOBAL           DEFAULT    n/a        1000            n/a
USR   GLOBAL           0          n/a        60001           guest_id
USR   peanuts.snoopy   DEFAULT    n/a        transparent     n/a
USR   peanuts.snoopy   0          n/a        60001           guest_id
USR   peanuts.linus    DEFAULT    n/a        60001           n/a
USR   peanuts.linus    0          n/a        100             n/a

GRP   GLOBAL           DEFAULT    n/a        60001           n/a
```

Figure 9-5: Example Output From **idload -n**

Step 4B: Run **idload**. If the output from **idload -n** was acceptable, type the **idload** command with no options to create the translation tables. The **global** rules and **host** rules for any computer that currently has your resources mounted will immediately take effect. Rules for any other computer that you mapped will take effect as soon as that computer mounts one of your resources.

Step 4C: Run **idload -k**. This will print the mapping that is currently in use on your computer. (Remember that rules for any other computer that you mapped will not be in effect until that computer mounts one of your resources.)

Once mapping is set up, it can be changed whenever you like. You can edit rules files and run **idload** again at any time. It doesn't matter if resources are mounted or even if RFS is running.

Starting/Stopping RFS

Before a non-primary machine can start RFS, RFS must be configured on the machine, and the primary must be up and running RFS.

Is RFS Running?

If you are not sure if RFS is running, type **rfadmin -q**. This will tell you whether RFS is running.

Another way is to check that processes related to RFS are active. To do this, type **ps -e**. These processes should be active:

listen
rfdaemon
nserve
rfudaemon
recovery
server (*optional*)

There may be multiple processes of some of these names running.

Initial RFS Start

The first time you start RFS on a non-primary machine, you should use the following command:

rfstart -p "*nodename.serve*"
`rfstart: Please enter machine password:`

where *nodename.serve* is the address for the primary name server for this domain.

RFS Password

You will be prompted for a password the first time you start RFS. The password must match the password entered when your machine was added to the domain member list in the primary name server (the **rfadmin -a** command). If password verification succeeds, your computer will save this password automatically so you do not have to enter it again.

Likewise, your machine will save the network address of the primary name server. Therefore, the next time you start up RFS, you will be able to do it via **init 3**.

RFS Password Mismatches

Any time you start RFS (**rfstart**) and your password does not match the one on the current domain name server, you will receive a warning, but **rfstart** will not fail.

Though RFS will be active, you may have a problem if the *domain/passwd* file from the primary domain name server is shared with other machines to use for verification. In that case, your remote **mount** requests will fail if the passwords don't match. For this reason, it is recommended that RFS passwords always be kept up-to-date on each computer and the primary name server. If passwords aren't important to you, you can simply enter a carriage return for the passwords on each computer and the primary.

If you do get warnings that your password is out of sync with the current domain name server and you want to fix it, you should handle it differently if the primary is the current domain name server than if the secondary has temporarily taken over.

First, find out which machine is the current name server and whether it is the primary or secondary by doing the following:

```
# rfadmin
the acting name server for domain domain is domain.nodename
# cat /usr/nserve/rfmaster
domain   P   domain.nodename
domain   S   domain.nodename
domain.nodename  A  network_address
domain.nodename  A  network_address
```

Then, depending on which machine is the current name server, do one of the following:

- Secondary is the current name server

  If the primary went down and a secondary took over as domain name server, the secondary may not have a *domain/passwd* file or may have one that is out-of-date. In this case, do not try to correct your password until the primary takes over as domain name server again.

- Primary is the current name server

  Try to correct your password by re-entering it with the **rfpasswd** command. If that does not work, follow the sequence shown below, replacing *domain.nodename* with your computer's RFS machine name.

  From the primary name server:

```
# rfadmin -r domain.nodename
# rfadmin -a domain.nodename
Enter password for nodename: type password
```

From your computer:

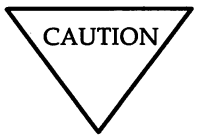
```
# rfstop
# rm /usr/nserve/loc.passwd
# rfstart
rfstart: Please enter machine password: type password
```

You should then make sure that any computer that verifies your computer's password copies the new *domain/passwd* file from the primary.

Changing RFS Password

If you want to change your RFS password later, you must use the **rfpasswd** command. This will change your RFS password, both on your computer and on the primary domain name server. Processing of the new password follows the same criteria as *passwd*(1) in the *User's/System Administrator's Reference Manual.*

Since changing passwords requires communication with the primary domain name server, RFS must be running on both your computer and the primary domain name server. You cannot change your RFS password if the primary is down and a secondary is the current domain name server.



When you change your password, computers that are authenticating your computer may not automatically receive the change. If you are unable to mount a resource from a remote machine after you change your password, check that the remote machine has copied the latest version of your domain's passwd file from your primary domain name server.

Automatic RFS Startup (init 3)

There are several steps involved in starting up RFS and sharing resources. To simplify this procedure, a new RFS run level has been defined: run level 3.

When you enter run level 3 using the **init 3** command, RFS is automatically started via */etc/rc3* from shell scripts in your computer's */etc/rc3.d* directory. These scripts start RFS, advertise local resources, and mount remote resources. When you leave run level 3 (using **shutdown** or **init 2**, for example), RFS processes will be stopped.

You can add your own shell scripts to those that start run level 3. You can also tailor the run level 3 shell scripts to suit the way you use RFS.

This section will describe those shell scripts used in run level 3 and suggest how to modify or add to them.

NOTE Before you can enter RFS mode, you must have already installed and configured RFS.

Entering Run Level 3

You can go into **init** level 3 in one of three ways:

1. From single-user mode (run level **s**)

   RFS mode is also a multi-user mode. Therefore, when you type **init 3** from single-user mode, all multi-user processes (**getty**, **cron**, etc.) will be started followed by RFS mode processes.

2. From multi-user mode (run level 2)

   When **init 3** is run from run level 2, init checks that all multi-user processes are running, then starts the Remote File Sharing mode processes. (**init 3** will not spawn another process for a level 2 script that is already running.)

3. At boot time

   By default, your system will enter run level 2 at boot time. You can change that to have run level 3 start automatically at boot time by changing the value for **initdefault** in the */etc/inittab* file so it reads as follows:

```
is:3:initdefault:
```

init 3 Processing

When **init 3** is run, all entries in the *inittab* file that indicate level 3 are started, including */etc/rc3*. */etc/rc3* executes all shell scripts in */etc/rc3.d* that begin with *S*.

RFS places only one file in */etc/rc3.d*: *S21rfs*. This file is linked to the *rfs* file in */etc/init.d*. Also, the *rfs* file is linked to *K50rfs* in */etc/rc2.d* and *K65rfs* in */etc/rc0.d*.

/etc/rc3 executes *S21rfs* with the **start** option upon entering run level 3. *S21rfs* then does the following:

- Validates that the domain name has been defined for your machine.
- Validates that the *rfmaster* file has been created. (This may have been created automatically the first time you ran **rfstart -p** if your machine is not the primary. The latest copy is then sent to your machine from the primary domain name server.)
- Executes the **rfstart** command continuously, with 60-second sleep intervals, until it succeeds or returns a fatal error.
- Executes **/etc/init.d/adv** to advertise all system resources you set up in your */etc/rstab* file. (The */etc/rstab* file contains an entire **adv** command line for each advertised resource.)
- Executes **/etc/rmountall** to mount all remote resources you listed in your */etc/fstab* file. Any remote mount that does not succeed will be tried continuously until it does via **/etc/rmount**. (See "Automatic Remote Mounts" in this chapter for the format of */etc/fstab*.)

When you leave run level 3 via **init 1** or **2**, **/etc/init.d/rfs** is executed with the **stop** option. This will execute **rfstop**.

NOTE

If for some reason RFS fails to terminate the **rfudaemon**, RFS may continue to run in the lower run state. You can always bring down RFS by running **unadv** and **fumount** for each advertised resource, **umount** for each mounted remote resource, and **rfstop**.

Changing init 3 Processing

Going into **init 3** makes some assumptions about how you use your RFS system. Here is a description of how to change some of the processing that takes place.

- Retry **rfstart**

  By default, **init 3** will keep trying to start RFS (**rfstart**) until it succeeds. If you want it to try a limited number of times, you must edit the */etc/rc3.d/S21rfs* file. Find the line "RETRIES=0" and change the number "0" (try forever) to the number of times you want it to retry.

- Retry mounts

  When you enter **init 3**, the system tries separately, every 60 seconds, to mount each resource listed in */etc/fstab* until it succeeds or you leave state 3. To change this behavior you can edit */etc/rmount*. Find the line "RETRIES=0" and change "0" (try forever) to the number of times you want to attempt to mount each resource. Find the line "TIME=60" and change "60" to the number of seconds you want it to wait between retries.

Adding RFS Mode Scripts

All files in */etc/rc3.d* and other */etc/rc?.d* directories are shell scripts, so you can read them to see what they do. You can modify the existing files, though it is preferable to add your own since the delivered scripts may change in future releases. To create your own scripts you should follow these rules:

- Place the file in */etc/init.d*.
- Link the file to files in appropriate run level directories using the naming convention described below.
- Have the file accept the **start** and/or **stop** options.

Name the files using the following conventions:

S*00name*
or
K*00name*

The file names can be split into three parts:

S or **K**	The first letter of each file defines whether the process should be started (**S**) or killed (**K**) upon entering the new run level.
00	The next two characters represent a number from 00 to 99. These numbers indicate the order in which the files will be started (S00, S01, S02, etc.) or stopped (K00, K01, K02, etc).
name	The rest of the file name is the */etc/init.d* file name to which this file is linked.

For example, the *init.d* file *rfs* is linked to the */etc/rc3.d* file *S21rfs* and *rc2.d* file *K50rfs*. When you enter **init 2**, this file is executed with the **start** option: **sh S68starlan start**. When you enter **init 0**, this file is executed with the **stop** option: **sh K67starlan stop**.

Stopping RFS

If you started RFS using **init 3**, you can stop it by going to a lower run state (**init 2, init S,** or **shutdown**). If you started RFS using **rfstart**, you can stop it by typing **rfstop**.

Before you can use **rfstop**, you must:

- Unadvertise all your resources (**unadv**).
- Unmount everything you have mounted from remote machines (**umount**).
- Make sure all your advertised resources are unmounted from remote machines (can be forced by using **fumount**).

These steps will happen automatically when you leave **init 3**.

If you are the primary name server, you should not stop RFS unless a secondary is up and ready to take over. If the primary goes down and no secondary is available to take over, computers in the domain that are not the primary or a secondary will be able to start RFS. Computers that are already running RFS will continue to run RFS; however, they will not be able to mount or advertise new resources.

Sharing Resources

This section describes how to share your local resources with other computers (advertising) on an RFS system and how to use the resources other machines have made available (mounting).

Local Resource Advertising

The **adv** command is used to advertise a local directory (one that physically resides on your machine) so that it is accessible to other machines. When you advertise a directory, you must assign it a resource name. This resource must have a unique name within your domain.

When **adv** is performed with the options listed below, the resource is registered with your domain name server. Any computer that has access to your domain can find a listing of your resource from your domain's advertise table (**nsquery** command). A remote computer will not know the exact location of the resource on your machine. All the remote computer will know is its resource name, the short description you assign, that it resides on your computer, and the read/write permissions.

You can set up your system so that resources are advertised automatically when you enter **init 3**. To do this, place the entire command line for each advertised resource in the */etc/rstab* file.

The syntax of the **adv** command to advertise a resource is

adv [**-r**] [**-d** "*description*"] *resource pathname* [*clients...*]

The syntax of **adv** to modify an advertised resource entry is one of the following:

adv -m *resource* **-d** "*description*" [*clients ...*]

adv -m *resource* [**-d** "*description*"] *clients ...*

The options are as follows:

-r The **-r** option, for read-only, is used to advertise the resource with read-only access. If it is not used, read/write access is assumed.

-d The **-d** option indicates that the next argument (*description*) is a description of the resource. The description can be from 0 to 32 characters and must be in quotes.

resource This is the resource name you assign. The name is limited to 14 printable ASCII characters; slashes (/), periods (.), spaces or tabs may not be used. (If you enter more than 14 characters, the name will be accepted and truncated.)

pathname This is the full pathname to the directory you want to share. The directory must be on your local system, and it cannot be already advertised.

clients... This is an optional list of one or more remote machines or domain names to which you want to restrict this resource. (A domain name must have a period (.) appended to it.) If clients are not included, the resource will be accessible to any RFS computer that can connect with your computer. You can also define aliases so a single name can represent a group of computers and/or domains. (See "Aliases" on the following page.)

-m *resource* This option is used to modify the *description* or *client* fields for an advertised resource listing. It cannot be used to change the read/write permissions of a resource.

Below are two examples of **adv** command lines:

```
adv -r -d "Department news" DNEWS /usr/news peanuts.
adv -d "My devices" MDEV /dev lucy linus doc.comp1
```

The first example advertises your */usr/news* directory with read-only permissions under the resource name **DNEWS** to all computers in the **peanuts** domain. The second advertises your */dev* directory as **MDEV** to computers **lucy** and **linus** in your domain and **comp1** in domain **doc**.

Automatic Advertising

You can set up all your **adv** commands to start automatically when the system enters the RFS state (**init 3**). Do this by placing each full **adv** command line in the */etc/rstab* file. As soon as **init 3** successfully starts RFS, all **adv** commands in */etc/rstab* will be run.

The following is an example of an */etc/rstab* file to automatically advertise the two resources shown previously:

```
# cat /etc/rstab
adv -r -d "Department news" DNEWS /usr/news peanuts.
adv -d "My devices" MDEV /dev lucy linus comp1.doc
#
```

Aliases

The **adv** command reads the */etc/host.alias* file to find the definitions of any aliases in the *clients* field. The format of the file is

alias name client1 client2 client3 ...

where *name* is replaced by the character string you want to represent the list of clients. Each client can be a machine name, domain name, or an alias name previously defined in the file. The three fields must be separated by blanks or tabs. If you have too many *clients* to fit on one line, you can extend an entry beyond one line by entering a backslash and then a carriage return.

Resource Security

These are the levels of security that protect your resource once you have advertised it:

verify computers — Only those remote computers that pass your security checks can even connect to your machine. You may have indicated that only those machines you have a record of can connect (see **rfstart -v**).

restrict resource	You may have advertised your resource so that only selected remote computers can mount it (see the *client* option of the **adv** command).
map IDs	The permissions remote users will have to your resources are set on a computer-by-computer basis. In other words, the user and group mappings you set up for a remote computer will apply to any of your resources that computer mounts.
UNIX System security	Normal UNIX System access security, governing read, write, and execute permissions, will apply to any advertised resource.

Local Advertise Table

All advertised resources for your computer are contained in your computer's local advertise table. Any user can use the **adv** command with no options to display the local advertise table. The output will be a listing like the one that follows:

```
# adv
CUSTOMER /usr/bin/cust read-only  "Atlanta customers"    lucy linus doc.tick
SCCS     /sccs         read/write "Project Y source"     unrestricted
CALENDAR /usr/bin/cal  read-only  "UNIX System calendar" peanuts. compgrp
```

The information will match what you entered using the **adv** command with appropriate options for each resource. Some of the information shown was implied when the resource was advertised. For example, access is read/write if **-r** is not specified and clients are not limited to certain machines (**unrestricted**) when no clients are identified. Clients listed in the last field can be

- computer names in your domain (**lucy**)
- computer names in other domains (**doc.comp1**)

- domain names (**peanuts.**)
- aliases listed in */etc/host.alias* (**compgrp**)
- **unrestricted** if the resource is not restricted to certain machines

Domain Advertise Table

All advertised resources for your domain are in the domain advertise table on your domain name server. The **nsquery** command is available to all users to list any or all of the advertised resources in a domain. The syntax of the command is

nsquery [**-h**] [*name*]

where the **-h** option can be used to suppress printing of the heading line, and the *name* option can be replaced by one of the following:

nodename	Lists the resources advertised by a particular computer in your domain.
domain.	Lists all resources advertised by all machines in a domain. (A period at the end of a name causes it to be interpreted as a domain name.)
domain.nodename	Lists all resources advertised by a particular computer in a domain outside your own domain.

If the *name* option is not used, **nsquery** will print a list of all advertised resources in your domain. Here is an example of output from an **nsquery** command:

```
# nsquery peanuts.lucy
RESOURCE      ACCESS         SERVER          DESCRIPTION

GRAPHICS      read/write     peanuts.lucy    Domain files
CALENDAR      read/write     peanuts.lucy    Monthly meetings
USERHELP      read-only      peanuts.lucy    System help information
```

For each available resource, **nsquery** will list the resource name (RESOURCE), the permissions (ACCESS), the computer that owns the resource (SERVER), and the description of the resource.

NOTE

The output from **nsquery** does not indicate whether you have permission to mount the resource.

Advertised Resources in Use

You can use the **rmntstat** command to find out what remote computers have mounted your advertised resources. This command can print output for all your resources or the one you choose. The syntax is as follows:

rmntstat [**-h**] [*resource*]

where **-h** will print the output without the heading, and *resource* can be used to restrict output to information for a particular resource. Here is an example of the output from **rmntstat**:

```
# rmntstat
RESOURCE        PATH            HOSTNAMES
DNEWS           /usr/news       peanuts.linus peanuts.lucy
MDEV            /dev            peanuts.linus
SPECIAL         unknown         peanuts.charlie
```

The output shows the resources your machine has advertised, where those resources are located on the local machine, and the computers that have mounted the resource.

NOTE

If **unknown** appears in the pathname field, it means you have unadvertised the resource, but it is still mounted on the listed remote machines.

Unadvertise

You can unadvertise any of your computer's resources using the **unadv** command. It will remove the resource from the advertise tables on your computer and the domain name server.

The domain administrator can use **unadv** to unadvertise any resource within the domain. (You should only use **unadv** when the machine has gone down; otherwise, the domain and your computer's advertise tables will not match.)

Unadvertising does not remove a currently mounted resource from a remote computer [see *umount*(1M)]. It does, however, prevent additional machines from mounting the resource. There are two reasons you may want to use this command:

1. Before you can unmount (**umount** or **fumount**) one of your file systems containing an advertised directory, it must be unadvertised.
2. If you want to restrict a previously shared directory to only local access, you will want to unadvertise it.

Because advertise commands can be set up to run automatically in **init 3**, you may have to remove them if you want them permanently unadvertised. (See the "Advertise Resources" section of this chapter for information on how to modify the */etc/rstab* file.)

The syntax of the **unadv** command is

unadv *resource*
or
unadv *domain.resource*

where *resource* is used to unadvertise one of your machine's resources, and *domain.resource* is used by a domain name server to unadvertise any resource in its domain. In the second case, the resource name is prefixed by the domain name in which it resides followed by a period (.).

Forced Unmount

You cannot unmount a local file system using the **umount** command if any part of that file system is mounted remotely. Normally, you should tell each administrator whose machine has mounted such a resource to unmount it. In this way, a resource can be removed in an orderly fashion.

When you have to unmount a local file system immediately, however, you can use the **fumount** command. **fumount** will remove a remotely mounted resource from all machines that have mounted it. You should only do this in cases where it is urgent that the resource be removed, because you may be cutting off remote processes that are accessing the resource.

The syntax of the **fumount** command is

fumount [**-w** *sec*] *resource*

where the **-w** option says to wait *sec* seconds before remotely unmounting the resource, and *resource* is replaced by the resource name.

When you execute **fumount**, this is what happens:

1. The resource is unadvertised.

2. If the **fumount** command is executed with a grace period of several seconds, the following shell script is run on all client machines currently using the resource:

   /usr/nserve/rfuadmin fuwarn *resource sec*

   By default, this shell script will write to all terminals on all client machines:

   ```
   resource will be disconnected from the system in sec seconds.
   ```

   (You can edit **rfuadmin** to tailor the action taken in response to **fumount**.)

3. After the grace period of *sec* seconds, the resource is removed from all remote machines it is mounted on. The following message is then sent to all terminals:

   ```
   resource has been disconnected from the system.
   ```

4. On each client machine, **rfuadmin** then executes **rmount**. **rmount** will try to remount the resource every 60 seconds until it succeeds.

See *rfuadmin*(1M) and *rmount*(1M) for further information on processing these commands.

Remote Resource Mounting

You can attach another computer's advertised resource to your system using the standard **mount** command. You simply choose an existing directory, preferably an empty one, or create a directory to use as a mount point and mount the resource using the **-d** option.

When you try to mount a remote resource, a request is sent to the computer that advertised the resource. If you have permission to mount the resource, the resource will be added to your mount table and connected to the mount point you specified. You can list the remote resources, as well as local file systems, mounted on your computer using the **mount** command without options.

The form of the **mount** command to mount a remote resource is as follows:

mount [**-r**] [**-c**] **-d** *resource directory*

The options are as follows:

-r The **-r** option indicates that the resource should be mounted read-only. If it is not used, the resource is mounted with read/write permissions. (A remote resource can only be mounted read/write if it was advertised that way.)

-c This option indicates that remote reads and writes for the remote resource you mount should not be cached in the local buffer pool. You will generally want the default (buffer caching on) since caching will cut down on network access and improve RFS performance. (See the "Monitoring" and "Parameter Tuning" sections of this chapter for more information on monitoring client caching activities.)

-d This option is used to indicate that you are mounting a remote resource.

resource This variable must be replaced by the resource identifier assigned by the computer that advertised the resource.

directory This variable must be replaced by the full path to the local directory on which you want to mount the resource.

Automatic Remote Mounts

You can set up your **mount** commands to run automatically when your computer enters the **init 3** state. You do this by adding mount information to the */etc/fstab* file. The format of */etc/fstab* for remote mounts is as follows:

resource directory **-d[r]**

resource is replaced by the resource name, *directory* is replaced by the directory where the resource will be mounted, and **-d** says this is a remote mount. You can use **-dr** instead of **-d** if you want to mount the remote resource read-only.

Mounting Guidelines

Below are some guidelines that apply to resources.

- Once you have advertised a resource from your computer, you can
  - Mount a local file system on a subdirectory of the advertised resource. The new file system will become part of the advertised resource. (You cannot mount directly on the advertised mount point, however.)
  - Mount a remote resource on subdirectories of your advertised resource. The remote resource you mount will not become part of the resource, however. Only your local users will be able to access it. (Remote users will be able to see this mount point directory but will get a "multihop" error message if they try to access the directory in any way.)

NOTE

You cannot mount a remote resource directly on an advertised directory.

- If a resource was advertised with read-only permissions, you must mount it read-only. If it was advertised read/write, you have a choice of mounting it read-only or read/write.

Mounting Rules

There are some rules you must follow to avoid unexpected results when mounting remote resources:

Rule #1 Mounting over basic directories

A directory containing files that define your local machine should not be used as a mount point for a remote resource. This will result in essential local files being inaccessible to your system.

For example, you shouldn't mount a remote */dev* on your machine's */dev* directory or you will make your machine's console inaccessible (*/dev/console*). As another example, if you mounted an */etc* directory on your *etc* directory, you would cover your local *inittab, passwd,* and *mnttab* files, to name a few. Some other directories that fall into this category are */, /usr, /usr/bin, /usr/nserve, /usr/net,* and */shlib.*

Rule #2 Mounting spool and work directories

Like Rule #1, Rule #2 has to do with mounting a directory from one computer on the same directory on another computer. In this case, the problem is spool files and workspace directories. Applications such as **uucp** and **lp** can run into problems when multiple machines are trying to create spool files or lock files in the same directory. For example, if you share the */usr/spool/locks* directory by using a tty device for **uucp** on one machine, you would prevent use of a device of the same name on another machine. Also, mounting */tmp* can cause collisions among temporary files.

Rule #3 File systems on remote devices

When a remote machine advertises a directory containing a device and that device contains a file system, you would not be able to mount the file system by simply mounting the resource containing the device. To access the file system on the remote device, the remote machine would have to mount the device locally, then advertise that mount point. (You can access the remote as a raw device, however, by simply mounting the resource containing the device.)

Rule #4 Using remote sticky bit programs

Mounting remote resources that contain executable files with the sticky bit on can improve performance of those files. When executed on your machine, the text portion of the sticky bit program will remain in main memory on your machine, thereby reducing the network overhead on future executions. From your perspective as a client, you should be careful not to mount too many sticky bit programs or you could unknowingly use a lot of memory.

If your machine is a server sharing sticky bit files, you should be aware that they are treated differently from strictly local sticky bit files. Before removing sticky bit programs from an advertised resource, you must unmount the resource from all client machines [see *fumount*(1M) in the *User's/System Administrator's Reference Manual*], remove the program, then readvertise the resource. You should do this to prevent out-of-date text for recompiled or deleted files from remaining in memory on client machines.

Local Mount Table

You can list the remote resources that are mounted on your computer as you would list local file systems by using the **mount** command with no options. Remote resource output from this command will appear in the following form:

directory **on** *resource permission* **on** *date*

where *directory* is the name of the directory where the remote *resource* is mounted, the *permission* is **read only/remote** or **read/write/remote**, and *date* is the time and date the resource was mounted.

The following is an example of output from the mount command with no options. The last two entries in this example are remote resource mounts:

```
$ mount
/ on /dev/dsk/0s1 read/write on Tue Sept 1 09:07:19 1987
/usr2 on /dev/dsk/0s4 read/write on Tue Sept 1 09:07:32 1987
/usr on /dev/dsk/0s3 read/write on Tue Sept 1 09:07:33 1987
/s/codes on LCODE  read/write/remote on Tue Sept 1 09:10:13 1987
/s/timing on TEMPO read only/remote on Tue Sept 1 09:10:27 1987
$
```

Remote Resource Disconnected

When a machine that shares its resources with you goes down or the network connection is broken, resources that you have mounted from the server will be disconnected.

An RFS daemon process (**/usr/nserve/rfudaemon**) runs an administrative shell script (**rfuadmin**) to try to clean up when a resource has been disconnected. It then tries to remount the remote resource as soon as it becomes available again.

rfudaemon

The **rfudaemon** process is run automatically when RFS is started (**rfstart**) and continues to run until it is stopped (**rfstop**). The **rfudaemon** process waits for one of the following events to occur, then passes that information to the **rfuadmin** administrative shell script.

- **disconnect** — When a link is cut to a remote resource, the **rfudaemon** process sends a **disconnect** message and the resource name to the **rfuadmin** shell script.

- **fumount** — When a resource is unmounted (**fumount**) by the server, the **rfudaemon** process sends an **fumount** message and the resource name to the **rfuadmin** shell script.

fuwarn When a server sends a message that a resource is about to be unmounted (**fumount**), the **rfudaemon** process sends a **fuwarn** message, the resource name, and the number of seconds before the resource will be unmounted to the **rfuadmin** shell script.

rfuadmin

When links to resources are disconnected, the response to the disconnect is handled at the user level by the **/usr/nserve/rfuadmin** shell script. By editing this shell script, you can tailor the response your system makes when the connection to a remote resource is lost. The **rfudaemon** process starts **rfuadmin** with one of the following arguments:

disconnect *resource*

When **rfuadmin** is started by **rfudaemon** with these arguments, **rfuadmin** sends this message to all terminals using the **wall** command:

```
resource has been disconnected from the system.
```

Then it executes **fuser** to kill all processes using the resource, unmounts the resource (**umount**) to notify the kernel, and starts **rmount** to try to remount the resource. The assumption is that the link was either broken by mistake or that as soon as the server makes the resource available again, the client will want to mount it.

fumount *resource*

When **rfuadmin** is started by **rfudaemon** with these arguments, the processing is similar to a disconnect.

fuwarn *resource seconds*

When **rfuadmin** is started by **rfudaemon** with these arguments, **rfuadmin** sends this message to all terminals:

```
resource is being removed from the system in sec seconds.
```

There are many reasons you may want to change the **rfuadmin** shell script. If access to a resource is lost, you may want to respond by trying to mount another resource. You may want to send different messages when a resource is lost.

NOTE

When a resource is disconnected, **rfuadmin** tries to remount the resource using **/usr/bin/rmount**. This command retries the remount every 60 seconds until it succeeds. To change this behavior, you must either edit **/etc/rfuadmin** so it no longer does an **rmount**, or edit **rmount** so it retries a limited number of times [see **rmount**(1M)].

Unmounting

You can unmount any remote resource you have mounted with the **umount** command. The syntax for using **umount** to unmount a remote resource from your computer is as follows:

umount -d *resource*

where *resource* is replaced by the name of the resource you are unmounting.

Before you run **umount**, you should make sure none of your users are using the resource with the **fuser** command. When the **fuser** command is run, it lists the processes on your computer that are accessing a mounted remote resource. It can then be used to kill all processes relating to a resource.

The form of **fuser** for reporting on remote resources mounted on your machine is

fuser [**-ku**] *resource* ...

where **-k** will kill all processes that have files open in any directory or subdirectory relating to the resource, and **-u** will list user names in the report of processes that have files open in any directory or subdirectory relating to the resource.

Mapping Remote Users

NOTE

The procedure in the "Complex User ID/Group ID Mapping" section is designed to act as a tutorial for setting up ID mapping. This section provides further reference information to support that section.

Your computer has a set of users, defined in the */etc/passwd* file. These users can also be members of groups that are defined in the */etc/group* file. The user and group ID assignments are used by the system to evaluate requests by the user for access to local files, directories, and devices.

When you share your directories with other computers using RFS, you have the ability to define the permissions each remote user will have to your resources. You do this by mapping remote users and groups into the permissions of existing users and groups on your computer. You also have the option of mapping remote users and groups into a special "guest ID" that doesn't map into permissions of any existing users and groups on your system.

If you do not want to map remote users, you do not have to. The default treats all remote users as the special guest ID, which has the ID number of MAXUID plus one. MAXUID is the maximum ID number defined for the system, so MAXUID+1 is always guaranteed not to overlap with any current or future users (by default, MAXUID+1 is 60001).

When a remote user checks the ownership of one of your resources, the user might see another special ID: MAXUID+2 (or 60002). No files will ever be owned by 60002 on the system where a file resides. MAXUID+2 is simply a way of telling remote users that a file or directory is not owned by them or any other users from their system. For example, if all users on a remote system were mapped to 60001, any files created by one of your local users would appear to be owned by user ID 60002.

How Mapping Works

When you set up your remote user and group mapping for a remote computer, you define how requests from users and groups will be handled. This mapping has an impact on the remote users' access to files and directories on your resources, as well as each remote user's view of ownership.

For example, say you map user ID **101** from machine **abc** into user ID **115** on your machine. When **101** from **abc** tries to create a file in a directory of one of your advertised resources, your machine will translate the request from **abc**'s **101** into a request from **115**. If local ID **115** has permissions to create a file in that directory, then the file will be created.

If you tried to **stat** the file on your machine (**ls -l**, for example) you would see that user ID **115** was the owner. However, if a **stat** comes from machine **abc**, your machine would do inverse mapping. Therefore, the user from **abc** would see the file as being owned by user ID **101**.

Inverse mapping from the machine that owns the resource (the server) provides the most consistent file system view to a remote user. It could potentially cause confusion, though. Continuing with the example, say that instead of just mapping **101** into **115**, you also mapped **102** from **abc** into **115** on your machine. A file created by **102** would correctly create the file as owned by **115** on your machine. However, when a user from **abc** **stat**s the file, it would always show ownership by the smaller numeric value: user ID **101**.

NOTE

This same result will occur if you gave several local user names the same numeric user ID.

If users are confused when files they create do not seem to belong to them, the situation described above could be the reason. This does not cause any problems with each user's ability to access the resource. However, it could break some programs that are dependent on local IDs. The most consistent way to map, however, is one-to-one remote to local IDs.

Mapping Components

You must use the **idload** command to do the user and group mappings. This command reads the user and group mapping rules you create, reads your computer's */etc/passwd* and */etc/group* files, if needed, and maps the remote users into your users' permissions. If you are using remote user and group names to map into your computer, you must have access to user and group lists from the remote computers, so **idload** can read the files and translate those names into the appropriate numeric ID numbers.

Rules Files

The rules files you create will tell **idload** how to map remote users. Both files are in */usr/nserve/auth.info* under the names *uid.rules* for user rules and *gid.rules* for group rules.

Figure 9-6 shows how the user rules file can be structured. The format of the group rules files is exactly the same. All lines in each file are optional.

global
default *local_id* | **transparent**
exclude [*remote_id-remote_id* ...] | [*remote_id*]
map [*remote_id:local* ...]

host *domain.nodename* ...
default *local* | **transparent**
exclude [*remote_id-remote_id* ...] | [*remote_id* ...] | [*remote_name* ...]
map [*remote:local* ...] | *remote* | **all**

Figure 9-6: Format of *uid.rules* and *gid.rules* files

The following notation is used in the previous figure:

local_name = a local user name
local_id = a local user ID number
remote_id = a remote user ID number
remote_name = a remote user name
local = a local name or ID number
remote = a remote name or ID number

A rules file is divided into blocks of information. Each block is either a **global** or **host** block. There is only one **global** block per file, but there can be one **host** block for each computer mapped.

global This line starts the block of global information. Each line of definitions after **global** and before the first **host** line will be applied to all computers that are not explicitly defined in **host** blocks. You can use **default**, **exclude**, and **map** inside **global** blocks.

You cannot map or exclude names in **global** blocks. You must use ID numbers.

host *domain.nodename ...*
This line starts a block of information for a particular computer. Each line of definitions following this line and before the next **host** line will be applied to the *domain.nodename* specified. You can use **default**, **exclude**, and **map** inside **host** blocks.

If you want to map more than one computer from a single set of *passwd* and *group* files, you can put several computer names on one line. In this case, **idload** will read the *passwd* and *group* files for the first computer referenced (if you map by name) and use the information in those files for all computers that are referenced.

A computer can only be mapped once in each rules file.

Each of the following lines of information can appear in either a **host** block or a **global** block. A name or an ID should only be mapped once in each block. If one is mapped more than once, the first reference is in effect and the others will produce warning messages from **idload**.

1. **default** *local* | **transparent**

   One **default** line can be put in each block to indicate how to handle remote users and groups that are not explicitly mapped or excluded.

   transparent means use the same numeric ID on your machine that the user had on the remote machine for undefined users. So if a request comes from remote uid **101**, that request will have the permissions of local uid **101**.

   local is replaced by a local user name or ID number. By default, all remote users will be mapped into the permissions of the local user indicated by name or ID. If a default line does not appear in a block, MAXUID+1 permission will be assigned.

2. **exclude** [*remote_id-remote_id*] | [*remote_id*] | [*remote_name*] ...

   Optional **exclude** lines can go into a block to exclude certain users from the default mapping. Zero or more ranges of ID numbers (*remote_id-remote_id*), single *remote_names*, or single *remote_id* numbers can be excluded. (*remote_name* is not available in the global block.)

   A user who is excluded will still have access to your resources but will only have permissions of the MAXUID+1 user. All **exclude** lines must go before any **map** lines in a block.

3. **map** [*remote:local*] | *remote* | **all**

   You can use **map** lines in each block to assign local permissions to particular remote users. There are several ways to use the **map** command. You can set any remote user's permissions to any local user's permissions by either local user *id#* or *name*, separating the two with a colon (:). By entering a single *remote_id* or *remote_name*, the remote user who matches will have the permissions of the local user of the same ID or name. For example:

   map mcn

   would give the remote **mcn** the same permissions of the local user **mcn**.

   The literal entry **all** maps all users by user name into the permissions of users with the same name on your computer.

Multiple **map** lines are valid. You cannot map by remote name in **global** blocks.

NOTE

map all and mapping by name are not allowed in a global block. **map all** will usually produce warning messages since multiple administrative logins will have uid 0 and **idload** will try to map each one to 0. There is no harm in this.

idload Command

Once the rules files are created, use the **idload** command to read your rules files and create mapping translation tables. When you run **idload**, the rules in **global** blocks and any **host** blocks that have resources currently mounted immediately take effect. All other **host** block rules will take effect when the remote machine mounts one of your resources.

The syntax of **idload** is

idload [**-n**] [**-k**] [**-g** *g_rules*] [**-u** *u_rules*] [*directory*]

The options are as follows:

-n This is the "no update mode" option. When it is used, **idload -n** will print the mapping that would result from the rules files without putting them into effect.

-k This option shows the mapping that is currently active on your machine. (Note that there will be mapping ready to take effect that is not shown as active when you do not currently have a connection to a remote machine.)

-g *g_rules* This option lets you use a group rules file other than */usr/nserve/auth.info/gid.rules* as input for group mapping rules.

-u *u_rules* This option lets you use a user rules file other than */usr/nserve/auth.info/uid.rules* as input for user mapping rules.

directory This option indicates that some directory other than */usr/nserve/auth.info* contains the *domain/nodename* directories where the *passwd* and *group* files for each remote

computer reside. If it is not used, */usr/nserve/auth.info* will be assumed.

Each time you set up or change your rules files, first run **idload** with the **-n** option. The results will show you the mapping that will occur when the command is run to actually load the IDs. You must then run **idload** for the rules to go into effect.

Remote Computer passwd and group Files

If you are mapping remote users by name, you will need lists of these users from each remote computer. These lists should be copies of the */etc/passwd* and */etc/group* files from each computer.

If **idload** finds a request for a remote user name in a **host** information block, it will check the directory for that computer for *passwd* and *group* files. The pathname to the remote computer's directory will be

/usr/nserve/auth.info/domain/nodename

on your system, where *domain* and *nodename* are replaced by the remote computer's domain and the remote computer's nodename, respectively (unless you overrule this using the **-g** and **-u** options).

NOTE

Mapping by name can be a very useful feature. However, if you map only by ID number or local name and avoid mapping by remote names, you will avoid the need to coordinate distributing and updating remote *passwd* and *group* files and rerunning **idload**.

Example Rules Files

This section describes some strategies you can use to map users. It describes the easiest way to deal with remote user permissions and progresses to the most complicated ways. Read through each example to decide what strategy is best for your computer.

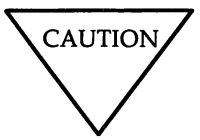
No Mapping

If you do not run **idload** to map users, all remote users will have the permissions of the user ID number MAXUID +1, which is the maximum ID number defined on your system plus one. Because there are no users on your system with that user ID number, remote users will only have access to files created by your users that are open to all users.

Mapping Remote IDs

If you map remote users using remote ID numbers and local ID numbers and names, you do not need to get any *passwd* and *group* files from remote computers. The following displays contain some simple examples of mapping that only involve remote ID numbers.

In Figure 9-7, all remote user IDs will be mapped into the same user ID permissions on your computer except for **root** (ID number **0**), which would only have special guest permissions. This would apply to all remote computers.



The exclude 0 line is strongly recommended to prevent possible security breaches from **root** users on other systems.

```
global
default transparent
exclude 0
```

Figure 9-7: *uid.rules* File: Setting Global Defaults

In Figure 9-8, users have the same permissions as in the previous example except remote user IDs **0** through **100** will have MAXUID +1 permissions and any user ID **732** would have the same permission as local user ID **106**.

```
global
default transparent
exclude 0-100
map 732:106
```

Figure 9-8: *uid.rules* File: Global Mapping by Remote ID

In Figure 9-9, the users from computer **lucy** in domain **peanuts** will not be mapped by the global rules. Instead, all users will have the permissions of local user **mpg** except that user IDs 0 through 50 will have MAXUID +1 permissions.

```
global
default transparent
exclude 0-100
map 732:106

host peanuts.lucy
default mpg
exclude 0-50
```

Figure 9-9: *uid.rules* File: Host Mapping by Remote ID

Mapping Remote Names

If you want to use specific remote user names to map into your local users' permissions, you will need to have access to *passwd* and *group* files from those computers on your system. Below are some examples of ways you can map remote user names.

map all

If you have the same set of user names on different machines but the user IDs differ, you may want to use **map all** as shown in Figure 9-10.

In Figure 9-10, each user name from computer **lucy** in domain **peanuts** will have the same permissions as the same user name on your computer. The only exceptions will be users **mary**, **root**, and **uucp**, who will have MAX-UID +1 permissions.

```
global
default transparent
exclude 0

host peanuts.lucy
exclude mary 0 uucp
map all
```

Figure 9-10: *uid.rules* File: Mapping by Name With **map all**

map name:name

You can also map particular remote user names into local user names or user IDs on your computer. Figure 9-11 is an example.

```
global
default transparent
exclude 0

host peanuts.lucy
default transparent
exclude 0
map mcn:jcb ral gwn:103
```

Figure 9-11: *uid.rules* File: Mapping Specific Users by Name

Here all users from the computers will be mapped into their same user ID with the following exceptions. Remote user **mcn** will have the permission of local user **jcb**, remote user **ral** will have permissions of local user **ral**, and remote user **gwn** will have permissions of local user ID **103**.

List Current Mapping

There are two ways to list the mapping you have set up: **idload -n** and **idload -k**. The **-n** option inspects the rules files and prints a listing of what would be in effect were you to load them. The **-k** option prints the mapping that is currently in effect in the kernel.

Figure 9-12 shows the result of **idload -n** used for the example shown previously. (The *gid.rules* file simply has the global block set at **default transparent**.) The **-n** option says to print the mapping that is set up in the file. You should do this before you run **idload** without options so you can see the mapping that will take effect.

```
# idload -n

TYPE   MACHINE          REM_ID      REM_NAME    LOC_ID          LOC_NAME

USR    GLOBAL           DEFAULT     n/a         transparent     n/a
USR    GLOBAL           0           n/a         60001           guest_id
USR    peanuts.lucy     DEFAULT     n/a         transparent     n/a
USR    peanuts.lucy     0           n/a         60001           guest_id
USR    peanuts.lucy     100         mcn         105             jcb
USR    peanuts.lucy     102         gwn         103             n/a
USR    peanuts.lucy     191         ral         101             ral

GRP    GLOBAL           DEFAULT     n/a         transparent     n/a
```

Figure 9-12: Output From **idload -n**

If you were to then run **idload**, the mapping shown above would take effect. If you were then to run **idload -k**, and the machine called **peanuts.lucy** did not have a resource mounted, you would see that the output in Figure 9-13 was active.

```
# idload -k

TYPE   MACHINE    REM_ID      REM_NAME    LOC_ID          LOC_NAME

USR    GLOBAL     DEFAULT     n/a         transparent     n/a
USR    GLOBAL     0           n/a         60001           guest_id

GRP    GLOBAL     DEFAULT     n/a         transparent     n/a
```

Figure 9-13: Output From **idload -k**

All mapping to **peanuts.lucy** will become active as soon as you are connected to it.

NOTE

The output from **idload** with the **-n** and **-k** options could be different if you have changed the rules files but not yet run **idload** without options. Also, the **-k** option will not show mapping for computers that are not currently mounting a resource from your machine, even though the mapping would be in effect as soon as the remote machine mounted one of your resources.

Domain Name Servers

One machine in each RFS domain must be chosen to be the primary name server, and zero or more can be secondary name servers. The duties of these machines are described briefly under the "Name Service" subsection in the "Overview" section of this chapter. This section describes the "how-to" of being a name server.

Before you run any of these tasks, you will want to know which machines are assigned as name servers and which machine is the current name server. To find the current name server, type

rfadmin

To find the name server assignments, type

cat /usr/nserve/rfmaster

The line in the *rfmaster* file that has a **P** in the second field designates the primary name server. If an **S** is in the second field, the entry designates a secondary name server. (Lines with **A** in the second field designate the network address of a primary or secondary.)

Primary Name Server

If your machine is the primary domain name server, you are responsible for maintaining domain information. The "Setting Up RFS" section of this chapter describes primary name server responsibilities as you set up your machine and domain. You may refer to the following subsections to change your RFS configuration after initial configuration:

- Create *rfmaster* File
- Add/Delete Domain Members
- Resource Sharing with Other Domains
- Multiple Domain Name Service

NOTE

If you want to change the primary and secondary designations in the *rfmaster* file for a domain that is currently running, you must follow this procedure to make sure those changes are properly put in place:

1) Stop RFS on all primary and secondary domain name servers for the domain (**rfstop** or **init 2**).

2) Change the *rfmaster* file on the old primary and the new primary.

3) Start the primary designated in the new *rfmaster* file (**rfstart** or **init 3**).

4) Start the secondaries designated in the new *rfmaster* file (**rfstart** or **init 3**).

Once changed on the name servers, each individual computer will pick up the change the next time it starts RFS.

Secondary Name Server

Because a secondary is only intended to take over domain name service temporarily, its main responsibility is to pass name server responsibility back to the primary as soon as possible; it does not happen automatically. Most domain maintenance (adding new computers or changing the *rfmaster* file) cannot be done while the secondary is acting domain name server. The secondary simply maintains information that machines need to mount and advertise resources.

To pass name server responsibility back to the primary once it is again running RFS, type the following from the secondary:

rfadmin -p

This command will pass the domain name server information to the primary or to one of the other computers listed in the domain's *rfmaster* file if it cannot contact the primary. (Note that name service will automatically be passed off when the current name server goes down.)

Recovery

As a domain name server, computers in your domain rely on your machine for information on domain resources and domain member machines. RFS is designed to recover quickly when communication is cut between machines and the name server. The following sections describe RFS events that can occur and the recovery mechanisms designed to handle them.

Primary Goes Down

All essential domain records are maintained on the primary domain name server. The primary regularly distributes the most critical of these records to secondary domain name servers. (These records do not include files and directories under */usr/nserve/auth.info*.)

If the primary goes down, domain name server responsibilities are passed to the first secondary name server listed in the *rfmaster* file. The secondary is only intended to take over temporarily. The reason is that a secondary has limited name service capabilities. This is done to maintain the definitive domain records on the primary. Changing the name server does not affect any currently mounted resources.

While a secondary is acting domain name server, you cannot

- Maintain domain member lists

  Computers cannot be added or deleted from domain member lists while a secondary is acting domain name server.

- Change RFS passwords

  Neither the secondary nor another computer can change RFS authentication passwords while a secondary is acting domain name server.

The secondary will maintain lists of advertised resources for the domain and continue basic name server functions so RFS activities can continue. In most cases, the computers in the domain shouldn't be aware the primary is down. When the primary comes back up, the secondary should pass name server responsibilities back to the primary using the **rfadmin -p** command.

NOTE

When a primary crashes without properly shutting down RFS and passing name server responsibilities to a secondary in an orderly fashion, the advertise table on the secondary may contain some errors. Resources from the primary may still be listed as available, and recently advertised resources from other computers may not appear on the list. You can fix the domain advertise table using the **unadv** and **adv -m** commands from the domain name server.

Primary and Secondaries Go Down

If all primary and secondary name servers go down at once, all information on advertised resources will be lost. Active mounts and links, however, are not disturbed. The problem is that when the primary comes back up, each computer will still think its resources are advertised, but the primary will have no record of these advertised resources.

As soon as the primary is running, each computer can make sure its advertised resources are in sync with those listed on the primary in one of two ways:

- Readvertise with **adv -m**

  This is a less drastic way to update the advertise tables on the primary. Readvertise each resource using the **adv -m** command from the computer where the resource resides. This command will get the primary and remote computer's advertise tables back in sync.

- Restart Remote File Sharing

  You can bring down RFS, bring it back up, and then readvertise your resources. You can do this automatically by going from **init 3** to **init 2** to **init 3**.

Monitoring

This section describes the commands used to monitor RFS activity, the reports they produce, and possible action you can take to make sure that your system is operating at peak efficiency. In general, these reports can help you decide if you want to

- Change parameter settings to better match the way your system is used.
- Move resources from machines with heavy RFS traffic to machines with lighter traffic.
- Use sticky bit programs across the network. (See the "Mounting" section of this chapter for special rules relating to sharing sticky bit programs.)

A description of all RFS tunable parameters and suggested initial settings appear at the end of this section.

The **-D** option of **sar** is used to produce RFS-specific information along with standard **sar** reports (**c**, **u**, and **b** options).

Remote System Calls (sar -Dc)

Your computer collects data each time a system call sends a message across an RFS network to access a remote file. You can print this information using **sar -Dc** (see Figure 9-14).

The report produced by **sar -Dc** contains the average system calls per second; average read and write system calls per second, including average characters read and written per second; and average **exec** per second (see Figure 9-14).

Information is divided into three categories: incoming requests (another computer's request for your resources), outgoing requests (your computer's request for a remote resource), and strictly local system calls.

```
$ sar -Dc
lucy lucy 3.0 2 computer    02/14/86

00:00:04  scall/s  sread/s  swrit/s  fork/s  exec/s   rchar/s  wchar/s
01:00:04
   in         4        1        2             0.00       350      220
   out        3        2        1             0.00       240      300
   local    133       30       12     0.73    1.33     11202     3813
02:00:04
   in         4        1        2             0.00       350      220
   out        3        2        1             0.00       240      300
   local    133       30       12     0.73    1.33     11202     3813
03:00:02
   in         4        1        2             0.00       350      220
   out        3        2        1             0.00       240      300
   local    133       30       12     0.73    1.33     11202     3813
04:00:02
   in         4        1        2             0.00       350      220
   out        3        2        1             0.00       240      300
   local    133       30       12     0.73    1.33     11202     3813

Average
   in         4        1        2             0.00       350      220
   out        3        2        1             0.00       240      300
   local    133       30       12     0.73    1.33     11202     3813
$
```

Figure 9-14: Output From **sar -Dc**

NOTE

Some statistics will not reflect the actual number of messages sent across the network since the client-caching feature allows some remote read requests to be satisfied from data in local buffers. Outgoing scall/s, sread/s, and rchar/s fields include statistics for these read "hits" of remote data in the client cache. Though these reads do not result in actual messages to the remote machine, they are still categorized as outgoing since they access remote data.

The following paragraphs describe how information from the **sar -Dc** report can be useful to you. If performance is poor, you can see how efficiently system read and write calls to and from your computer are using the RFS network. For incoming (in) and outgoing (out) system calls, divide the characters read or written by the reads and writes, respectively.

If your computer is attempting more than about 30 remote system calls per second (in and out scall/s), you are probably nearing capacity. Performance problems will probably result from this much demand. Remote **exec**s also put a heavy demand on a computer. Selective use of sticky bit programs can help improve performance.

You may want to consider moving resources to machines where they are most in demand. (See the **fusage** command to determine what resources are being used most heavily.)

CPU Time (sar -Du)

You can list the percent of total central processing unit (CPU) time spent on system calls from remote computers (%sys remote) with the **sar -Du** command (see Figure 9-15).

```
$ sar -Du
lucy lucy 3.0 2 computer   02/14/86

00:00:04      %usr      %sys     %sys     %wio    %idle
                       local   remote
01:00:04         7         21       10       28       44
02:00:04        11          9       10        4       76
03:00:02         8         18       10       17       57
04:00:02         2          4       10        1       93
05:00:03         1          4       10        1       93
06:00:02         2          5       10        2       91
07:00:02         1          4       10        1       94
08:00:02         2          5       10        2       91
08:20:02        26         16       10       11       48
08:40:02        18         11       10        9       62
09:00:17        25         21       10       13       41
09:20:18        23         21       10       11       45
09:40:20        21         24       10       15       39
10:00:09        21         29       10       17       33
10:20:14        29         28       10       13       31
10:40:18        19         20       10        7       54

Average          9         12       10        8       71
$
```

Figure 9-15: Output From **sar -Du**

If the percent of CPU time spent servicing remote system calls is high, your local users may be suffering. (However, if the computer is a server machine, you would expect %sys remote to be high.)

To reduce the time spent servicing remote requests, you may want to place the resource(s) in demand on another computer (see the **fusage** command) or limit resource access by changing some of the tunable parameters. (See the section titled "Parameter Tuning.") You may also want to make sure clients are doing I/O in an efficient way (see **sar -Dc**).

Client Caching (sar -Db and sar -C)

The client-caching feature of RFS improves RFS performance by reducing the number of times data is retrieved across the network. With client caching, the first read of data will bring the data into local buffers. Once data is in the local buffer, it will remain there so that subsequent reads can get the data locally.

Client caching is assigned by default on a systemwide basis (RCACHETIME parameter) and when you mount a remote resource. You will almost always want to take advantage of the improved performance of client caching. There are only two very rare occasions when you may not want to use client caching:

- If buffer space is limited on your system, you may choose to turn off client caching for some resources or the entire system.
- If you are using programs that do their own private network buffering, you may not want to use client caching.

You can produce two **sar** reports to monitor caching activities.

Caching Buffer Usage

The **-b** option of **sar** reports the buffer pool usage for local (disk) reads and writes. The **sar -Db** option reports the same information plus information on buffer pool usage of locally mounted remote resources (see Figure 9-16).

```
$ sar -Db

charlie charlie 3.1 2 computer    09/03/86

14:37:15 bread/s lread/s %rcache bwrit/s lwrit/s %wcache pread/s pwrit/s
14:37:18
   local      2       40       93       1       3       64       0       0
   remote     1       11       92       1       1        0
14:37:21
   local      2       39       92       1       3       63       0       0
   remote     0       10       94       1       1        0
14:37:24
   local      2       40       93       1       3       64       0       0
   remote     1       12       93       1       1        0

Average
   local      2       40       93       1       3       64       0       0
   remote     1       11       93       1       1        0
```

Figure 9-16: Output From **sar -Db**

The fields on this report are as follows:

`bread/s` The number of read buffer misses per second. (Each miss results in a read message to the server.)

`lread/s` The number of read cache accesses per second.

`%rcache` Read cache hit ratio (100 – ((breads)/(lreads) * 100)).

`bwrit/s` Number of write buffer misses per second. All writes are sent to the server. Cache buffers affected by the writes are updated (write-through policy). This field indicates the numbers of `lwrites` that did not require a write-through. If data did not require a write-through it means that no data affected by the `lwrit` was present in the cache. (The information in this field has no performance implications when compared to using caching versus not using caching.)

`lwrit/s` The total remote write cache accesses per second.

`%wcache` Write cache hit ratio (100 – ((bwrits)/(lwrits) * 100)).

`pread/s` Not reported for remote use.

`pwrit/s` Not reported for remote use.

Cache Consistency Overhead

Information on the overhead related to maintaining cache consistency is listed with **sar -C** (see Figure 9-17).

```
$ sar -C

charlie charlie 3.1 2 computer    09/03/86

14:36:56 snd-inv/s snd-msg/s rcv-inv/s rcv-msg/s dis-bread/s blk-inv/s
14:36:59       0.0       1.1       0.0       1.5         0.0       0.2
14:37:02       0.0       0.6       0.0       0.5         0.0       0.4
14:37:05       0.3       0.6       0.0       0.5         0.0       0.1

Average        0.1       0.9       0.0       0.8         0.0       0.2
```

Figure 9-17: Output From **sar -C**

The fields on this report are as follows:

`snd-inv/s` The number of invalidation messages sent by the server per second to inform client machines about changes to server files.

`snd-msg/s` The total number of outgoing RFS messages sent per second.

`rcv-inv/s` The number of invalidation messages received by client from the server. Each message informs the client that the contents of one or more of its cache buffers may have been modified by a write on the server. The client machine reacts by invalidating data in the affected buffers so the buffers can be used for other purposes.

`rcv-msg/s` The total number of incoming RFS messages received per second.

`dis-bread/s`
When an invalidation message is received, caching is turned off until the writing process closes or until a time interval has elapsed (set by the tunable parameter RCACHETIME). This counter tracks the number of buffer reads that normally would be eligible for caching in a resource with caching turned on but are not added to the buffer pool because caching for this resource is temporarily turned off. It indicates the penalty of running uncached and provides a basis for tuning the RCACHETIME parameter.

`blk-inv/s` The number of buffers removed from the client cache as a result of receiving an invalidation message while a remote file is open or reopening a remote file that has been modified since the last close on the client.

Server Processes (sar -S)

Every request from a remote computer to access your resources is handled by a server process. When there are too many requests for the servers to handle, they are delayed and placed on the request queue. Requests leave the request queue when servers are available. Information on server availability and requests awaiting service are listed with **sar -S** (see Figure 9-18).

```
$ sar -S

lucy lucy 3.0 2 computer    02/14/86

00:00:04  serv/lo-hi   request   request   server   server
            3 - 6      %busy     avg lgth  %avail   avg avail
01:00:04      3           0         0        100       3
02:00:04      3           0         0        100       3
03:00:04      4          80         8         20       2
04:00:04      6         100        25          0       0

Average      15          50        15         70       2
$
```

Figure 9-18: Output From **sar -S**

As an administrator, you can set the number of server processes available to service remote system calls (see "Parameter Tuning"). There are two server variables you can set: MINSERVE and MAXSERVE. MINSERVE is the number of servers that are initially running to service remote requests. MAXSERVE is the maximum number of servers that may ever exist. If demand goes beyond what the MINSERVE servers can handle, extra servers can be dynamically allocated so the total number of servers can be as high as the value of MAXSERVE. These processes disappear when they are no longer needed.

Information from **sar -S** can be used to tune your server parameters as in Figure 9-19.

Too Few Servers

If the receive queue is almost always busy (request %busy), you may want to raise the number of servers. Here is how to decide the parameter to raise:

- Raise the MAXSERVE if the total average servers is high.
- Raise the MINSERVE if the total average servers is low.

Too Many Servers

If servers are available nearly 100% of the time (server %avail), you may have allocated too many servers. To decide which parameter to lower:

- Check the number of total servers. If this number is near the MINSERVE value, you can lower MINSERVE. Try reducing it by 50% or by the number of idle servers.
- Check the total servers that are idle. If this number is near the MAXSERVE value, you can lower MAXSERVE. Try reducing it by 50%.

Resource Usage (fusage)

You can find out how extensively remote computers are using your resources with the **fusage** command. It reports how many kilobytes were read and written from your resources, broken down by remote computers that have access to the resources. The form for **fusage** for reporting on a resource you have advertised is

fusage *advertised-directory*

where *advertised-directory* is the full pathname to one of the directories you have advertised. The **fusage** command with no options produces a full report of data usage for all disks and advertised directories on your system, as shown in Figure 9-19.

```
# fusage

FILE USAGE REPORT FOR charlie

        /dev/dsk/0s1          /

                              /
                                   charlie       649 KB
                                 Clients           0 KB
                                   TOTAL         649 KB

        /dev/dsk/0s3          /usr

                              /usr
                                   charlie       563 KB
                                 Clients           0 KB
                                   TOTAL         563 KB
```

Figure 9-19: Output From **fusage**

If a remote computer's requests for your resources are high, it may be causing performance problems on your computer. With the output from **fusage**, you can see what resources are being particularly hard hit. You may then decide to move the resource. You may want to move or copy a resource to a computer that is constantly accessing it.

Remote Disk Space (df)

You can use the standard **df** command with a remote resource name to see the space left on the disk on which the remote resource resides. The form of the command to report on a remote resource is

df *resource*

where *resource* is the name of a remote resource mounted on your machine. (The **df** command with no options will produce information for all mounted remote resources plus all locally mounted devices.) Figure 9-20 contains an example of the **df** command using resource names as options.

```
# df USERsrc USERmail

/usr/src     (USERsrc      ):      5436 blocks      2202 i-nodes
/usr/mail    (USERmail     ):      5436 blocks*     2202 i-nodes
```

Figure 9-20: Output From **df**

NOTE When multiple remote resources that reside on the same disk are reported, all listings of space on that disk after the first resource will be noted with an asterisk.

If you have write permission to a resource, you have as much access to file system space as a user on the system who owns a resource. This command will tell you the potential disk space available for you to write in. (Note that the space reported will only be for the top file system related to each resource.)

Parameter Tuning

There are several parameters you can tune to best suit the way you use RFS. The RFS parameters control the amount of resources you devote to RFS service. Each network transport provider may also have some tunable parameters that may affect performance characteristics of that particular network. See the network documentation for your network for more details.

Refer to the "Remote File Sharing Parameters" section in Chapter 5, Customizing Your Computer, for a description of RFS tunable parameters.

Appendix A: Change Other User's Password

Appendix A: Change Other User's Password

Change Password Procedure

To change the password for other users, use the following procedure:

1. Log in and use the **/bin/su** command to obtain *root* permissions by typing

```
$ /bin/su Enter
```

2. Type the *root* password and strike Enter.
3. When the # prompt appears, type

```
# passwd login_name Enter
```

and respond to the prompts. *login_name* is the login name of the password that you want to change.

4. When prompted for the new password,

```
New password:
```

enter the new password you want.

The password you enter will not be displayed on the screen.

You will receive an error message in the following circumstances:

- if you enter the old password incorrectly
- if the new password is not six characters long
- if the new password does not have two alphabetic characters and at least one special character in the first eight
- if the password resembles the login name by being a reverse or circular shift
- if the new password does not differ from the old password by 3 or more characters
- if you enter the new password incorrectly the second time

5. When prompted to repeat the new password, type your password again:

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

If the two password entries are the same, the password is assigned. If the two password entries don't match, the following message appears:

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

If this message appears, type the new password again and then re-enter the new password again.

Appendix B: Adding Basic Networking

Appendix B: Adding Basic Networking

Basic Networking Procedures

Adding Basic Networking involves the following basic steps.

1. Choose to physically connect your computer to another computer by adding one of the following:

a direct link	Physically connect the null modem cable from the built-in serial port on your computer to a port on another computer. See "Physical Connection of computer to DTE Direct Link" in this Appendix for details.
a modem	For the modem selected, set the appropriate options per "Recommended Switch Settings" in this Appendix or per modem documentation. Then physically connect the modem using RS-232 connectors with customized modem cable to the computer. See "Physical Connection of computer to Modem (DCE)" in this Appendix for details.

2. Logically connect the modem or direct link to the UNIX Operating System. This involves using the administration menu to update the appropriate support files to reflect the presence of a direct link or modem. See "Basic Networking Software and Direct Links" in this Appendix and Chapter 8, Basic Networking Administration, for details.

Direct Links and Modems

This section discusses the following configurations:

- computer to Data Terminal Equipment (DTE) direct link
- AT&T computer to Data Communications Equipment (DCE) such as a modem.

Your computer will connect with any other machine with an RS-232 port. The modems supported with your computer will be any kind of auto dial modems.

An advantage of using a direct link is that the link is always available and the time required to access the link is short. Direct links would be beneficial only when:

- The two machines transfer large amounts of data on a regular basis
- The two machines are located no more than several hundred cable feet apart.

NOTE	The procedure for setting up direct links through additional serial ports on expansion cards may differ. Refer to the expansion card documentation for that information.

The amount of cable used to link two machines is dependent on the environment the cable is run. The standard for RS-232 connections is 50 feet or less. As the cable length is increased, noise on the lines may become a problem. This means that the transmission rate must be decreased or limited distance modems should be placed on each end of the line. Normally, you should not use more than 1000 cable feet to connect the two machines.

The advantage of using a modem is that a port is not dedicated to only one computer. You can also be networked to a remote computer located anywhere in the world where the telephone network exists. The disadvantages are that the port of the remote computer is often busy and the transmission rate is slower.

Physical Connection of Computer to DTE Direct Link

Connecting a computer to another (DTE) RS-232-C device (e.g., another computer, UNIX PC, etc.) requires the use of a null-modem cable that must be constructed as follows:

Pin 1 to 1
Pin 2 to 3
Pin 3 to 2
Strap pin 4 to 5 in the same plug
Pin 6 to 20
Pin 7 to 7
Pin 8 to 20
Pin 20 to 6
Pin 20 to 8.

In Figure D-1, **in** means external source and **out** means computer is source. In wiring asynchronous cables from modem to built-in port, the pins have the following meanings:

Pin	Description
1	Frame Ground
2	Data **in**to computer
3	Data **out** of computer
4	Clear to send **in** (must be positive to emit data—will float positive)
5	Request to send **out** (normally positive)
6	Data terminal ready **out** (operates if minor device number includes 128)
7	Signal Ground
8	Data Carrier detect **in** (must be positive to receive data—will float positive)
20	Data set ready **in**

Figure B-1: Pin Descriptions for Null-Modem and Modem Cable

Wiring for Direct Link

Null-modem cables are commercially available for the direct link. If you desire to customize your own null-modem cable, nine leads must be wired for a connection to be made as shown in Figure D-2. Do not attach wiring to unused signals.

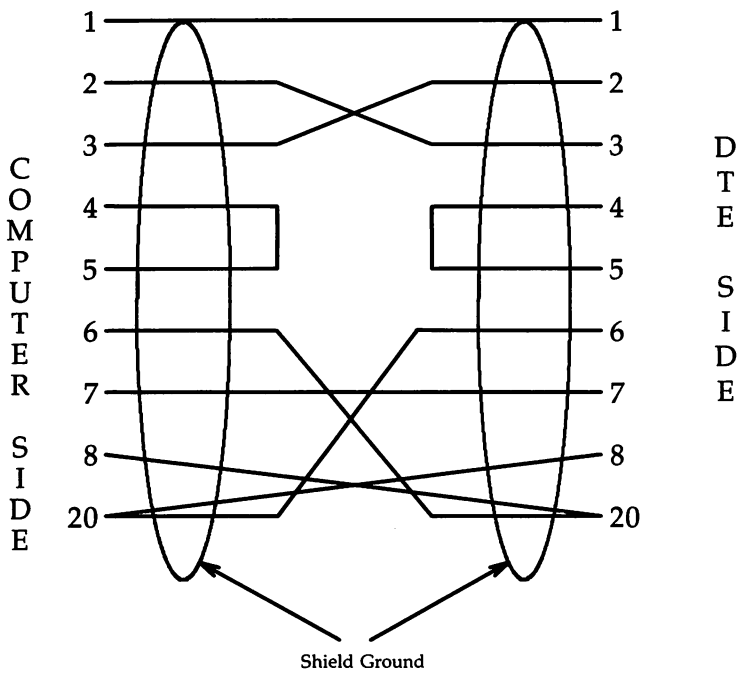
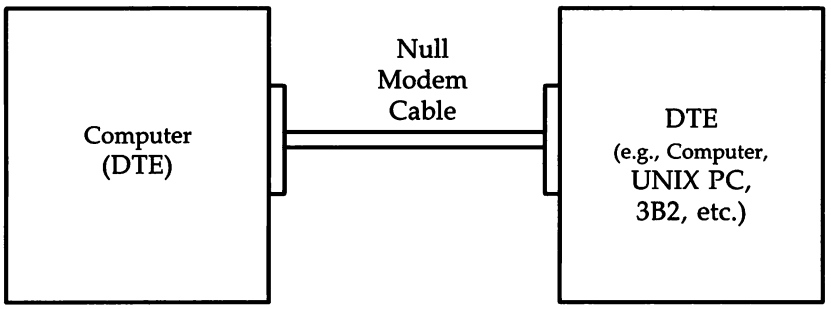


Figure B-2: Connector Wiring Diagram for DTE Direct Link to Computer

Figure D-3 shows a simple illustration of how an computer connects to a DTE direct link.



DTE -- Data Terminal Equipment

Figure B-3: Physical Connection of Computer to DTE Device (Direct Link)

Basic Networking Software and Direct Links

Ideally, systems that have a direct link should run common and current releases of the UNIX System to have the full set of capabilities available. (Bidirectional ports that are supported by the **uugetty** program were introduced with UNIX System V, Release 2.0, Version 1.) However, lack of commonality does not prevent utilization of the Basic Networking feature. This section describes the software files that must be modified on your computer in order to accommodate a direct link connection. You may want to consult the documentation provided with your machine if you're linking directly to a remote machine other than an computer.

The following support files must be updated to reflect the presence of a Direct Link:

- **/usr/lib/uucp/Devices**
- **/etc/inittab**
- **/usr/lib/uucp/Systems**

Refer to the "Setting Up Mail" in Chapter 4, System Administration, for information on setting up these files.

Physical Connection of Computer to Modem (DCE)

A DCE device such as a modem can connect to your computer with an RS-232 cable. The computer's serial connector must have a DTE configuration and the modem is required to have a DCE configuration. The pin descriptions for modem cable are shown is Figure D-1. The following are the pin connections for the RS-232 modem cable:

Pin 1 to 1
Pin 2 to 2
Pin 3 to 3
Pin 6 to 6
Pin 7 to 7
Pin 8 to 8
Pin 20 to 20.

Wiring for Modems

Wire to pins only used at both ends. Do not attach wiring to unused signal. Seven leads must be wired to customize modem cable as shown in Figure D-4.

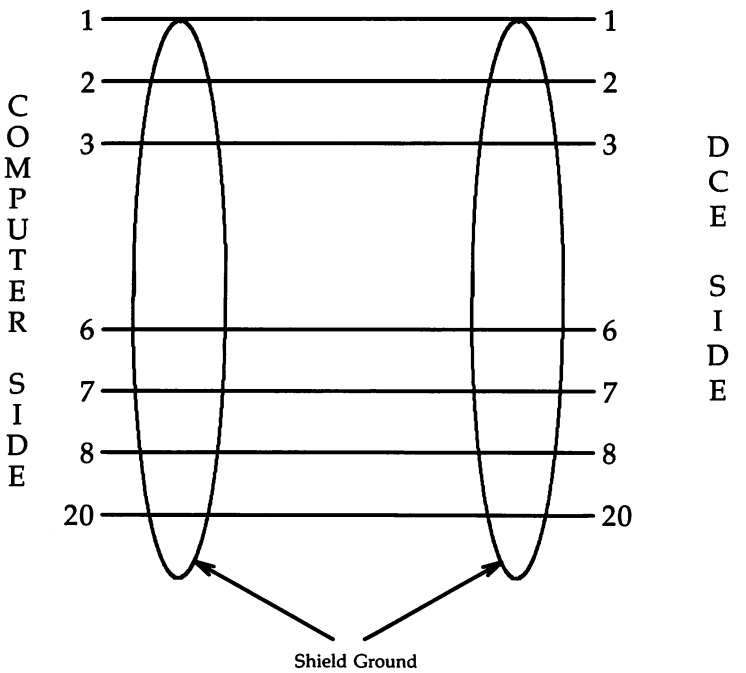
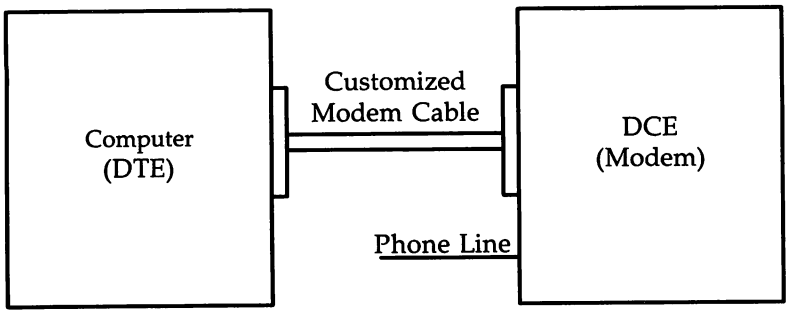


Figure B-4: Connector Wiring Diagram for Connecting Modem to Computer

Figure D-5 shows a simple illustration of how an computer connects to a DCE device such as a modem.



DTE--Data Terminal Equipment

DCE--Data Communication Equipment

Figure B-5: Physical Connection of Computer to DCE Device

Setting Up Modems

Initial Modem Installation

An initial modem setup occurs the first time the modem is installed. The steps that should be followed are:

1. Set up the modem option switches.
2. Make sure the modem power switch is off.
3. Mechanically connect (i.e., attach the cables) the modem to the computer and the telephone line.
4. Plug the modem in.
5. Turn on the power to the modem.
6. Do procedure Serial Port Setup found in Chapter 4, System Administration to set up the serial port.
7. Return to Appendix B, Adding Basic Networking, after you have set up the serial port.

The software commands necessary to configure the modem will now be executed. These commands are sent to the modem, so the modem must be connected to the computer and the power to the modem must be on. In addition, a number of configuration files on the computer must be modified. These steps may take a minute.

In addition to initializing the modem during the Serial Ports Setup, the modem needs to be initialized at boot time. This may add up to 15 seconds to the system boot time.

Reinitializing the Modem

There are circumstances which may require the modem to be reinitialized. This may happen if the modem loses power after the system is booted. If, for example, the modem does not automatically answer calls when it is configured as a Host or Both Caller and Host, if the speaker is active, if the modem dials via pulse dialing, or if it just doesn't seem to be working correctly, it may be necessary to reinstall.

To reinstall, try the following:

1. Disconnect the power to the modem.
2. From the Administration menu, highlight Peripherals Setup and strike Enter.
3. From the Peripherals Setup menu, highlight Serial Ports Setup and strike Enter.
4. From the Serial Ports Setup form, move the cursor to the "Device Type:" field and strike the CHOICES function key.
5. From the Device Types pop-up menu, highlight None and strike Enter.
6. Strike SAVE. You'll return to the Peripherals Setup menu.
7. Reconnect and turn the modem power on.

8. From the Peripherals Setup menu, highlight Serial Ports Setup and strike Enter.
9. With the cursor resting on the "Serial Port Number" field, strike CHOICES.
10. From the Port Number pop-up menu, highlight the serial port number you want and strike Enter.
11. Move the cursor to the "Device Speed:" field and strike CHOICES.
12. From the Device Speed pop-up menu, highlight the device speed you want and strike Enter.
13. Move the cursor to the "Device Type:" field and strike the CHOICES function key.
14. From the Device Types pop-up menu, highlight Modem and strike Enter.
15. Strike the SAVE function key.
16. From the Connect to Modem form, strike the CHOICES function key.
17. From the Modems pop-up menu, highlight the modem name you want and strike Enter.
18. From the Device Connection menu, highlight the correct device connection you want and strike Enter.
19. Strike the SAVE function key.
20. When the confirmation message appears telling you the serial port is now set up for a modem, strike CONT.

These steps are necessary since many modems require options that are set by the software. These steps send the options that are set by the software to the modem when the modem loses them because of a loss of power or other circumstances.

Recommended Switch Settings

The following modems are hardware-configured. They have switch settings that must be manually set in order for the modems to work correctly. Modems that are software-configured will have the switch settings automatically set through the software setup scripts.

Hardware configured modems that have a carrier detect (CD) switch must have that switch set low or off.

AT&T 2212C

- Caller Only

  Internal switches set to factory defaults

- Host Only

  Internal switches set to factory defaults

- Both Host and Caller

  Internal switches set to factory defaults.

AT&T 2224B

- Caller Only

  All switches set to factory defaults

- Host Only

  All switches set to factory defaults

- Both Host and Caller

  All switches set to factory defaults.

NOTE

The volume control for the speaker is located behind the front panel. If the speaker comes on when the modem is in use, flip open the front panel and move the speaker loudness control to its lowest position.

When installing this modem or changing the speed, be sure that the speed indicated on the front panel switch matches the speed set in the menu. The modem will originate calls at the speed indicated in the menu. However, it will only answer calls at the speed indicated by the front panel switch. If the menu and front panel switch do not agree, calls may be lost without there being any other indication of an error.

AT&T 4000 Models 1A01, 1A02, 4024

There are no hardware switches in the standalone (external) AT&T 4000 modems. All options are set in software by the setup scripts. The AT&T 4000 1A01 and 1A02 modems are two different versions of the same modem. Both of these modems are supported by the computer. The version number may be found on the bottom of the modem.

AT&T 4000 Model 4112

When configuring this modem use the COM2 device. The following are the 4 switch settings for the 4112 modem:

1. off
2. off
3. off
4. on Strap CD off.

Hayes SMARTMODEM 1200

- Caller Only

1.	up	Modem disconnects on loss of DTR
2.	up	English result codes
3.	down	Send result codes
4.	up	Echo characters
5.	down	Auto answer disable
6.	up	Strap CD off
7.	up	See manual; up = single line
8.	down	Enable command recognition.

- Host Only

1.	up	Modem disconnects on loss of DTR
2.	up	English results codes
3.	down	Send result codes
4.	up	Echo characters
5.	up	Auto answer enabled
6.	up	Strap CD off
7.	up	See manual; up = single line
8.	down	Enable command recognition.

- Both Host and Caller

1.	up	Modem disconnects on loss of DTR
2.	up	English result codes
3.	down	Send results codes
4.	up	Echo characters
5.	up	Auto answer enabled
6.	up	Strap CD off
7.	up	See manual; up = single line
8.	down	Enable command recognition.

NOTE The Hayes SMARTMODEM 1200 must be reinstalled if the modem loses power after the computer is rebooted.

Hayes SMARTMODEM 2400

There are no option switches on the Hayes SMARTMODEM 2400.

NOTE The Hayes SMARTMODEM 2400 must be reinstalled if the modem loses power after the computer is booted.

Penril 300/1200 AD

- Caller Only
  - A. Front panel switches
    1. The HS button should be struck.
    2. All other buttons should be out.
  - B. The internal switches should all be set to the factory defaults.
- Host Only
  - A. Front panel switches
    1. The HS button should be struck.
    2. All other buttons should be out.
  - B. The internal switches should all be set to the factory defaults.
- Both Host and Caller
  - A. Front panel switches
    1. The HS button should be struck.
    2. All other buttons should be out.
  - B. The internal switches should all be set to factory defaults.

Ventel EC1200-31

There are several modems manufactured by Ventel that are marketed under the model number EC1200-31. These modems can be supported using factory default settings or by setting the Hayes compatible "AT" dialer switch. With either mode you choose, the carrier detect switch (CD) must be set low.

- Caller Only

A. External Switch Settings

1.	open	Use factory default.
2.	open	Use factory default.
3.	open	Use factory default.
4.	open	Use factory default.

B. Internal Switch Settings

1.	open	Modem disconnects on loss of DTR.
2.	close	Strap CD on only when carrier is present.
3.	open	Enables Ventel compatible command recognition.
4.	open	Speaker off.

- Host Only

A. External Switch Settings

1.	open	Use factory default.
2.	open	Use factory default.
3.	open	Use factory default.
4.	open	Use factory default.

B. Internal Switch Settings

1.	open	Modem disconnects on loss of DTR.
2.	close	Strap CD off.
3.	open	Enables Ventel compatible command recognition.
4.	open	Speaker off.

- Both Host and Caller

A. External Switch Settings

1.	open	Use factory default.
2.	open	Use factory default.
3.	open	Use factory default.
4.	open	Use factory default.

B. Internal Switch Settings

1.	open	Modem disconnects on loss of DTR.
2.	close	Strap CD off. The light will come on only when a valid carrier is detected by the modem.
3.	close	Hayes compatible AT dialer enabled. Modem assumes Hayes verbose on power up.
4.	open	Speaker off.

NOTE

When the Ventel EC1200-31 is powered on, the MB/HO light will sometimes stay on. The modem should be powered off then on until this light is out.

Ventel 1200-EC

- Caller Only

A. External Switch Settings

1.	open	English result codes
2.	open	Send result codes
3.	open	Echo characters
4.	close	Don't auto answer.

B. Internal Switch Settings

1.	open	Modem disconnects on loss of DTR.
2.	open	Strap CD on only when carrier is present.
3.	close	Enables Hayes compatible command recognition.
4.	open	Speaker off.

- Host Only

A. External Switch settings

1.	open	Don't care - use factory default.
2.	close	Don't send result codes.
3.	close	Don't echo character.
4.	open	Auto answer.

B. Internal Switch Settings

1.	open	Modem disconnects on loss of DTR.
2.	open	CD on only when carrier is present.
3.	close	Don't care - use factory default.
4.	open	Speaker off.

C. Both Host and Caller

D. External Switch Settings

1.	open	English result codes
2.	open	Send result codes
3.	open	Echo characters
4.	open	Auto answer.

E. Internal Switch Settings

1	open	Modem disconnects on loss of DTR.
2.	open	Strap CD on only when carrier is present.
3.	close	Enables Hayes compatible command recognition.
4.	open	Speaker off.

NOTE

When the Ventel 1200-EC is powered on, the MB/OH light will sometimes stay on. The modem should be powered off then on until this light is out.

Ventel MD212

- Caller Only
    A. All front panel switches should be out.
    B. All internal switches should be set to the factory defaults.
- Host Only
    A. All front panel switches should be out.
    B. All internal switches should be set to the factory defaults.
- Both Host and Caller
    A. All front panel switches should be out.
    B. All internal switches should be set to the factory defaults.

Appendix C: fsck Error Messages

Appendix C: fsck Error Messages

This appendix describes the error messages you might receive when using **fsck**.

Initialization

Before a file system check can be performed, certain tables have to be set up and certain files opened. This section describes the opening of files and the initialization of tables. Error conditions resulting from command line options, memory requests, opening of files, status of files, file system size checks, and creation of the scratch file are listed below. The **fsck** program terminates on initialization errors.

Legal Options

Legal **fsck** options are **-f**, **-b**, **-y**, **-n**, **-s**, **-S**, **-t**, **-q**, **-b**, and **-D**. The **-y** option is recommended for **fsck**. This option will answer yes to all questions prompted by **fsck** and requires no intervention by you. Another recommended option is **-s** which forces rebuilding the free list in optimal order. The free list gets disorganized with use. Rebuilding the free list improves performance on subsequently created files. Use the following command line for **fsck**:

/etc/fsck -s -y *file_system_name*

See the *fsck*(1M) page in the *User's/System Administrator's Reference Manual* for additional information.

Error Messages

```
Bad -t option
```

The **-t** option is not followed by a filename. The **fsck** program terminates on this error condition.

```
Invalid -s argument, defaults assumed
```

The **-s** option is not suffixed by 3, 4, or blocks-per-cylinder:blocks-to-skip. The **fsck** program assumes a default of 400 blocks-per-cylinder and 7 blocks-to-skip.

`Incompatible options: -n and -s`

It is not possible to salvage the free-block list without modifying the file system. The **fsck** program terminates on this error condition.

`Cannot fstat standard input`

The attempt to **fstat** standard input failed. This error condition indicates a serious problem that may require additional assistance. The **fsck** program terminates on this error condition.

`Cannot get memory`

The request for memory for virtual memory tables failed. This error condition indicates a serious problem that may require additional assistance. The **fsck** program terminates on this error condition.

`Cannot open checklist file: F`

The default file system *checklist* file *F* (usually */etc/checklist*) cannot be opened for reading. The **fsck** program terminates on this error condition. Check access modes of *F*.

`Cannot stat root`

The request for statistics about the root directory `"/"` failed. This error condition indicates a serious problem that may require additional assistance. The **fsck** program terminates on this error condition.

`Cannot stat F`

The request for statistics about the file system *F* failed. The **fsck** program ignores this file system and continues checking the next file system given. Check access modes of *F*.

`F is not a block or character device`

The **fsck** program has been given a regular filename by mistake. It ignores this argument and continues checking the next file system given. Check the file type of *F*.

`Cannot open F`

The file system *F* cannot be opened for reading. The **fsck** program ignores this and continues checking the next file system given. Check the access modes of *F*.

`Size check: fsize X isize Y`

More blocks are used for the i-node list *Y* than there are blocks in the file system *X*, or there are more than 65,535 i-nodes in the file system. The **fsck** program ignores this file system and continues checking the next file system given.

`Cannot create F`

The request to create a scratch file *F* failed. The **fsck** program ignores this file system and continues checking the next file system given. Check the access modes of *F*.

`CAN NOT SEEK: BLK B (CONTINUE)`

The request for moving to a specified block number *B* in the file system failed. The occurrence of this error condition indicates a serious problem that may require additional assistance.

Possible responses to the `CONTINUE` prompt are

YES — Attempt to continue to run file system check. Often, however, the problem persists. This error condition does not allow a complete check of the file system. A second run of **fsck** should be made to recheck this file system. If the block was part of the virtual memory buffer cache, **fsck** will terminate with the message "Fatal I/O error."

NO — Terminate program.

```
CAN NOT READ: BLK B (CONTINUE)
```

The request for reading a specified block number *B* in the file system failed. The occurrence of this error condition indicates a serious problem that may require additional assistance.

Possible responses to the `CONTINUE` prompt are

YES — Attempt to continue to run file system check. Often, however, the problem persists. This error condition does not allow a complete check of the file system. A second run of **fsck** should be made to recheck this file system. If block was part of the virtual memory buffer cache, **fsck** will terminate with the message "Fatal I/O error."

NO — Terminate program.

`CAN NOT WRITE: BLK B (CONTINUE)`

The request for writing a specified block number *B* in the file system failed. The file system should not be opened for writing.

Possible responses to the `CONTINUE` prompt are

YES — Attempt to continue to run file system check. Often, however, the problem persists. This error condition does not allow a complete check of the file system. A second run of **fsck** should be made to recheck this file system. If block was part of the virtual memory buffer cache, **fsck** terminates with the message "Fatal I/O error."

NO — Terminate program.

Phase 1: Check Blocks and Sizes

This phase concerns itself with the i-node list. This part lists error conditions resulting from checking i-node types, setting up the zero-link-count table, examining i-node block numbers for bad or duplicate blocks, checking i-node size, and checking i-node format.

UNKNOWN FILE TYPE I=I (CLEAR)

The mode word of the i-node *I* indicates that the i-node is not a special character i-node, regular i-node, or directory i-node.

Possible responses to the `CLEAR` prompt are

YES — Deallocate i-node *I* by zeroing its contents. This invokes the UNALLOCATED error condition in Phase 2 for each directory entry pointing to this i-node.

NO — Ignore this error condition.

LINK COUNT TABLE OVERFLOW (CONTINUE)

An internal table for **fsck** containing allocated i-nodes with a link count of zero has no more room.

Possible responses to the `CONTINUE` prompt are

YES — Continue with program. This error condition does not allow a complete check of the file system. A system run of **fsck** should be made to recheck this file system. If another allocated i-node with a zero link count is found, this error condition will be repeated.

NO — Terminate the program.

B BAD I=I

I-node *I* contains block number *B* with a number lower than the number of the first data block in the file system or greater than the number of the last block in the file system. This error condition may invoke the EXCESSIVE BAD BLKS error condition in Phase 1 if i-node *I* has too many block numbers outside the file system range. This error condition invokes the BAD/DUP error condition in Phase 2 and Phase 4.

EXCESSIVE BAD BLKS I=I (CONTINUE)

There is more than a tolerable number (usually 10) of blocks claimed by other i-nodes.

Possible responses to the `CONTINUE` prompt are

YES — Ignore the rest of the blocks in this i-node and continue to check using the next i-node in the file system. This error condition does not allow a complete check of the file system. A second run of **fsck** should be made to recheck this file system.

NO — Terminate the program.

B DUP I=I

I-node *I* contains block number *B* that is already claimed by another i-node. This error condition may invoke the EXCESSIVE DUP BLKS error condition in Phase 1 if i-node *I* has too many block numbers claimed by other i-nodes. This error condition invokes Phase 1B and the BAD/DUP error condition in Phase 2 and Phase 4.

EXCESSIVE DUPS BLKS I=I (CONTINUE)

There is more than a tolerable number (usually 10) of blocks claimed by other i-nodes.

Possible responses to the `CONTINUE` prompt are

YES — Ignore the rest of the blocks in this i-node and continue to check using the next i-node in the file system. This error condition does not allow a complete check of the file system. A second run of **fsck** should be made to recheck this file system.

NO — Terminate the program.

DUP TABLE OVERFLOW (CONTINUE)

An internal table in **fsck** containing duplicate block numbers has no more room.

Possible responses to the `CONTINUE` prompt are

YES	Continue with program. This error condition does not allow a complete check of the file system. A second run of **fsck** should be made to recheck this file system. If another duplicate block is found, this error condition will repeat.
NO	Terminate the program.

POSSIBLE FILE SIZE ERROR I=I

The i-node *I* size does not match the actual number of blocks used by the i-node. This is only a warning. If the **-q** option is used, this message will not print.

DIRECTORY MISALIGNED I=I

The size of a directory i-node is not a multiple of the size of a directory entry (usually 16). This is only a warning. If the **-q** option is used, this message will not print.

PARTIALLY ALLOCATED INODE I=I (CLEAR)

I-node *I* is neither allocated nor unallocated.

Possible responses to the `CLEAR` prompt are

YES	Deallocate i-node *I* by zeroing its contents.
NO	Ignore this error condition.

Phase 1B: Rescan for More DUPS

When a duplicate block is found in the file system, the file system is rescanned to find the i-node that previously claimed that block. This part lists the error condition when the duplicate block is found.

B DUP I=I

I-node *I* contains block number *B* that is already claimed by another i-node. This error condition invokes the BAD/DUP error condition in Phase 2. I-nodes with overlapping blocks may be determined by examining this error condition and the DUP error condition in Phase 1.

Phase 2: Check Path Names

This phase concerns itself with removing directory entries pointing to i-nodes from Phase 1 and Phase 1B that have error conditions. This part lists error conditions resulting from root i-node mode and status, directory i-node pointers in range, and directory entries pointing to bad i-nodes.

ROOT INODE UNALLOCATED. TERMINATING

The root i-node (always i-node number 2) has no allocated mode bits. The occurrence of this error condition indicates a serious problem that may require additional assistance. The program stops.

ROOT INODE NOT DIRECTORY (FIX)

The root i-node (usually i-node number 2) is not directory i-node type.

Possible responses to the `FIX` prompt are

YES	Replace the root i-node type to be a directory. If the root i-node data blocks are not directory blocks, a large number of error conditions will be produced.
NO	Terminate the program.

DUPS/BAD IN ROOT INODE (CONTINUE)

Phase 1 or Phase 1B has found duplicate blocks or bad blocks in the root i-node (usually i-node number 2) for the file system.

Possible responses to the `CONTINUE` prompt are

YES — Ignore DUPS/BAD error condition in root i-node and attempt to continue to run the file system check. If root i-node is not correct, then this may result in a large number of other error conditions.

NO — Terminate the program.

I OUT OF RANGE I=I NAME=F (REMOVE)

A directory entry *F* has an i-node number *I* that is greater than the end of the i-node list.

Possible responses to the `REMOVE` prompt are

YES The directory entry *F* is removed.

NO Ignore this error condition.

UNALLOCATED I=I OWNER=O MODE=M SIZE=S MTIME=T NAME=F (REMOVE)

A directory entry *F* has an i-node *I* without allocate mode bits. The owner *0*, mode *M*, size *S*, modify time *T*, and filename *F* are printed. If the file system is not mounted and the **-n** option is not specified, the entry will be removed automatically if the i-node it points to is size 0.

Possible responses to the `REMOVE` prompt are

YES The directory entry *F* is removed.

NO Ignore this error condition.

DUP/BAD I=I OWNER=O MODE=M SIZE=S MTIME=T DIR=F (REMOVE)

Phase 1 or Phase 1B has found duplicate blocks or bad blocks associated with directory entry *F*, i-node *I*. The owner *O*, mode *M*, size *S*, modify time *T*, and filename *F* are printed.

Possible responses to the `REMOVE` prompt are

YES The directory entry *F* is removed.

NO Ignore this error condition.

DUP/BAD I=I OWNER=O MODE=M SIZE=S MTIME=T FILE=F (REMOVE)

Phase 1 or Phase 1B has found duplicate blocks or bad blocks associated with directory entry *F*, i-node *I*. The owner *O*, mode *M*, size *S*, modify time *T*, and filename *F* are printed.

Possible responses to the `REMOVE` prompt are

YES The directory entry *F* is removed.

NO Ignore this error condition.

BAD BLK B IN DIR I=I OWNER=O MODE=M SIZE=S MTIME=T

This message only occurs when the **-q** option is used. A bad block was found in DIR i-node *I*. Error conditions looked for in directory blocks are nonzero padded entries, inconsistent "." and ".." entries, and embedded slashes in the name field. This error message indicates that the user should at a later time either remove the directory i-node if the entire block looks bad or change (or remove) those directory entries that look bad.

Phase 3: Check Connectivity

This phase concerns itself with the directory connectivity seen in Phase 2. This part lists error conditions resulting from unreferenced directories and missing or full *lost+found* directories.

UNREF DIR I=I OWNER=O MODE=M SIZE=S MTIME=T (RECONNECT)

The directory i-node *I* was not connected to a directory entry when the file system was traversed. The owner *O*, mode *M*, size *S*, and modify time *T* of directory i-node *I* are printed. The **fsck** program forces the reconnection of a nonempty directory.

Possible responses to the `RECONNECT` prompt are

YES — Reconnect directory i-node *I* to the file system in directory for lost files (usually *lost+found*). This may invoke *lost+found* error condition in Phase 3 if there are problems connecting directory i-node *I* to *lost+found*. This may also invoke CONNECTED error condition in Phase 3 if link was successful.

NO — Ignore this error condition. This invokes UNREF error condition in Phase 4.

SORRY, NO lost+found DIRECTORY

There is no *lost+found* directory in the root directory of the file system; **fsck** ignores the request to link a directory in *lost+found*. This invokes the UNREF error condition in Phase 4. Check access modes of *lost+found*.

SORRY, NO SPACE IN lost+found DIRECTORY

There is no space to add another entry to the *lost+found* directory in the root directory of the file system; **fsck** ignores the request to link a directory in *lost+found*. This invokes the UNREF error condition in Phase 4. Clean out unnecessary entries in *lost+found* or make *lost+found* larger.

DIR I=I1 CONNECTED, PARENT WAS I=I2

This is an advisory message indicating a directory i-node *I1* was successfully connected to the *lost+found* directory. The parent i-node *I2* of the directory i-node *I1* is replaced by the i-node number of the *lost+found* directory.

Phase 4: Check Reference Counts

This phase concerns itself with the link count information seen in Phase 2 and Phase 3. This part lists error conditions resulting from unreferenced files; a missing or full *lost+found* directory; an incorrect link count for files, directories, or special files; unreferenced files and directories; bad and duplicate blocks in files and directories; and incorrect total free-i-node counts.

UNREF FILE I=I OWNER=O MODE=M SIZE=S MTIME=T (RECONNECT)

I-node *I* was not connected to a directory entry when the file system was traversed. The owner *O*, mode *M*, size *S*, and modify time *T* of i-node *I* are printed. If the **-n** option is omitted and the file system is not mounted, empty files will be cleared automatically. Nonempty directories are not cleared.

Possible responses to the RECONNECT prompt are

YES — Reconnect i-node *I* to file system in the directory for lost files (usually *lost+found*). This can cause a *lost+found* error condition in Phase 4 if there are problems connecting i-node *I* to *lost+found*.

NO — Ignore this error condition. This invokes a CLEAR error condition in Phase 4.

SORRY. NO lost+found DIRECTORY

There is no *lost+found* directory in the root directory of the file system; **fsck** ignores the request to link a file in *lost+found*. This invokes the CLEAR error condition in Phase 4. Check access modes of *lost+found*.

SORRY. NO SPACE IN lost+found DIRECTORY

There is no space to add another entry to the *lost+found* directory in the root directory of the file system; **fsck** ignores the request to link a file in *lost+found*. This invokes the CLEAR error condition in Phase 4. Check size and contents of *lost+found*.

(CLEAR)

The i-node mentioned in the immediately previous error condition cannot be reconnected.

Possible responses to the `CLEAR` prompt are

YES	Deallocate i-node mentioned in the immediately previous error condition by zeroing its contents.
NO	Ignore this error condition.

LINK COUNT FILE I=I OWNER=O MODE=M SIZE=S MTIME=T COUNT=X SHOULD BE Y (ADJUST)

The link count for i-node *I*, which is a file, is *X* but should be *Y*. The owner *O*, mode *M*, size *S*, and modify time *T* are printed.

Possible responses to the `ADJUST` prompt are

YES Replace link count of file i-node *I* with *Y*.

NO Ignore this error condition.

LINK COUNT DIR I=I OWNER=O MODE=M SIZE=S MTIME=T COUNT=X SHOULD BE Y (ADJUST)

The link count for i-node *I*, which is a directory, is *X* but should be *Y*. The owner *O*, mode *M*, size *S*, and modify time *T* of directory i-node *I* are printed.

Possible responses to the `ADJUST` prompt are

YES Replace link count of directory i-node *I* with *Y*.

NO Ignore this error condition.

LINK COUNT F I=I OWNER=O MODE=M SIZE=S MTIME=T COUNT=X SHOULD BE Y (ADJUST)

The link count of *F* i-node *I* is *X* but should be *Y*. The filename *F*, owner *O*, mode *M*, size *S*, and modify time *T* are printed.

Possible responses to the `ADJUST` prompt are

YES Replace link count of i-node *I* with *Y*.

NO Ignore this error condition.

UNREF FILE I=I OWNER=O MODE=M SIZE=S MTIME=T (CLEAR)

I-node *I*, which is a file, was not connected to a directory entry when the file system was traversed. The owner *O*, mode *M*, size *S*, and modify time *T* of i-node *I* are printed. If the **-n** option is omitted and the file system is not mounted, empty files will be cleared automatically. Nonempty directories are not cleared.

Possible responses to the `CLEAR` prompt are

YES Deallocate i-node *I* by zeroing its contents.

NO Ignore this error condition.

UNREF DIR I=I OWNER=O MODE=M SIZE=S MTIME=T (CLEAR)

I-node *I*, which is a directory, was not connected to a directory entry when the file system was traversed. The owner *O*, mode *M*, size *S*, and modify time *T* of i-node *I* are printed. If the **-n** option is omitted and the file system is not mounted, empty files will be cleared automatically. Nonempty directories are not cleared.

Possible responses to the CLEAR prompt are

YES Deallocate i-node *I* by zeroing its contents.

NO Ignore this error condition.

BAD/DUP FILE I=I OWNER=O MODE=M SIZE=S MTIME=T (CLEAR)

Phase 1 or Phase 1B has found duplicate blocks or bad blocks associated with the file i-node *I*. The owner *O*, mode *M*, size *S*, and modify time *T* of i-node *I* are printed.

Possible responses to the CLEAR prompt are

YES Deallocate i-node *I* by zeroing its contents.

NO Ignore this error condition.

BAD/DUP DIR I=I OWNER=O MODE=M SIZE=S MTIME=T (CLEAR)

Phase 1 or Phase 1B has found duplicate blocks or bad blocks associated with directory i-node *I*. The owner *O*, mode *M*, size *S*, and modify time *T* of i-node *I* are printed.

Possible responses to the `CLEAR` prompt are

YES Deallocate i-node *I* by zeroing its contents.

NO Ignore this error condition.

FREE INODE COUNT WRONG IN SUPERBLK (FIX)

The actual count of the free i-nodes does not match the count in the super-block of the file system. If the **-q** option is specified, the count will be fixed automatically in the super-block.

Possible responses to the `FIX` prompt are

YES Replace count in super-block by actual count.

NO Ignore this error condition.

Phase 5: Check Free List

This phase concerns itself with the free-block list. This part lists error conditions resulting from bad blocks in the free-block list, bad free-block count, duplicate blocks in the free-block list, unused blocks from the file system not in the free-block list, and the total free-block count incorrect.

EXCESSIVE BAD BLKS IN FREE LIST (CONTINUE)

The free-block list contains more than a tolerable number (usually 10) of blocks with a value less than the first data block in the file system or greater than the last block in the file system.

Possible responses to the `CONTINUE` prompt are

YES	Ignore the rest of the free-block list and continue execution of **fsck**. This error condition will always invoke "BAD BLKS IN FREE LIST" error condition in Phase 5.
NO	Terminate the program.

EXCESSIVE DUP BLKS IN FREE LIST (CONTINUE)

The free-block list contains more than a tolerable number (usually 10) of blocks claimed by i-nodes or earlier parts of the free-block list.

Possible responses to the `CONTINUE` prompt are

YES — Ignore the rest of the free-block list and continue execution of **fsck**. This error condition will always invoke "DUP BLKS IN FREE LIST" error condition in Phase 5.

NO — Terminate the program.

BAD FREEBLK COUNT

The count of free blocks in a free-list block is greater than 50 or less than 0. This error condition will always invoke the "BAD FREE LIST" condition in Phase 5.

X BAD BLKS IN FREE LIST

X blocks in the free-block list have a block number less than the first data block in the file system or greater than the last block in the file system. This error condition will always invoke the "BAD FREE LIST" condition in Phase 5.

X DUP BLKS IN FREE LIST

X blocks claimed by i-nodes or earlier parts of the free-list block were found in the free-block list. This error condition will always invoke the "BAD FREE LIST" condition in Phase 5.

X BLK(S) MISSING

X blocks unused by the file system were not found in the free-block list. This error condition will always invoke the "BAD FREE LIST" condition in Phase 5.

FREE BLK COUNT WRONG IN SUPERBLOCK (FIX)

The actual count of free blocks does not match the count in the super-block of the file system.

Possible responses to the `FIX` prompt are

YES	Replace count in super-block by actual count.
NO	Ignore this error condition.

BAD FREE LIST (SALVAGE)

Phase 5 has found bad blocks in the free-block list, duplicate blocks in the free-block list, or blocks missing from the file system. If the **-q** option is specified, the free-block list will be salvaged automatically.

Possible responses to the SALVAGE prompt are

YES	Replace actual free-block list with a new free-block list. The new free-block list will be ordered to reduce the time spent by the disk rotating into position.
NO	Ignore this error condition.

Phase 6: Salvage Free List

This phase concerns itself with the free-block list reconstruction. This part lists error conditions resulting from the blocks-to-skip and blocks-per cylinder values.

Default free-block list spacing assumed

This is an advisory message indicating that blocks-to-skip (gap size) is greater than blocks-per-cylinder, blocks-to-skip is less than 1, blocks-per-cylinder is less than 1, or blocks-per-cylinder is greater than 500. The default values of 7 blocks-to-skip and 400 blocks-per-cylinder are used. These values were set previously when the **mkfs** (make file system) command was used to make the file system.

Glossary

This glossary defines terms and acronyms used in this document that may not be familiar to you.

Absolute Pathname — The pathname that is used to specify a command or program from the *root* directory.

Application — The software designed to perform a particular kind of work. For example, you use the Word Processing application to create and edit documents you want to print.

Backup — A spare copy of data or software that you keep in case the original is damaged or lost.

Bad Track — A part of the hard disk known as a track that is not usable.

Character — A letter, number, or symbol.

Command — An instruction used to tell the computer to perform a function or carry out an activity.

Configuration — The way that the computer is set up to allow for particular uses or situations.

Copy — To duplicate information.

Daemon — A program that runs as a background process to handle UNIX System activities, e.g., file transfers, command executions, cleanup routines, etc.

Default — A value that the computer uses if you do not specify a value.

Delete — To remove, erase, or discard data.

Encrypt — To make a file unreadable by anyone who does not know the password to the file.

Error Message — A response from a program indicating that a problem has arisen or something unexpected has happened, requiring your attention.

Floppy Disk Drive	A device that reads and writes information on a floppy diskette.
Format	(1) To prepare a new floppy diskette or hard disk for use with the computer. (2) The way data is displayed. Pertains to the way the data appears on your screen or printed copy.
Hard Disk	A device that stores operating systems, programs, and data files.
Install	Involves the procedures used to set up the hardware and software of a computer so that it can be used. Installing often includes customizing the system for a particular situation or user.
Meta Character	A set of characters the UNIX System shell interprets as having a special meaning.
Modem	A device that modulates and demodulates data transmitted over communication lines.
Operating System	The software that controls and allocates the resources, such as memory, disk storage, and the screen display for the computer.
Option	An addition to a command to improve or provide an extra enhancement to the command. The option is usually depicted with a minus (-) sign in front of it.
Partition	A section of the hard disk that is used to store an operating system and data files or programs. By dividing the disk into partitions, you can use the space allocated in a more efficient and organized manner.
Printer	A machine that prints information transferred from a computer.
Printer Interface Program	A program used to "set the printer up" for a print request. Each printer has an interface program.

Program	A set of step-by-step instructions that tells a computer how to do a particular task.
Relative Pathname	The pathname that is used to specify a command or program from the current directory.
Software	Computer programs that have been stored on a disk or other media.
Spooling	The term "spool" is an acronym for **s**imultaneous **p**eripheral **o**perations **o**n-**l**ine. The line printer spooling system allows you to send a file to be printed while you continue with other work.
Syntax	The format of a command line.
System	A general term for a computer and its software and data.
UNIX System	A general-purpose, multi-user, interactive, time-sharing operating system used with your computer.
Utilities	A group of programs combined into a package that represent a specific application available with your computer.
Utility	A program, usually from a set of programs, that represents a specific application available with your computer.
Write-Protect Notch	A rectangular cutout on one edge of a floppy disk. If this notch is covered with a piece of special tape that comes with the floppy diskette, new information cannot be written on the disk. Data on the floppy diskette cannot be altered.

Index

A

B

C

D

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

V

W

X

ORDER FORM

QUANTITY	TITLE/AUTHOR	TITLE CODES	PRICE		TOTAL
_____	**1.** Portability Gde., 3/E, 7 volumes, X/OPEN	68581-8	$130.00g	paper	_____
_____	**2.** Portability Gde., 3/E: Sys. V Spec. Commands & Util., Vol. 1, X/OPEN	68583-4	$30.00g	paper	_____
_____	**3.** Portability Gde., 3/E: Sys. V Spec. Calls & Libraries, Vol. 2, X/OPEN	68584-2	$30.00g	paper	_____
_____	**4.** Portability Gde., 3/E: Sys. V Spec. Sup. Defns., Vol. 3, X/OPEN	68585-9	$30.00g	paper	_____
_____	**5.** Portability Gde., 3/E: Prog. Lang., Vol. 4, X/OPEN	68586-7	$30.00g	paper	_____
_____	**6.** Portability Gde., 3/E: Data Mgt., Vol. 5, X/OPEN	68587-5	$30.00g	paper	_____
_____	**7.** Portability Gde., 3/E: Networking, Vol. 6, X/OPEN	68588-3	$30.00g	paper	_____
_____	**8.** Portability Gde., 3/E: Operating Sys., Vol. 7, X/OPEN	68589-1	$30.00g	paper	_____
_____	**9.** UNIX® Sys. V/386 Prog.'s Guide, AT&T	94091-6	$34.95g	paper	_____
_____	**10.** UNIX® Sys. V/386 STREAMS Primer, AT&T	94087-4	$21.95g	paper	_____
_____	**11.** UNIX® Sys. V/386 STREAMS Prog.'s Guide, AT&T	94088-2	$24.95g	paper	_____
_____	**12.** UNIX® Sys. V/386 Network Prog.'s Guide, AT&T	94085-8	$24.95g	paper	_____
_____	**13.** UNIX® Sys. V/386 Prog.'s Ref. Manual, AT&T	94086-6	$34.95g	paper	_____
_____	**14.** UNIX® Sys. V/386 User's Ref. Manual, AT&T	94093-2	$34.95g	paper	_____
_____	**15.** UNIX® Sys. V/386 User's Guide, 2/E, AT&T	94092-4	$24.95g	paper	_____
_____	**16.** UNIX® Sys. V/386 Utilities Release Notes, AT&T	93612-0	$21.95g	paper	_____
_____	**17.** UNIX® Sys. V/386 Sys. Admin. Guide, AT&T	94089-0	$24.95g	paper	_____
_____	**18.** UNIX® Sys. V/386 Sys. Admin. Ref. Manual, AT&T	94090-8	$24.95g	paper	_____
_____	**19.** UNIX® Sys. V Prog.'s Guide, AT&T	94043-7	$36.95g	paper	_____
_____	**20.** UNIX® Sys. V STREAMS Primer, AT&T	94052-8	$21.95g	paper	_____
_____	**21.** UNIX® Sys. V STREAMS Prog.'s Guide, AT&T	94053-6	$24.95g	paper	_____
_____	**22.** UNIX® Sys. V Network Prog.'s Guide, AT&T	94046-0	$24.95g	paper	_____
_____	**23.** UNIX® Sys. V Prog.'s Ref. Manual, AT&T	94047-8	$36.95g	paper	_____

_____	**24.** UNIX® Sys. V User's Ref. Manual, AT&T	94048-6	$34.95g	paper	_____
_____	**25.** UNIX® Sys. V User's Guide, 2/E, AT&T	94054-4	$25.95g	paper	_____
_____	**26.** UNIX® Sys. V Utilities Release Notes, AT&T	94055-1	$21.95g	paper	_____
_____	**27.** UNIX® Sys. V Sys. Admin. Guide, AT&T	93613-8	$34.95g	paper	_____
_____	**28.** UNIX® Sys. V Sys. Admin. Ref. Manual, AT&T	93614-6	$24.95g	paper	_____
_____	**29.** The C Programming Language, 2/E, Kernighan/Ritchie	11037-9 11036-1	$40.00sf $28.00sf	cloth paper	_____ _____
_____	**30.** The C Programming Language, 1/E, Kernighan/Ritchie	11016-3	$27.00sf	paper	_____
_____	**31.** The C Answer Book, 2/E, Tondo/Gimpel	10965-2	$21.33sf	paper	_____
_____	**32.** The C Answer Book, 1/E, Tondo/Gimpel	10987-6	$21.00sf	paper	_____
_____	**33.** ANSI C: A Lexical Guide, Mark Williams Co.	03781-2	$35.00sf	paper	_____
_____	**34.** Doing Business with C, Swartz	21725-7	$26.95g	paper	_____
_____	**35.** Advanced C Programming, Rochkind	01024-9	$32.95g	paper	_____
_____	**36.** C Trainer, Feuer	10974-4	$24.33sf	paper	_____
_____	**37.** C: A Reference Manual, 2/E, Harbison et al.	10980-1	$25.95g	paper	_____
_____	**38.** C Companion, Holub	10978-3	$22.67sf	paper	_____
_____	**39.** Programming in C with a Bit of UNIX®, Moore	73009-3	$25.95g	paper	_____
_____	**40.** Learning To Program in C, Plum	52784-6	$33.00sf	paper	_____
_____	**41.** C Notes, Zahn	10977-7	$17.95g	paper	_____
_____	**42.** C Prog. in Berkeley UNIX® Envirmnt., Horspool	10997-5	$27.00sf	paper	_____
_____	**43.** C Prog.'s Handbook, Bolsky	11007-2	$22.95g	paper	_____
_____	**44.** Crafting C Tools, Campbell	18841-7	$25.95g	paper	_____
_____	**45.** C Puzzle Book, Feuer	10992-6	$25.00sf	paper	_____
_____	**46.** Numerical Software Tools in C, Kempf	62727-3	$28.00sf	paper	_____
_____	**47.** C Programming Guidelines, Plum	10999-1	$34.00sf	paper	_____
_____	**48.** A Software Tools Sampler, Miller	82230-4	$26.67sf	paper	_____
_____	**49.** Systems Software Tools, Biggerstaff	88176-3	$19.95g	paper	_____
_____	**50.** UNIX® Relational Database Mgt., Manis et al.	93862-1	$34.00sf	paper	_____
_____	**51.** UNIX® Prog. Envirmnt., Kernighan/Pike	93768-0	$24.95g	paper	_____
_____	**52.** Advanced UNIX® Prog., Rochkind	01180-9	$29.95g	paper	_____

_____	**53.** Portable C & UNIX® Sys. Prog., Lapin	68649-3	$24.95g	paper	_____
_____	**54.** UNIX® Sys. Software Readings, AT&T UNIX® Pacific Co.	93835-7	$21.95g	paper	_____
_____	**55.** UNIX® Sys. Readings & Applications, Vol. I, AT&T	93853-0	$19.00sf	paper	_____
_____	**56.** UNIX® Sys. Readings & Applications, Vol. II, AT&T	93984-3	$19.00sf	paper	_____
_____	**57.** *vi* User's Handbook, Bolsky	94173-2	$18.95g	paper	_____
_____	**58.** Guide to *vi*, Sonnenschein	37131-0	$19.95g	paper	_____
_____	**59.** *Troff* Typesetting, Emerson et al.	93095-8	$27.95g	paper	_____
_____	**60.** Intro. to Compiler Construction, Schreiner et al.	47439-5	$38.00sf	cloth	_____
_____	**61.** UNIX® C Shell Field Gde., Anderson et al.	93746-6	$27.95g	paper	_____
_____	**62.** Preparing Documents with UNIX®, Brown et al.	69997-5	$27.95sf	cloth	_____
_____	**63.** Oper. Sys. Des. & Implementation, Tanenbaum	63740-5	$42.00sf	cloth	_____
	MINIX for the IBM PC/XT/AT:				
	1) 512K for the AT	58441-7	$110.00sf	software	_____
	2) 640K for the PC/PC XT	58442-5	$110.00sf	software	_____
	3) MINIX for the IBM PC/XT/AT Ref. Manual	58440-9	$32.00sf	paper	_____
_____	**64.** Operating Sys. Design, Vol. I: XINU Approach (PC Ed.), Comer/Fossum	63818-9	$44.00sf	cloth	_____
_____	**65.** Operating Sys. Design, Vol. I: XINU Approach, Comer	63753-8	$46.00sf	cloth	_____
_____	**66.** Design of UNIX® O/S, Bach	20179-8	$20.00sf	paper	_____
_____	**67.** Oper. Sys. Des., Vol. II: Internetworking with XINU, Comer	63741-3	$46.00sf	cloth	_____
_____	**68.** Internetworking with TCP/IP, Comer	47015-3	$36.00sf	cloth	_____
_____	**69.** UNIX® Sys. V Network Prog.'s Guide, AT&T	94046-0	$24.95g	paper	_____
_____	**70.** UNIX® Admin. Guide for Sys. V, Thomas/Farrow	94288-8	$34.95g	paper	_____
_____	**71.** UNIX® Sys. V Sys. Admin. Guide, AT&T	93613-8	$34.95g	paper	_____
_____	**72.** UNIX® Sys. V Sys. Admin. Ref. Manual, AT&T	93614-6	$24.95g	paper	_____
_____	**73.** UNIX® Sys. User's Handbook, Bolsky	93776-3	$18.95g	paper	_____
_____	**74.** Making Use of the UNIX® O/S, Budgen	R4434-8	$24.00sf	paper	_____

_____	**75.** UNIX® for People, Birns et al.	93744-1	$28.00g	paper	_____
_____	**76.** UNIX® Primer, Lomuto et al.	93773-0	$28.95g	paper	_____
_____	**77.** UNIX® RefGuide, McNulty	93895-1	$27.95g	paper	_____
_____	**78.** DOS: UNIX® Systems, Seyer/Mills	21864-4	$27.95g	paper	_____
_____	**79.** Beyond Photography: The Digital Darkroom, Holzmann	07441-9	$26.95g	paper	_____
	Digital Darkroom Software, Holzmann	21274-6	$25.00g	software	_____
_____	**80.** Clipper™ 32-Bit Microproc. User's Manual, Fairchild	13805-7	$23.95g	paper	_____
_____	**81.** Programmer's Survival Guide, Ruhl	73037-4	$16.95g	paper	_____

SPECIAL OFFER!
When ordering 3 or more copies (of the same or different titles) take 10% off the total list price. When ordering 5 or more copies (of the same or different titles) take 15% off the total list price.
SAVE!
If payment accompanies order, plus your state's sales tax where applicable, Prentice Hall pays postage and handling charges. Same return privilege refund guaranteed.

☐ PAYMENT ENCLOSED–shipping and handling to be paid by publisher (please include your state's sales tax where applicable).
☐ SEND BOOKS ON 15-DAY TRIAL BASIS & bill me (with small charge for shipping and handling).

Name _____

Address _____

City _____ State _____ Zip _____

I prefer to charge my ☐ VISA ☐ MasterCard

Card Number _____ Expiration Date _____

Signature _____
All prices in this catalog are subject to change without notice.

MAIL TO:
Prentice Hall, Book Distribution Center, Route 59 at Brook Hill Drive, West Nyack, NY 10995

Available at better bookstores or direct from Prentice Hall.
Attention Corporate Customers: For orders in excess of 20 copies, please call 201-767-2498.
For orders of fewer than 20 copies please call 201-767-5937.

For Government Orders please contact LEARNING TRENDS, 201-767-5994.

D-UNIX-BH(2)

AT&T UNIX System V/386 Library

- UNIX System V/386 Release 3.2 Utilities Release Notes **AT&T**
- UNIX System V/386 Release 3.2 Streams Primer **AT&T**
- UNIX System V/386 Release 3.2 User's Guide **AT&T**
- UNIX System V/386 Release 3.2 Programmer's Reference Manual **AT&T**
- UNIX System V/386 Release 3.2 Streams Programmer's Guide **AT&T**
- UNIX System V/386 Release 3.2 Network Programmer's Guide **AT&T**
- UNIX System V/386 Release 3.2 Programmer's Guide Vol. I **AT&T**
- UNIX System V/386 Release 3.2 Programmer's Guide Vol. II **AT&T**
- UNIX System V/386 Release 3.2 System Administrator's Guide **AT&T**
- UNIX System V/386 Release 3.2 System Administrator's Reference Manual **AT&T**

PRENTICE HALL, Englewood Cliffs, N.J. 07632

ISBN 0-13-944893-4