68NW9209H45A

SYSTEM V/88 Release 3.2 System Administrator's Guide

(Part 2)

MOTOROLA

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS MAIL PERMIT NO. 2565 PHOENIX, ARIZONA

POSTAGE WILL BE PAID BY ADDRESSEE

MOTOROLA INC.

Computer Group, Microcomputer Division, DW164
2900 South Diablo Way
Tempe, AZ 85282-9741

MOTOROL

Motorola welcomes your

Manual Title _____

Part Number _____

Your Name _____

Your Title _____

Company _____

Address _____

General Information:

Do you read this manu

☐ Install the product

☐ Reference inform

In general, how do yo

☐ Index ☐ Table o

Completeness: ☐ Ex

What topic would you

Presentation: ☐ Excellent ☐ Very Good ☐ Good ☐ Fair ☐ Poor

What features of the manual are most useful (tables, figures, appendixes, index, etc.)?

Is the information easy to understand? ☐ Yes ☐ No If you checked no, please explain:

Is the information easy to find? ☐ Yes ☐ No If you checked no, please explain:

Technical Accuracy: ☐ Excellent ☐ Very Good ☐ Good ☐ Fair ☐ Poor

If you have found technical or typographical errors, please list them here.

Page Number	Description of Error

SYSTEM V/88 Release 3.2

System Administrator's Guide

Part 2

(68NW9209H45A)

PREFACE

The *System Administrator's Guide, Part 2* provides more detailed information about the ten sets of procedures described in *Systems Administrator's Guide, Part 1*. It also contains five appendices and a glossary.

This guide assumes that you know the mechanics of using a computer terminal to enter commands, and that you have an awareness of such system fundamentals as the directory structure and the shell. We also expect that you feel comfortable using your computer; you know how to turn it on and how to use the diskette drive.

First Edition February 1990

Contents

P3 Processor Operations Procedures

P4 Disk Management Procedures

P5 File System Administration Procedures

Figures

Figures

Tables

1 System Security

Introduction

This chapter deals with maintaining the security of your computer system:

- Important security guidelines

  Guidelines for setting passwords and permissions; protecting the system from unauthorized access.

- Logins and passwords

  Aging passwords to control the amount of time passwords may be kept for logins.

  Locking logins to prevent their being used.

  Protecting administrative commands and logins with passwords.

- Set-UID and Set-GID

  Preventing unauthorized use of programs conditioned to execute via an administrative login.

Important Security Guidelines

The security of the system is ultimately the responsibility of all who have access to the system. No system is totally secure; the system is not tamperproof. Some of the items to consider are:

- Anyone with physical access to the machine, especially to a small computer, can walk off with it.
- To allow only the necessary permissions for owner, group, and others, set the access permissions to directories and files .
- All logins should have passwords. Change passwords regularly. Do not pick obvious passwords. Six to eight character nonsense strings using letters and numbers are recommended over standard names. Logins that are not needed should be either removed or blocked.
- Any system with dial-up ports is not really secure. Top secret information should not be kept on a system with dial-up ports.

- Users who make frequent use of the **su** command can compromise the security of your system by accessing files belonging to other users without the other users' knowledge. This command is also dangerous since you must know another user's login password to use it. The more people who know a given login and password, the less secure access is to the system. For this reason, a log is kept on the use of the command. Check the file **/usr/adm/sulog** to monitor use of the **su** command. The format of **/usr/adm/sulog** is described in Appendix B, *Administrative Directories and Files.*
- Login directories, **.profile** files, and files in **/bin**, **/usr/bin**, **/usr/lbin**, and **/etc** that are writable by others are security give-aways.
- Encrypt sensitive data files. The **crypt**(1) command together with the encryption capabilities of the editors (**ed** and **vi**) provide protection for sensitive information. This requires the Security Administration Utilities package (domestic customers only).
- Log off the system if you must be away from the data terminal. Do not leave a logged-in terminal unattended, especially if you are logged in as **root**.
- Be mindful of users having permission to mount file systems contained on removable media (diskettes). There is currently no mechanism in the operating system to protect against mounting file systems that contain:
  - — Set-UID or Set-GID programs.
  - — Device nodes with general write permissions.

Logins and Passwords

The discussion of logins and passwords covers:

- password aging
- sample **/etc/passwd** entries
- locking unused logins
- special administrative logins
- the **login**(1) program

Password Aging

The password aging mechanism forces users to change their password on a periodic basis. Provisions are made to prevent a user from changing a new password before a specified time interval. Password aging is selectively applied to logins by editing the **/etc/passwd** file. Realistically, password aging forces a user to adopt at least two passwords for a login. Also, refer to the **/etc/shadow** section at the end of this Chapter.

The password aging information is appended to the encrypted password field in the **/etc/passwd** file. The password aging information consists of a comma and up to four bytes (characters) in the format:

,*Mmww*

where:

, is the delimiter between the password itself and the aging information.

M is the **M**aximum duration of the password (in weeks, following the notation in Figure 1-1).

m is the **m**inimum time interval before the existing password can be changed by the user (in weeks, following the notation in Figure 1-1).

ww is the week (counted from the beginning of 1970) when the password was last changed; two characters, *ww*, are used. You do not enter this information; the system automatically adds these characters to the password aging information.

All times are specified in weeks (0 through 63) by a 64 character alphabet. Figure 1-1 shows the relationship between the numerical values and character codes. Any of the character codes may be used in the four fields of the password aging information.

Character	Number of Weeks
. (period)	0 (zero)
/ (slash)	1
0 through 9	2 through 11
A through Z	12 through 37
a through z	38 through 63

Figure 1-1. Password Aging Character Codes

Two special cases apply for the character codes:

- If *M* and *m* are equal to zero (the code, **..**), the user is forced to change the password at the next login. No further password aging is then applied to that login.
- If *m* is greater than *M* (for example, the code, **./**), only **root** is able to change the password for that login.

Sample /etc/passwd Entries

Password administration can be set up in a variety of ways to meet the needs of different organizations. Some examples are discussed in the following sections.

Change Every Two Weeks

The following shows the password aging information required to establish a new password every 2 weeks (0) and to deny changing the new password for 1 week (/).

1. Here is a typical login/password entry in the **/etc/passwd** file for the typical user **jqu**:

```
jqu:RTKESmMOE2m.E:100:1:J. Q. Username:/usr/jqu:
```

2. To cause **jqu** to change the password at least every 2 weeks, but keep it at least for 1 week, you should use the code **0/**. After you edit the **/etc/passwd** file, adding **,0/** to the password field, the entry appears as:

```
jqu:RTKESmMOE2m.E,0/:100:1:J. Q. Username:/usr/jqu:
```

   After the password entry is changed, **jqu** will have to change the password at the next login and every 2 weeks thereafter.

3. After **jqu**'s first login following the change, the system automatically adds the two-character, "last-time-changed" information to the password field:

```
jqu:RTKESmMOE2m.E,0/W9:100:1:J. Q. Username:/usr/jqu:
```

   In this example, **jqu** changed the password in week **W9**.

Change One Time Only

The following shows the password aging information required to establish for one time only, a new password when the user next logs in (using the **..** code). This procedure forces users to know some password (chosen by System Administrator) before assigning their own. It is useful for creating new user accounts with a temporary password instead of a `NULL` password. If this procedure were not performed, anyone could login while the password was `NULL`.

1. Here is the typical login/password entry for user **jqu** again:

```
jqu:RTKESmMOE2m.E:100:1:J. Q. Username:/usr/jqu:
```

2. To cause **jqu** to change the password at the next login (and to cause this only once), you should use the code **..** After you edit the **/etc/passwd** file, adding **,..** to the password field, the entry appears as:

```
jqu:RTKESmMOE2m.E,..:100:1:J. Q. Username:/usr/jqu:
```

   After the password entry is changed, **jqu** will have to change the password at the next login only.

3. After **jqu** supplies the new password, the system automatically removes the aging code (,..) from the password field:

```
jqu:EFDNLqsFUj.fs:100:1:J. Q. Username:/usr/jqu:
```

Note that the encrypted password information has changed.

Change by root Only

The following shows the password aging information required to establish a password for a login so that only the **root** login can change the password (using the **./** code).

1. Here is the typical login/password entry for user **jqu** again:

```
jqu:RTKESmMOE2m.E:100:1:J. Q. Username:/usr/jqu:
```

2. To prevent **jqu** from changing the password, you should use the code **./**. After you edit the **/etc/passwd** file, adding **,./** to the password field, the entry displays as:

```
jqu:RTKESmMOE2m.E,./:100:1:J. Q. Username:/usr/jqu:
```

Now only **root** can change the password for the **jqu** login. If **jqu** tries to change the password, a permission denied message displays.

Locking Unused Logins

If a login is not used or needed, you should do one of two things:

- remove the entry from **/etc/passwd**
- disable (lock) the login

A login is locked by editing the **/etc/passwd** file and changing the encrypted password field to contain one or more characters that are not used by the encryption process. One way to do this is to use the expression **Locked;**. In this expression, the semicolon (;) is an unused encryption character. The text, however, only serves to remind you that the login is locked. Another expression, **not valid**, could also be used to prevent the use of a login because a space is another unused encryption character. Also, refer to the **/etc/shadow** section at the end of this Chapter.

The following line entry from the **/etc/passwd** file shows the "locked" **bin** login:

```
bin:Locked;:2:2:0000-Admin(0000):/bin:
```

Special Administrative Passwords

There are two familiar ways to access the system: either via a conventional user login or the **root** login. If these were the only two ways to access the system, however, effective use of the system would have to be curtailed (because **root** would own many directories) or many users would have to know the **root** password (a bad security risk) or the system would be wide open (because **root** would own few directories). All of these conditions are undesirable.

A good mix of system use and system security is available to you through the use of special system logins and administrative commands that can be password-protected (see Procedure 1.4). The following commands, which are also logins, that can be password-protected perform functions that might be needed by many users on your computer system:

Function	Use
setup	is used to set up the computer. Once the machine has been set up, you do not want anyone doing it again without your knowledge.
sysadm	allows access to many useful administrative functions that do not require a user to log in as root.
powerdown	powers the computer down.
checkfsys	initiates a file system check on the specified file systems.
makefsys	makes a new file system on the specified media.
mountfsys	mounts the specified file system for use.
umountfsys	unmounts the specified, previously mounted file system.

NOTE

The **chk**(1), **fs**(1), **mnt**(1), and **umnt**(1) commands are also available for removable media file system operations. These commands are *not* password protected. They can be rendered useless, however, by removing entries from the **filesys** file (see **filesys**(4)).

These commands allow access to selected directories and system functions. They may be used as login names at the **login** prompt as well as commands. If you log in to the system with one of these names, the system will execute the command after login and exit to the **login** prompt once you quit or complete the function performed by the command.

Most of these system functions allow a user access to critical portions of the operating system. For this reason, it is recommended that you assign passwords to these commands. Once you assign passwords to these commands, any user attempting to log on to your system using one of these commands as a login (and any user attempting to execute one of these commands from the shell) is prompted for the password. It is recommended that the passwords to these commands/logins be known to only a few users.

Login	Use
root	This login has no restrictions on it and it overrides all other logins, protections, and permissions. It allows the user access to the entire operating system. The password for the **root** login should be very carefully protected.
sys	This login has the power of a normal user login over the files it owns, which are in **/usr/src**.
bin	This login has the power of a normal user login over the files it owns, which are in **/bin**.
adm	This login has the power of a normal user login over the object files it owns, which are in **/usr/adm**.
uucp	This login owns the object and spooled data files in **/usr/lib/uucp**.
nuucp	This login is used by remote machines to log into the system and initiate file transfers via **/usr/lib/uucp/uucico**.
rje	This login has an entry in **/etc/passwd**. There are no files currently associated with this login.
daemon	This is the login of the system daemon, which controls background processing.
lp	This login owns the object and spooled data files in **/usr/spool/lp**.

By default, the system and administrative logins above do not have passwords.

Set-UID and Set-GID

The set-user identification (set-UID) and set-group identification (set-GID) bits must be used very carefully if any security is to be maintained. These bits are set through the **chmod**(1) command and can be specified for any executable file. When any user runs an executable file that has either of these bits set, the system gives the user the permissions of the owner of the executable.

System security can be compromised if a user copies another program onto a file with **-rwsrwxrwx** permissions. For example, if the switch user (**su**) command has the write access permission allowed for others, anyone can copy the shell onto it and get a password-free version of **su**. The following paragraphs provide a few examples of command lines that can be used to identify the files with a set-UID.

For more information about the set-UID and set-GID bits, see **chmod**(1) and **chmod**(2).

Check Set-UIDs Owned by root

The following command line lists all set-UID programs owned by **root**. The results are mailed to **root**. All mounted paths are checked by this command starting at **/**. Any surprises in **root**'s mail should be investigated.

```
# find / -user root -perm -4100 -exec ls -l {} \; | mail root
you have mail
# mail
From root Mon Aug 27 07:20 EDT 1984
-r-sr-xr-x   1 root   bin     38836 Aug 10 16:16 /usr/bin/at
-r-sr-xr-x   1 root   bin     19812 Aug 10 16:16 /usr/bin/crontab
-r-sr-xr-x   1 root   bin     27748 Aug 10 16:16 /usr/bin/shl
---s--x--x   1 root   sys     46040 Aug 10 15:18 /usr/bin/ct
-r-sr-xr-x   1 root   sys     12092 Aug 11 01:29 /usr/lib/mv_dir
-r-sr-sr-x   1 root   bin     33208 Aug 10 15:55 /usr/lib/lpadmin
-r-sr-sr-x   1 root   bin     38696 Aug 10 15:55 /usr/lib/lpsched
---s--x---   1 root   rar     45376 Aug 18 15:11 /usr/rar/bin/sh
-r-sr-xr-x   1 root   sys     11416 Aug 11 01:26 /bin/mkdir
-r-sr-xr-x   1 root   sys     11804 Aug 11 01:26 /bin/rmdir
-r-sr-xr-x   1 root   bin     12524 Aug 11 01:27 /bin/df
-rwsr-xr-x   1 root   sys     21780 Aug 11 01:27 /bin/newgrp
-r-sr-sr-x   1 root   sys     23000 Aug 11 01:27 /bin/passwd
-r-sr-xr-x   1 root   sys     23824 Aug 11 01:27 /bin/su
? d
#
```

In this example, an unauthorized user (**rar**) has made a personal copy of **/bin/sh** and has made it set-UID to **root**. This means that **rar** can execute **/usr/rar/bin/sh** and become the super user.

Check Set-UIDs in the root File System

The following command line reports all files with a set-UID for the root file system. The **ncheck**(1M) command, by itself, can be used on a mounted or unmounted file system. The normal output of the **ncheck –s** command includes special files. Here, the **grep** command is used to remove device files from the output. The filtering done in this example to remove the device files is applicable only for the root file system (**/dev/dsk/c1d0s0**). The output of the modified **ncheck** is used as an argument to the **ls** command. The use of the **ls** command is possible only if the file system is mounted.

```
# ls –l 'ncheck –s /dev/dsk/c1d0s0 || cut –f2 | grep –v dev'
-r-sr-xr-x   1 root     bin        12524 Aug 11 01:27 /bin/df
-rwxr-sr-x   1 root     sys        32272 Aug 10 15:53 /bin/ipcs
-r-xr-sr-x   2 bin      mail       32852 Aug 11 01:28 /bin/mail
-r-sr-xr-x   1 root     sys        11416 Aug 11 01:26 /bin/mkdir
-rwsr-xr-x   1 root     sys        21780 Aug 11 01:27 /bin/newgrp
-r-sr-sr-x   1 root     sys        23000 Aug 11 01:27 /bin/passwd
-r-xr-sr-x   1 bin      sys        27964 Aug 11 01:28 /bin/ps
-r-xr-sr-x   2 bin      mail       32852 Aug 11 01:28 /bin/rmail
-r-sr-xr-x   1 root     sys        11804 Aug 11 01:26 /bin/rmdir
-r-sr-xr-x   1 root     sys        23824 Aug 11 01:27 /bin/su
-r-xr-sr-x   1 bin      sys        21212 Aug 10 16:08 /etc/whodo
#
```

In this example, nothing looks suspicious.

Check Set-UIDs in Other File Systems

The following command line entry shows the use of the **ncheck** command to examine the **usr** file system (**/dev/usr**) for files with a set-UID. In this example, the complete path names for the files start with **/usr**. **/usr** is not part of the **ncheck** output.

```
# ncheck -s /dev/usr/00s2 | cut -f2
/dev/00s2
/bin/at
/bin/crontab
/bin/shl
/bin/sadp
/bin/timex
/bin/cancel
/bin/disable
/bin/enable
/bin/lp
/bin/lpstat
/bin/ct
/bin/cu
/bin/uucp
/bin/uuname
/bin/uustat
/bin/uux
/lib/mv_dir
/lib/expreserve
/lib/exrecover
/lib/accept
/lib/lpadmin
/lib/lpmove
/lib/lpsched
/lib/lpshut
/lib/reject
/lib/mailx/rmmail
/lib/sa/sadc
/lib/uucp/uucico
/lib/uucp/uusched
/lib/uucp/uuxqt
/rar/bin/sh
#
```

In this example, the **/usr/rar/bin/sh** should be investigated.

The login Program

Four versions of **login** are supported on the system. They offer different variations of password use and restrictions for the superuser:

login.secure
Requires a password; the superuser can log in only on the console.

login.unsecure
Does not require a password if the password field in **/etc/passwd** or **/etc/shadow** is empty; the superuser can log in on any terminal.

login.notcons
Requires a password; the superuser can log in on any terminal.

login.nopass
Does not require a password if the password field in **/etc/passwd** or **/etc/shadow** is empty; the superuser can log in only on the console.

The System Administrator may link any one of these versions to **/bin/login**; by default, **login.secure** is linked to **login**.

Security Enhancements

Several features and enhancements have been added to Release 3.2 to enhance security. These changes include various logging features, the shadow password file, the sticky bit for directories, and **/bin/sh** enhancements. There are also several new security-related tools and procedures.

This section provides details about some of these features.

/usr/spool/cron

The **/usr/spool/cron** directory that contains directories for **at**(1), **cron**(1M), and **crontab**(1) jobs is no longer accessible to users. (The directory mode will now be set to 700.) If you want to read your own file, you must use **crontab –l**.

login(1)

In the past, 10 login attempts were permitted before a line was dropped. That limit is now five attempts. After the fifth unsuccessful login attempt, **login**(1) will sleep for 20 seconds and then drop the line.

passmgmt(1M)

The **passmgmt**(1M) command does not check the system parameter MAXUID in this release. To avoid potential conflict with UIDs reserved for RFS, do not use a number larger than 60,000 as an argument to the **–u** option.

passwd(4)

Except for **root** and **setup** entries in the **/etc/passwd** file, passwords for default entries are locked on systems that have been restored (either partially or fully).

Sticky Bit

The sticky bit now has meaning when set on a directory. Until now, removing a file (or directory) required the parent directory to be writable by the attempting process. Now the sticky bit is set on the parent directory. Therefore, if you want to remove a file or directory, you must be sure of two things: first, that the parent directory is writable, and second, that at least one of the following is true:

- the user must own the file,
- the user must own the parent directory,
- the file itself must be writable by the user, or
- the user must have superuser privileges.

The sticky bit on a directory is set by a regular user via the **chmod**(1) command or the **chmod**(2) system call.

uucp(1C)

The group ID (GID) for all **uucp** directories is now **uucp** instead of **sys**. If the GID on any other file is equal to 5 (**uucp**).

loginlog(4)

To turn on the mechanism that logs unsuccessful attempts to access the system, the administrator must create the file **/usr/adm/loginlog**. If this file exists and five consecutive unsuccessful login attempts occur, all are logged in **loginlog**, then **login** sleeps for 20 seconds before dropping the line. If a person makes fewer than five unsuccessful attempts, none of them is logged.

If **loginlog** does not exist, five failed login attempts still causes the system to sleep for 20 seconds and drop the line, but nothing is logged.

The **loginlog** file is a text file that contains one entry for each unsuccessful attempt. Entries in **/usr/adm/loginlog** have the following format:

login name:tty specification:time

The *login name* field contains the login name used in the failed login attempt. The *tty specification* field contains the terminal location of the login attempt and *time* contains the approximate time of the login attempt.

The default status is for this file not to exist and for logging to be off. To enable logging, create the log file with read and write permission for **root** only.

Step 1: Reset the default file creation privileges in a separate shell level:

```
/bin/sh
umask 066
```

Step 2: Create the loginlog file:

```
> /usr/adm/loginlog
```

Step 3: Set the group to **sys**:

```
chgrp sys /usr/adm/loginlog
```

Step 4: Change the ownership of the file to **root**:

```
chown root /usr/adm/loginlog
```

Step 5: Return from the newly created shell level:

exit

> CAUTION
>
> **This file may fill up quickly. To use this information and to prevent the file from getting too large, it is important to check and to clear the contents of the loginlog file occasionally.**

Enhanced Shell (/bin/sh)

In Release 3.2, there are two versions of the shell. Copies of both reside in the **/usr/adm** directory. One shell, **/usr/adm/sh.sec**, is installed as the default **/bin/sh** and is an enhanced version of the Bourne shell. This enhancement resets the effective user or group ID when:

- the real and effective group or user ID are not equal and,
- the effective group or user ID is less than 100 (excluding group ID 1).

The other version of the shell, **/usr/adm/sh.bck**, does not do any of the ID verification and is identical to the old **/bin/sh**.

The following procedures describe how to determine which version you are currently running, how to determine if the enhanced shell should be backed out, and how to switch from one version of the shell to the other.

Determining Which Shell You Are Running

The easiest way to determine which version of the shell you are running is to examine the output of a **what**(1) of **/bin/sh** by typing:

what /bin/sh

If you are running the enhanced shell, the output of **what** contains **/usr/adm/sh.sec.sl**. If you are running the **bck** version of the shell, the output of **what** contains **/usr/adm/sh.bck.sl**.

Criteria for Backing Out the Enhanced Shell

If an application does not work correctly with the enhanced shell, you can use the guidelines in this section to determine whether the enhanced version of the shell should be backed out.

Step 1: Make sure that all commands were entered without error.

Step 2: Determine if the application that failed

- uses a user or group ID less than 100 (see Chapter 2)
- and expects to pass effective permissions (IDs) to subprograms

If both these conditions exist, the shell is a strong candidate for replacement for the application to work as expected.

Procedure for Backing Out the Enhanced Shell

If you are running an application that fails consistently with this new release, you may want to consider reinstalling the **bck** shell version that resides in **/usr/adm** (see the previous section *Criteria for Backing Out the Enhanced Shell*). Any users currently on the system will not execute the newly installed shell until they log out and log in again. Any application that accesses the shell must be stopped and restarted, as well. We recommend that you do these procedures when no users are logged in.

Step 1: Log in as **root**.

Step 2: The **bck** shell is stored in **/usr/adm/sh.bck**. Copy it to **/bin**, the default shell directory:

```
cp /usr/adm/sh.bck /bin
```

Step 3: Change execution permissions on the new version of the shell:

```
chmod 555 /bin/sh.bck
```

Step 4: Change the ownership and group of the shell to **bin** and **root**, respectively:

```
chown bin /bin/sh.bck chgrp root /bin/sh.bck
```

NOTE

Because the shell is always running, the procedures in steps 5 through 7 must be followed *exactly* for this procedure to be effective.

Step 5: Save the current shell as **OLDsh**:

mv –f /bin/sh /bin/OLDsh

Step 6: Link the **bck** version of the shell to **/bin/sh**:

ln –f /bin/sh.bck /bin/sh

Step 7: Link the **bck** version of the shell to **/bin/rsh** (the restricted shell):

ln –f /bin/sh /bin/rsh

Step 8: Remove the copy of the **bck** version:

rm –f /bin/sh.bck

Installing the Enhanced Version of the Shell

If you are running the **bck** version of the shell, you may want to reinstall the enhanced version that resides in **/usr/adm**. Any users currently on the system will not execute the newly installed shell until they log out and log in again. Any application that accesses the shell must be stopped and restarted, as well. We recommend that you do these procedures when no users are logged in.

Step 1: Log in as **root**.

Step 2: The enhanced version of the shell is stored in **/usr/adm/sh.sec**. Copy it to **/bin**, the default shell directory:

cp /usr/adm/sh.sec /bin

Step 3: Change execution permissions on the new version of the shell:

chmod 555 /bin/sh.sec

Step 4: Change the ownership and group of the shell to **bin** and **root**, respectively:

chgrp root /bin/sh.sec chown bin /bin/sh.sec

NOTE

Because the shell is always running, the procedures in steps 5 through 7 must be followed *exactly* for this procedure to be effective.

Step 5: Save the current shell as OLDsh:

mv –f /bin/sh /bin/OLDsh

Step 6: Link the enhanced version of the shell to **/bin/sh**:

ln –f /bin/sh.sec /bin/sh

Step 7: Link the enhanced version of the shell to **/bin/rsh** (the restricted shell):

ln –f /bin/sh /bin/rsh

Step 8: Remove the copy of the enhanced version:

rm –f /bin/sh.sec

Shadow Password File

To protect encrypted user passwords, an optional security feature allows a system administrator to move all password and aging information from the publicly readable password file **/etc/passwd** to an access-restricted file called the shadow password file. The shadow password file contains one entry per login. Each entry consists of the following information:

username
the user's login name (ID).

password
a 13-character encrypted password for the user and a *lock* string to indicate that the login is not accessible (or no string to show that there is no password for the login).

lastchanged
the number of days between January 1, 1970, and the date that the password was last modified.

min
the minimum number of days required between password changes.

max
the maximum number of days the password is valid.

Installing /etc/shadow

Initial conversion of a system's single password file, **/etc/passwd**, to the new scheme using two files (**/etc/passwd** and **/etc/shadow**), is done by running the privileged command **pwconv**(1M), which creates **/etc/shadow** with information from **/etc/passwd**. The command populates **/etc/shadow** with the user's login name, password, and password aging information.

Further updates of these password files should be done by the command **passmgmt**(1M) and by the enhanced **passwd**(1) command. The **passwd** command updates the password and aging information in the appropriate password file. The **passmgmt** command is used to add or to change all other information in the password file(s). The **sysadm chgpasswd** command can also be used to update the password file(s).

The **pwconv** command may be run more than once. If the two files, **/etc/passwd** and **/etc/shadow**, should ever become inconsistent (for example, because someone manually changes one of the files) they may be made consistent by running **pwconv** again.

NOTE

If password aging information does not exist in **/etc/passwd** for a given user, none is added to **/etc/shadow** but the "last changed" information is updated. This occurs only when **pwconv** creates **/etc/shadow** or adds an entry not previously in the shadow password file.

Backing out /etc/shadow

Certain applications may not work with the new security password file changes. A possible indication of this problem is that you will be unable to log in.

If you are running such an application, you may have to "back out" the shadow password change so that this application will run. The **pwunconv**(1M) command (**/usr/bin/pwunconv**) accomplishes the reverse of the **pwconv** command. This command converts a system from the two-password file scheme back to the one-password file scheme. System administrators can run this command to solve compatibility problems caused by the introduction of the shadow password file.

To run the routine for "unconverting" your password scheme, type:

pwunconv

This will perform the following tasks:

- If a login ID has entries in both **/etc/passwd** and **/etc/shadow**, **pwunconv** will:
  - — Transfer the password portion in **/etc/shadow** to the corresponding entry in **/etc/passwd**.
  - — If the password aging information is present (*max* is greater than or equal to 0), aging information in **/etc/shadow** will be translated and transferred into the corresponding entry in **/etc/passwd**. Note that because there are different formats for storing aging information (days in **/etc/shadow** versus weeks in **/etc/passwd**), **pwunconv** will convert days into weeks.
  - — If the password aging information is not present (*max* is less than 0), but the *lastchanged* field in **/etc/shadow** contains a zero (**login** will force the password to expire), the aging field in the **/etc/passwd** entry will contain a "." which forces the user to change the password at the next login.
- If a login ID has an entry in **/etc/passwd**, but not in **/etc/shadow**, the entry will not be affected.
- If a login ID has an entry in **/etc/shadow**, but not in **/etc/passwd**, the entry will not be added to **/etc/passwd**.

When converting from days (in **/etc/shadow**) to weeks (in **/etc/passwd**), integer addition and division is used to round up the aging fields, thus reducing the impact of division truncation. For example, one day is converted to one week; eight days becomes two weeks.

Restoring the System After Forgetting the root Password

If you forget the **root** password you need to do a partial restore of the system. Because of the new shadow password file, the instructions in Procedure 1.5 of this guide are incomplete. If you forget your password after the shadow password file has been installed on your system, you need to do the following steps to restore your old system.

Partially Restore Your System

Follow all seven of the procedures in Procedure 1.5 of this guide. Then return to this chapter and continue with the following section (*Completing the Password Restoration*).

Completing the Password Restoration

After completing Procedure 1.5 of this guide, complete the following steps:

Step 1: Retrieve your shadow password (**/etc/shadow**) file:

mv /usr/old/etc/shadow /etc/shadow

Step 2: Change the **root** password again:

passwd root

2 User Services

Introduction

This chapter provides a variety of services to the users of your computer system:

- Login administration

  Assigning user and group IDs to persons authorized to be users of your system; maintaining the **/etc/passwd** and **/etc/group** files.

- The users' environments

  Setting up a master profile and helping users develop individual profiles; establishing environment variables.

- User communications services

  Establishing and maintaining services, e.g., message-of-the-day, news, mail.

- Anticipating user requests

  Developing an organized plan for responding to user problems.

Login Administration

Adding Users

Before users are permitted to log in to your system, they must be listed in the **/etc/passwd** file. The **adduser** selection from the **sysadm usermgmt**(1) menu or the login "setup" leads you through a series of prompts that create an entry in the **/etc/passwd** file (see Procedure 2.1). If you prefer, you can make the necessary changes to the **passwd** file yourself using an editor. You need to login as **root** to do this; **/etc/passwd** is generally installed as a read-only file. An entry in the **passwd** file consists of a single line with seven colon-separated fields (not recommended if you have an **/etc/shadow** file):

```
abc:gOQ3xsvO5bWuM,9/TA:103:123:Allen B. Cipher:/usr/abc:
```

The fields are:

login name
A valid name for logging onto the system. A login name is from 3 to 8 characters; the first character must be alphabetic. It is usually chosen by the user.

password
The encrypted form of the password, if any, associated with the login name in the first field. All encrypted passwords occupy 13 bytes. The actual password can be a maximum of 8 characters. At least one character must be numeric. This is to discourage users from choosing ordinary words as passwords.

When you add a user to the file you may use a default password, e.g., **passwd9**, and instruct the user to change it at the first login. Following the encrypted password, separated by a comma, there may be a field that controls password aging. See *Password Aging* in Chapter 1, *System Security.*

user id
The user-ID number (**uid**) is between 0 and 60,000. The number must not include a comma. Numbers below 100 are reserved. User-ID 0 is reserved for the super-user. The System Administration menu package does not permit you to specify a number below 100 when adding a user.

group id
The same conditions apply to the group-ID (**gid**) number as to the **uid**, except that the group-ID 1 is reserved for the "other" group.

account
The name of and optional additional information about the user. There is no required format for this field.

home directory
The home directory is where the user is placed upon logging in. The name is usually the same as the login name, preceded by a parent directory, e.g., **/usr**. The **modadduser** menu of the System Administration package allows you to specify the default parent directory. The home directory is the origination point of the user's directory tree.

program
The name of a program invoked at the time the user logs in. If the field is empty, the default program is **/bin/sh**. This field is most commonly used to invoke a special shell, such as **/bin/rsh** (restricted shell).

As noted previously, the password field may contain a subfield that controls the aging of passwords. A description of how the process works can be found in Chapter 1 and in **passwd**(4) in the *Programmer's Reference Manual*. The effect is to force users periodically to select a new password. If password aging is not implemented, a user can keep the same password indefinitely.

If you inspect the **/etc/passwd** file on your computer, you will see several commands listed among the user login names. These are commands, e.g., **sysadm makefsys**(1), that can have passwords assigned to them.

Changing or Deleting passwd Entries

As with adding users, there are **sysadm usermgmt** menu selections for changing or deleting user entries from the **/etc/passwd** file (see Procedures 2.2 and 2.3). You have the option, however, of using an editor to make the changes.

Sometimes a user forgets their password. When that happens (e.g., **abc**), you can login as **root** and enter the command:

```
# passwd abc     (The # prompt shows you are root.)
New password: passwd9 (The password entered is not echoed.)
Re-enter new password: passwd9
```

Since you did this as the superuser (**root**), you were not prompted for the old password. The command changes **abc**'s password to **passwd9**. You should make sure the user changes the password immediately. Alternately, if users are required (via **login**(1)) to have passwords, you could edit **/etc/passwd** and remove the users passwords entirely. Upon login, the user is prompted for a new password.

When you delete a login from **/etc/passwd** using the **sysadm usermgmt** menu, all the users files and directories are removed. If you remove an entry from the **passwd** file using an editor, you have only removed the entry. The users files remain.

Group IDs

Group IDs are used to establish another level of ownership and access to files and directories. Users with some community of interest can be identified as members of the same group. Any file created by a member of the group carries the group-ID as a secondary identification. By manipulating the permissions field of the file, the owner (or someone with the effective user-ID of the owner) can grant read, write, or execute privileges to other group members.

Information about groups is kept in the **/etc/group** file. The following is a sample entry from this file:

```
prog::123:jqp,abc
```

Each entry is one line; each line has the following fields:

group name
: The group name is from 3 to 8 characters; the first must be alphabetic.

password
: Do not use the password field.

group id
: The group id is a number from 0 to 60000. The number must not include a comma. Numbers below 100 are reserved.

login names
: The login names of group members are in a comma-separated list. A login name should be a member of no more than one group. There is nothing to prevent a user from having more than one login name, however, as long as each is unique within the system.

User's Environment

The key element in establishing an environment in which users can successfully communicate with the computer is the profile. There are two types of profiles:

1. The system profile.

   This is an ASCII text file, **/etc/profile**, that contains commands, shell procedures, and environment variables. Whenever a user logs in, the **login** process executes this file.

2. An individual user's profile.

   This is an executable commands file, **.profile**, that may reside in a user's home directory. The individual profile can contain additional commands and variables that further customize a user's environment. If one exists, it too is executed at login time, after the execution of **/etc/profile**.

A sample **/etc/profile** is shown in Figure 2-1.

```
#  The profile that all logins get before using their own .profile.
trap "" 2 3
export LOGNAME
#  Login and -su shells get /etc/profile services.
#  -rsh is given its environment in its .profile.
case "$0" in
-su)
        export PATH
        ;;
-sh)
        export PATH
        #  Allow the user to break the Message of the Day only.
        trap "trap '' 2" 2
        cat -s /etc/motd
        trap "" 2
        if mail -e
        then
                echo "you have mail"
        fi
        if [ ${LOGNAME} != root ]
        then
                news -n
        fi
        ;;
esac
umask 022
trap 2 3
```

Figure 2-1. A Default **/etc/profile**

The profile contains:

- Some environment variables that are exported (see *Environment Variables* later in this chapter).
- A file named **/etc/motd** is **cat**-ted (see *Message of the Day* later in this chapter).

- If the user is not **root**, the names of news items displays (**news –n**; see *news* later in this chapter).
- If the user has mail (**mail –e**), a message about it displays (see *mail* and *mailx* later in this chapter).
- By default, files created by users are not writable by group or other (see *umask* later in the chapter).

For information on the shell programming commands used in Figure 2-1, see **sh**(1) in the *User's Reference Manual.*

Environment Variables

An array of strings called the environment is made available by **exec**(2) when a process begins. Since **login** is a process, the array of environment strings is made available to it. An example of a typical array of strings is shown in Figure 2-2.

```
PS1=$
LOGNAME=abc
PWD=/usr/abc
HOME=/usr/abc
PATH=:/bin:/usr/bin:/usr/lbin
SHELL=/bin/sh
MAIL=/usr/mail/abc
TERM=5420
PAGER=pg
TZ=EST5EDT
TERMINFO=/usr/lib/terminfo
EDITOR=vi
```

Figure 2-2. Environment Array for a Typical User

The environment variables shown in Figure 2-2 give values to 12 names for user **abc**. Other programs make use of the information. For example, the user's terminal is defined as a vt100 (TERM=vt100). When the user invokes the editor **vi**(1), **vi** checks the file referenced by TERMINFO (**/usr/lib/terminfo**) where it learns the characteristics of a vt100 terminal (e.g., 23-line screen). New strings can be defined at any time. By convention, they are defined with the variable in uppercase, followed by an equal sign, followed by the value. Once defined, an environment variable can be made global for the user through the **export** statement. The individual **.profile** file can contain whatever the user wants. By default, the **sysadm adduser** command copies the file **/etc/stdprofile** to the new users **.profile**.

umask

A system default controls the permissions mode of any files or directories created by a user. The computer has default values of 666 for files and 777 for directories. That means that for files everyone automatically gets read and write permission. For directories, everyone gets read, write, and execute permission. (Execute permission on a directory means the ability to **cd** to the directory and to copy files from it.)

Users frequently set up a user mask in their **.profile** by means of the **umask**(1) command. **umask** alters the default permission levels by a specified amount. For example, the following command leaves the permission level for owner unchanged, lowers the permission level for group by 2, and reduces the permissions for others to zero:

```
umask 027
```

The system default was 666; this user mask changes it to 640, which translates into read and write permission for the owner, read permission for the group, and no permission for others.

There may be a **umask** command in **/etc/profile**. If there is, it does not change a user's ability to put one in **.profile**.

Default Shell and Restricted Shell

Generally, when a user logs in, the default program that is started is **/bin/sh**. There may be cases, however, where a user needs to be given a restricted shell.

A restricted shell is one where the user is not allowed to:

- change directories
- change the value of $PATH
- specify pathnames or command names containing a slash (/). That is, the user of a restricted shell may not access files or directories other than the present working directory or those included in $PATH
- redirect output

The restrictions are enforced after **.profile** has been executed.

The administrator can use a restricted shell strategy to limit certain users to the execution of a small number of commands or programs. By setting up a special directory for executables (**/usr/rbin**, for example), and controlling PATH so it only references that directory, the administrator can restrict the user's activity in whatever way is appropriate.

User Communications Services

Several ways of communicating with and among users are available in the operating system. Some of the most frequently used are described in this section.

Message of the Day

Items of broad interest that you want to make available to all users can be put in the **/etc/motd** file. The contents of **/etc/motd** display on the user's terminal as part of the login process. The login process executes a file called **/etc/profile**, which is an executable shell script that, among other things, commonly contains the command:

```
cat /etc/motd
```

Any text contained in **/etc/motd** displays for each user each time the user logs in. For this information to have any impact on users, use it sparingly and clean out outdated announcements. The following is a typical use for the Message of the Day facility:

```
5/30:  The system will be unavailable from 6-11pm Thursday,
5/30 - preventive maintenance.
```

Part of the preventive maintenance should be to remove the notice from **/etc/motd**.

news

Another electronic bulletin board facility is the **/usr/news** directory and the **news**(1) command. The directory is used to store announcements in text files, the names of which are usually used to provide a clue to the content of the news item. The **news** command is used to print the items on your terminal.

The **/etc/profile** file is also used to inform users about news items. A typical **/etc/profile** contains the line:

```
news –n
```

The **–n** argument causes the names of files in the **/usr/news** directory to be printed on a user's terminal as the user logs in. Item names display only for current items, e.g., items added to the **/usr/news** directory since the user last looked at the news. The idea of currency is implemented as: when you read a news item, an empty file named **.news_time** is written in your login directory. As with any other file, **.news_time** carries a time stamp indicating the date and time the file was created. When you log in, a comparison is made between the time stamp of your **.news_time** file and time stamp of items in **/usr/news**.

Unlike the Message of the Day where users have no ability to turn the message off, with **news**, users have a choice of several possible actions:

read everything
: If the user enters the command **news** with no arguments, all news items posted since the last time the user typed in the command prints on the user's terminal.

select some items
If the **news** command is entered with the names of one or more items as arguments, only those items selected are printed.

read and delete
After the **news** command is entered, the user can stop any item from printing by pressing the DELETE key. Pressing the DELETE key twice in a row stops the program.

ignore everything
If the user is too busy to read announcements at the moment, they can safely be ignored. Items remain in **/usr/news** until removed. The item names continue to display each time the user logs in.

flush all items
If the user wants to eliminate the display of item names without looking at the items, a couple of techniques will work. The following updates the time-accessed and time-modified fields of the **.news_time** control file:

```
$ touch .news_time
```

The following prints the news items on the NULL device:

```
$ news > /dev/null
```

write to All Users

The ability to write to all logged-in users, via the **wall**(1) command, is an extension of the **write**(1) command. It is fully effective only when used by the super-user. While **wall** is a useful device to get urgent information out quickly, users tend to find it annoying to have messages print out on their terminal right in the middle of whatever else is going on. The effect is not destructive, but is irritating. Many users guard against this distraction by including the following command in their **.profile**:

```
mesg n
```

This blocks other ordinary users from interjecting a message into your **stdout**. The **wall** command, when used by the super-user, overrides the **mesg n** command. It is best to reserve this for those times when you, as the System Administrator, need to ask users to get off the system. The use of the **wall** command is described in Procedure 2.5.

mail and mailx

The operating system offers two electronic mail utilities through which users can communicate among themselves. If your system is connected to others by networking facilities, **mail**(1) and **mailx**(1) can be used to communicate with persons on other systems.

mail is the basic utility for sending messages. **mailx** uses **mail** to send and receive messages, but adds to it a multitude of extras that are useful for organizing messages into storage files, adding headers, and many other functions.

When **mailx** is used, a set-up file is helpful. You can find a description of how to use a **.mailrc** set-up file in the *User's Guide* and in the **mailx**(1) pages of the *User's Reference Manual*.

Anticipating User Requests

As the system administrator for your computer you can expect users to look to you to help solve any number of problems. In addition to the system log described in Chapter 3, you will find it helpful to keep a user trouble log. The problems that users run into fall into patterns. If you keep a record of how problems were resolved, you will not have to start from scratch when a problem recurs.

3 Processor Operations

Introduction

This chapter describes the day-to-day operations of your computer system:

- General operating policy

  Guidelines for balancing the needs of system maintenance and the interests of your user community; suggestions for record keeping; lists of important administrative directories and files.

- Operating Levels

  Definition of the operating levels of the system; how they are controlled.

General Operating Policy

Many administrative tasks require the system to be shut down to a run level other than the multi-user state (see the section on *Operating Levels*). This means that conventional users cannot access the system. When the machine is taken out of the multi-user state, the users on the machine at the time are requested to log off. You should do these types of tasks when they will interfere the least with the activities of the user community.

Sometimes it is necessary to take the system down with little or no notice provided to the users. Try to provide the user community as much notice as possible about events affecting the use of the machine. When the system must be taken out of service, also tell the users when to expect the system to be available. Use the news (**/etc/news/***headline*) and the Message of the Day (**/etc/motd**) to keep users informed about changes in hardware, software, policies, and procedures.

At your discretion, the following items should be done as prerequisites for any task that requires the system to leave the multi-user state:

1. When possible, schedule service-affecting tasks to be done during periods of low system use. For scheduled actions, use the Message of the Day (**/etc/motd**) to inform users of future actions.
2. Check to see who is logged in before taking any actions that would affect a logged-in user. The **/etc/whodo** and **/bin/who** commands can be used to see who is on the system.

3. If the system is in use, provide the users advanced warning about changes in system states or pending maintenance actions. For immediate actions, use the **/etc/wall** command to send a broadcast message announcing that the system will be taken down at a given time. Give the users a reasonable amount of time to finish their activities and log off before taking the system down.

Maintaining a System Log

In a multi-user environment, it is strongly recommended that a complete set of records be maintained. A system log book can be a valuable tool when troubleshooting transient problems or when trying to establish system operating characteristics over a period of time. Some of the things that you should consider entering into the log book are:

- maintenance records (dates and actions)
- printouts of error messages and diagnostic phases
- equipment and system configuration changes (dates and actions)

The format of the system log and the types of items noted in the log should follow a logical structure. You should update it on a periodic basis. How you use your system will dictate the form and importance of maintaining a system log.

Administrative Directories and Files

This section briefly describes the directories and files that are frequently used by a System Administrator. For more detail about the purpose and contents of these directories and files, see Appendix B. For additional information on the formats of the system files, refer to Section 4 of the *Programmer's Reference Manual.*

root Directories

The directories of the **root** file system (*/*) are:

backups
Directory that contains the executable and control files for the backup/retore facility.

bin
Directory that contains public commands.

dev
Directory containing special files that define all devices on the system.

diag
Directory containing bootable diagnostics program.

etc
Directory that contains administrative programs and tables.

install
Directory used by the System Administration Menu package to mount utilities packages for installation and removal (**/install** file system).

lbin
Directory containing programs for support of **sysadmn**(1M) functions.

lib
Directory that contains public libraries.

local
Directory containing various programs used by the System Administrator.

lost+found
Directory used by **fsck**(1M) to save disconnected files.

stand
Directory that contains the copy of the operating system loaded by the disk-based boot loader.

tmp
Directory used for temporary files.

/u/service
Directory containing programs used by FSD for customer support.

usr
Directory used to mount the **/usr** file system. (See Chapter 5 for a description of this file system.)

Important System Files

The following files and directories are important in the administration of your computer:

/etc/checklist
File used to define a default list of file system devices to be checked by **/etc/fsck**.

/etc/fstab
File used to specify the file system(s) to be mounted by **/etc/mountall** and remote file systems to be mounted by **/etc/rmountall**. Also used by **mount**(1M).

/etc/gettydefs
File containing information used by **/etc/getty** to set the speed and terminal settings for a line.

/etc/group
File describing each group to the system.

/etc/init.d
Directory containing executable files used in upward and downward transitions to all system run levels. These files are linked to files beginning with **S** (start) or **K** (stop) in **/etc/rc***n***.d**, where *n* is replaced by the appropriate run level.

/etc/inittab
File containing the instructions to define the processes created or terminated by **/etc/init** for each initialization state.

/etc/motd
File containing a brief Message of the Day output by **/etc/profile**.

/etc/passwd
File identifying each user to the system.

/etc/profile
File containing the standard (default) environment for all users.

/etc/rc0
File executed by **/etc/shutdown** that executes shell scripts in the **/etc/rc0.d** directory for transitions to system run levels 0, 5, and 6.

/etc/rc0.d
Directory containing files executed by **/etc/rc0** for transitions to system run levels 0, 5, and 6. Files in this directory are linked from files in the **/etc/init.d** directory and begin with either a **K** or an **S**. **K** indicates processes that are stopped; **S** indicates processes that are started when entering run levels 0, 5, or 6.

/etc/rc2
File executed by **/etc/init** that executes shell scripts in **/etc/rc2.d** on transitions to system run level 2.

/etc/rc2.d
Directory containing files executed by **/etc/rc2** for transitions to system run levels 2 and 3. Files in this directory are linked from files in the **/etc/init.d** directory and begin with either a **K** or an **S**. **K** indicates processes that should be stopped; **S** indicates processes that should be started when entering run levels 2 or 3.

/etc/rc3
File executed by **/etc/init** that executes shell scripts in **/etc/rc3.d** on transitions to system run level 3 (Remote File Sharing (RFS) state).

/etc/rc3.d
Directory containing files executed by **/etc/rc3** for transitions to system run level 3 (RFS mode). Files in this directory are linked from the **/etc/init.d** directory and begin with either a **K** or an **S**. **K** indicates processes that should be stopped; **S** indicates processes that should be started when entering run level 3.

/etc/rstab
File used to specify the RFS resources from your machine that are automatically offered to remote machines on entering system run level 3 (RFS state).

/etc/shadow
When present, this file contains the encrypted user passwords. If not present, the encrypted passwords are kept in **/etc/passwd**.

/etc/shutdown
File containing a shell script that gracefully shuts down the system in preparation for system backup or for scheduled downtime.

/etc/TIMEZONE
File used to set the time zone shell variable TZ.

/etc/utmp
File containing the information on the current run state of the system.

/etc/wtmp
File that contains a history of system logins. This file should be checked periodically for size.

/usr/adm/sulog
File containing a history of **su** command usage. This file should be checked periodically for size.

/usr/lib/cron/log
File that contains a history of all actions taken by **/etc/cron**. This file should be checked periodically for size.

/usr/lib/help/HELPLOG
File that contains a history of all actions taken by **/usr/bin/help** (if it is enabled on the system).

/usr/lib/spell/spellhist
File that contains a history of all words that **spell** fails to match (if the Spell Utilities are installed on the system).

/usr/news
Directory containing news files. This directory should be checked periodically; discard old files.

/usr/spool/cron/crontabs
Directory that contains crontab files for the **adm, root,** and **sys** logins and ordinary users listed in **cron.allow**.

Each of these files is described in more detail in Appendix B.

Operating Levels

General

After you have set up your computer for the first time (plugged it in, hooked together all the hardware, installed the system software, and booted it as documented in Chapter P3, *Processor Operations Procedures*, and your Delta Series System Manual), you and other users can use the system. Whenever you turn it on (including the first time), the system comes up in a multi-user environment in which:

- the file systems are mounted.
- the **cron** daemon is started for scheduled tasks.
- the basic networking functions of **uucp** are available for use.
- the spooling and scheduling functions of the LP package are available for use.
- users can log in. The **getty**s are spawned on all connected terminal lines listed in **/etc/inittab** to have **getty**s respawned. (**gettys** are not on when the system is installed; you must turn them on.)

This is defined as the multi-user state. It is also referred to as **init** state 2 because all the activities of initializing the system are under the control of the **init** process. The "2" refers to entries in the special table **/etc/inittab** used by **init** to initialize the system to the multi-user state.

Not all activities, however, can be performed in the multi-user state. For example, if you were able to unmount a file system while users were accessing it, you would cause a lot of data to be lost. Hence, for unmounting and other system administration tasks, there is a need for another state, the single-user state.

The single-user state is an environment in which only the console has access to the system and the root file system alone is mounted. You are free to do tasks that affect the file systems and the system configuration because you are the only one on the system.

There are other system states (see Table 3-1). One of the more confusing things about the discussion of system states is that there are many terms used to identify the same thing: the particular operating level of the system.

Here is a list of frequently encountered synonyms:

- run state
  run level
  run mode
- init state
  system state

Likewise, each system state may be referred to in many ways, for example:

- single-user
- single-user mode
- run level 1, run level 2

Each state or run level clearly defines the operation of the computer. Each is defined in Table 3-1.

Table 3-1. System States

Run Level	Description
0	Power-down state. On machines that do not support powerdown, this mode is used to bring the system to an inactive state where it is safe to manually remove power.
1, s, or S	Single-user mode is used to install/remove software utilities, run file system backups/restores, and to check file systems. Though **s** and **1** are both used to go to single user state, **s** only kills processes spawned by **init** and does not unmount file systems. State **1** unmounts everything except root and kills all user processes, except those that relate to the console.
2	Multi-user mode is the normal operating mode for the system. The default is that the **root** (/) and user (**/usr**) file systems are mounted in this mode. When the system is powered up it is put in multi-user mode.
3	Multi-user/RFS mode is used to start RFS, connect your computer to an RFS network, mount remote resources, and offer your resources automatically.
4	User-defined run state.
5	Firmware mode is used both to access programs that reside in the computer's ROM or non-volatile RAM and to run programs in the **root** file system under the control of the ROM/NVRAM. An example of the former is running the machine's hardware diagnostics. An example of the latter is executing **/stand/unix** to reboot the system. On some machines, this mode (from the **init** point of view) puts the system in an inactive state where it is safe to manually enter the machine's firmware monitor.
6	Halt and reboot the operating system. The system comes up in multi-user mode (run level 2). On machines where auto-reboot is not possible, entering run level 6 causes the system to enter an inactive (effectively halted) state in which it is safe to manually reboot.

How init Controls the System State

SYSTEM V/88 always runs in one of the states described in Table 3-1. The actions that cause the various states to exist are under the control of the **init** process, which is the first general process created by the system at boot time. It reads the file **/etc/inittab**, which defines exactly which processes exist for which run level.

In the multi-user state (run level 2), **init** scans the file for entries that have a tag for the run level (tag is a 2) and executes everything after the last colon (:) on the line containing the tag. These tags represent the run levels in the table in Figure 3-1.

Look at **/etc/inittab** and notice that it is similar to the following listing.

NOTE

If **/etc/inittab** was removed by mistake and is missing during shutdown, **init** will enter the single-user state (**init s**). While entering single-user state, **/usr** will remain mounted and processes not spawned by **init** will continue to run. You should replace **/etc/inittab** before changing states again.

```
fs::sysinit:/etc/bcheckrc </dev/console >/dev/console 2>&1
sm::sysinit:/etc/sglusr </dev/console >/dev/console 2>&1
mt:23:bootwait:/etc/brc </dev/console >/dev/console 2>&1
is:2:initdefault:
p3:s1234:powerfail:/etc/shutdown -y -i0 -g0 >/dev/console 2>&1
s0:056:wait:/etc/rc0 >/dev/console 2>&1 </dev/console
s1:1:wait:/etc/shutdown -y -i8 -g0 >/dev/console 2>&1 </dev/console
s2:23:wait:/etc/rc2 >/dev/console 2>&1 </dev/console
s3:3:wait:/etc/rc3 >/dev/console 2>&1 </dev/console
of:0:wait:/etc/uadmin 2 0 >/dev/console 2>&1 </dev/console
fw:5:wait:/etc/uadmin 2 2 >/dev/console 2>&1 </dev/console
rb:6:wait:/etc/uadmin 2 1 >/dev/console 2>&1 </dev/console
co:234:respawn:/etc/getty console console
co:23:off:/etc/getty contty 9600 # 2nd port on processor card
00:23:off:/etc/getty tty00 9600 # 1st port on mvme335 card
01:23:off:/etc/getty tty01 9600 # 2nd port on mvme335 card
02:23:off:/etc/getty tty02 9600 # 3rd port on mvme335 card
03:23:off:/etc/getty tty03 9600 # 4th port on mvme335 card
```

The format of each line is:

id:level:action:process

where:

id
is four characters that uniquely identify an entry.

level
is zero or more numbers and letters (**0** through **6**, **s**, **a**, **b**, and **c**) that determines what *level*(s) *action* is to take place in. If *level* is **NULL**, the *action* is valid in all levels.

action
is one of the following:

sysinit
run *process* before **init** sends anything to the system console (**Console login:**).

bootwait
start *process* the first time **init** goes from single-user to multi-user state after the system is booted. (If **initdefault** is set to **2**, the process runs right after the boot.) **init** starts the process, waits for its termination and, when it dies, does not restart the process.

wait
when going to *level*, start *process* and wait until it is finished.

initdefault
when **init** starts, it will enter *level*; the *process* field for this *action* has no meaning.

once
run *process* once and do not start it again if it finishes.

powerfail
tells **init** to run *process* whenever a direct powerdown of the computer is requested (e.g., by a daemon when a switch to emergency power is detected).

respawn
if *process* does not exist, start it, wait for it to finish, and then start another.

ondemand
synonymous with **respawn**, but used only with *level* **a**, **b**, or **c**.

off
when in *level*, kill process or ignore it.

process
is any executable program, including shell procedures.

#
is used to add a comment to the end of a line. Everything after a # on a line is ignored by **init**.

When changing levels, **init** kills all processes not specified for that level. The following sections discuss the more manageable pieces of this table to get a clearer idea of how the system is controlled by **init**.

A Look at Entering the Multi-User State

Powering Up

When you power up your system, it enters multi-user state by default. (You can change the default by modifying the **initdefault** line in your **inittab** file.) In effect, going to the multi-user state follows these broad lines (see Figure 3-1):

1. You turn on the computer.
2. The operating system is loaded and the early system initializations are started by **init**.
3. The run level change is prepared by the **/etc/rc2** procedure.
4. Finally, the system is made public via the spawning of **getty**s along the terminal lines.

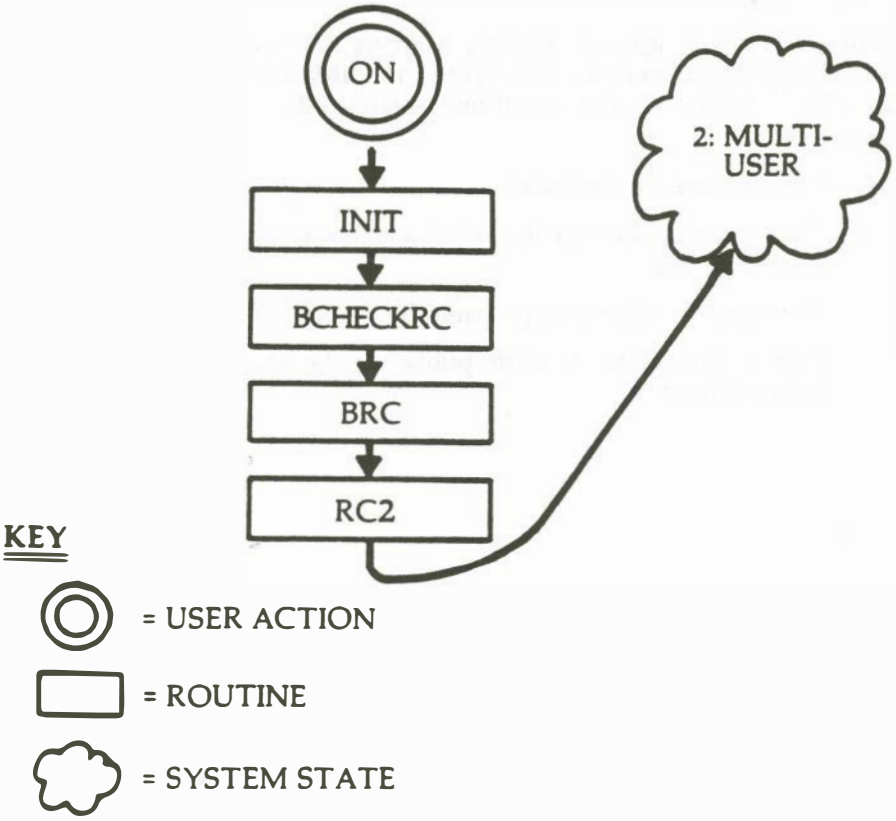


Figure 3-1. A Look at System Initialization

Early Initialization

Just after the operating system is first loaded into core via the specialized boot programs, the **init** process is created. It immediately scans **/etc/inittab** for entries of the type **sysinit**:

```
fs::sysinit:/etc/bcheckrc </dev/console >/dev/console 2>&1
```

They are executed in sequence and perform the necessary early initializations of the system. Note that each entry indicates a standard input/output relationship with **/dev/console**. This is the way communication is established with the system console before the system has been brought to the multi-user state.

Preparing the Run Level Change

Now the system must be placed in a particular run level. First, **init** scans the table to find an entry that specifies an *action* of the type *initdefault*. If it finds one, it uses the run level of that entry as the tag it will use to select the next entries to be executed. In the following **/etc/inittab** sample, the **initdefault** entry specifies run level 2 (the multi-user state) as the level to select and execute other entries:

```
is:2:initdefault:
s2:23:wait:/etc/rc2 >/dev/console 2>&1 </dev/console
co:234:respawn:/etc/getty console console
cont:23:off:/etc/getty contty 9600 # 2nd port on processor card
00:23:off:/etc/getty tty00 9600 # 1st port on mvme335 card
01:23:off:/etc/getty tty01 9600 # 2nd port on mvme335 card
02:23:off:/etc/getty tty02 9600 # 3rd port on mvme335 card
03:23:off:/etc/getty tty03 9600 # 4th port on mvme335 card
```

The other entries shown above specify the actions necessary to prepare the system to change to the multi-user run level. First, **/etc/rc2** is executed. It executes all files in **/etc/rc2.d** that begin with the letter **S**, accomplishing, among other things, the following:

- setting up and mounting the file systems
- starting the **cron** daemon
- making **uucp** available for use
- making line printer (lp) system available for use
- starting a **getty** for the Console
- starting the error logging daemon
- starting **getty** on the lines connected to the ports

At this moment, the full multi-user environment is established, and your system is available for users to log in (see Procedures 3.1 and 3.4).

A Look at the System Life Cycle

Changing Run Levels

Changing run levels follows these broad lines (see Figure 3-2):

1. The System Administrator enters a command that directs **init** to execute entries in **/etc/inittab** for a new run level.
2. Key procedures, such as **/etc/shutdown**, **/etc/rc0**, **/etc/rc2**, and **/etc/rc3**, are run to initialize the new state.
3. The new state is reached. If it is either state 1 or 5 (or possibly 6), the System Administrator can proceed.

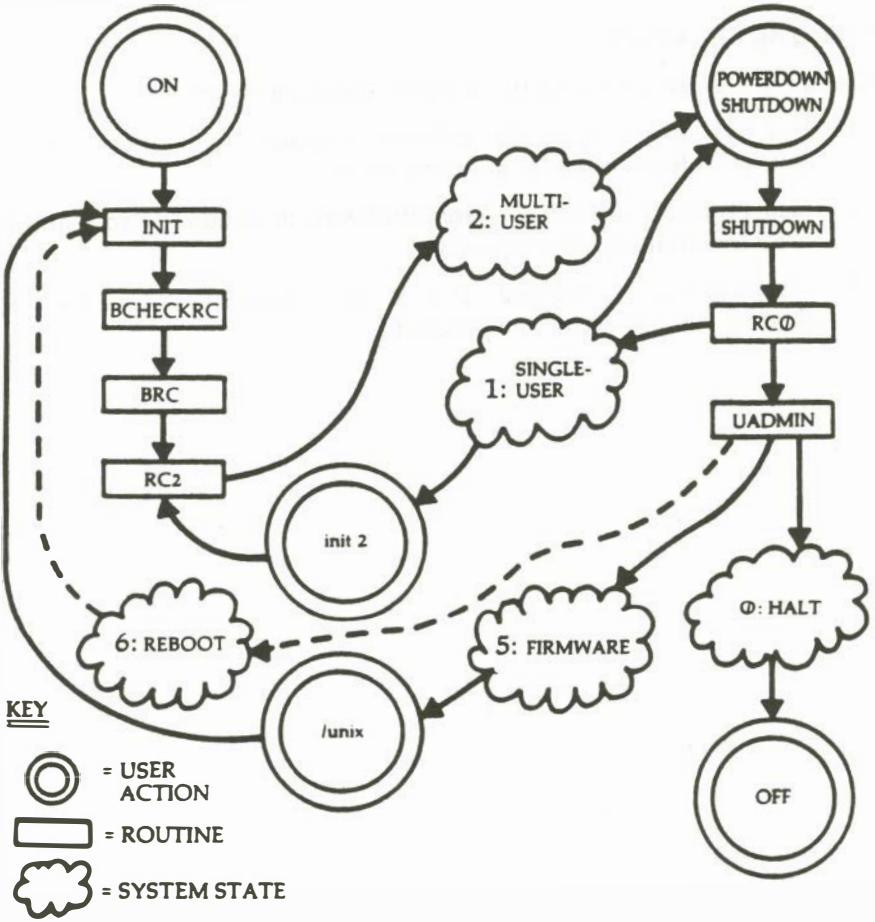


Figure 3-2. A Look at the System Life Cycle

Run Level Directories

Run levels 0, 2, and 3 each have a directory of files that are executed in transitions to and from that level. These directories are **rc0.d**, **rc2.d**, and **rc3.d**, respectively. All files in these directories are linked to files in **/etc/init.d**. The run level filenames appear as:

S*00name*

or

K*00name*

The filenames can be split into three parts:

S or **K**
The first letter defines whether the process should be started (**S**) or stopped (**K**) on entering the new run level.

00
The next two characters are a number from 00 to 99. They indicate the order in which the files will be started (S00, S01, S02, etc.) or stopped (K00, K01, K02).

name
The rest of the filename is the **/etc/init.d** filename this file is linked to.

For example, the **init.d** file **cron** is linked to the **rc2.d** file and **rc0.d** file **K70cron**. When you enter **init 2**, this file is executed with the **start** option: **sh S75cron start**. When you enter **init 0**, this file is executed with the **stop** option: **sh K70cron stop**. This particular shell script executes **/usr/bin/cron** when run with the start option and kills the **cron** process when run with the stop option.

Because these files are shell scripts, you can read them to see what they do. You can modify the files, though it is preferable to add your own since the delivered scripts may change in future releases. When creating your own scripts, follow these rules:

- Place the file in **/etc/init.d**.
- Link the file to files in appropriate run state directories using the naming convention described above.
- Have the file accept the start and/or stop options.

Going to Single-User Mode

At times, you need to go to single-user mode to perform administrative functions, e.g., backing up the hard disk to tape (see Procedure 3.3). The normal way to go to single-user mode is through the **etc/shutdown** command. This procedure executes all the files in **/etc/rc0.d** and **/etc/shutdown.d** directories by calling the **/etc/rc0** procedure; it accomplishes, among other things, the following:

- closing all open files and stopping all user processes
- stopping all daemons and services
- writing all system buffers out to the disk
- unmounting all file systems except root

The entries for single-user processing in the sample **/etc/inittab** are:

```
s1:1:wait:/etc/shutdown -y -iS -g0 >/dev/console 2>&1 </dev/console
p3:s1234:powerfail:/etc/shutdown -y -i0 -g0 >/dev/console 2>&1
```

There are two major ways to start the shutdown processing:

1. Enter the **shutdown** command (recommended).
2. Enter the **init 1** command, which forces the **init** process to scan the table. The first entry it finds is the **s1** entry; it starts the shutdown processing.

Now the system is in the single-user environment, and you can perform the appropriate administrative tasks.

Run Level 3

init 3 is used to enter the RFS state (run level 3). This procedure executes **/etc/rc2** and **/etc/rc3** to run processes in all directories associated with those two states. On top of the multi-user state (state 2) processes, the processes in **/etc/rc3.d** will:

- start RFS and connect you to the RFS network
- advertise your resources to remote computers
- mount remote resources on your computer

If you are in a RFS environment, you can change your **initdefault** entry in **/etc/inittab** from **2** to **3**, to automatically come up in RFS mode when you reboot. Details on RFS are discussed in the Chapter 10.

Run Levels 5 and 6

To go to the firmware mode or to reboot the system due to reconfiguration, use the **shutdown** command with the **–i5** or the **–i6** option. Note that the pertinent entries in **/etc/inittab** are similar or identical:

```
s0:056:wait:/etc/rc0 >/dev/console 2>&1 </dev/console
fw:5:wait:/etc/uadmin 2 2 >/dev/console 2>&1 < /dev/console
rb:6:wait:/etc/uadmin 2 1 >/dev/console 2>&1 </dev/console
```

For both system states, the **/etc/rc0** procedure is called. Then, the **/etc/uadmin** procedure with different arguments for each state is called. This procedure invokes the **uadmin**(2) system call, which directly prepares either system state.

Turning the System Off

The final step in the life cycle of the system is turning it off. The following entries apply to powering down the system:

```
p3:s1234:powerfail:/etc/shutdown -y  -i0 -g0 >/dev/console 2>&1
s0:056:wait:/etc/rc0 >/dev/console 2>&1 </dev/console
of:0:wait:/etc/uadmin 2 0 >/dev/console 2>&1 </dev/console
```

Before turning off system power, you should enter the **powerdown** command or directly invoke the **/etc/shutdown –i0** command.

In either case, the **/etc/shutdown** and **/etc/rc0** procedures are called to clean up and stop all user processes, daemons, and other services and to unmount the file systems. Finally, the **/etc/uadmin** procedure is called; it indicates when the last step (physically removing power from the system) is feasible. On some machines execution of the **/etc/uadmin** procedure can initiate the automatic removal of power under firmware control.

4 Disk Management

Introduction

This chapter discusses the disk devices on your computer:

- Disk device types and sizes
- Making devices known to the operating system
- Formatting disks
- Verifying disk usability

This chapter does not discuss file systems or the information stored on disk devices. Those subjects are covered in Chapter 5, *File System Administration.*

Device Types

The computer has two types of disk devices: sealed, integral hard disk units and drives for removable diskettes.

Possible configurations on the different models of the computer permit you to have two or more hard disk units.

All system software and user files are kept on the hard disk device(s). The cartridge tape is used primarily as a means of getting software or user files into the system where they can be used, or out of the system for storage.

Hard Disk Devices

The hard disk units come in various sizes:

- 182Mb
- 300Mb
- 390Mb
- 600Mb
- 1.2Gb

The units are sealed to protect them from dust, smoke, and other contaminants in the air. Sealing has both advantages and disadvantages. It means that the computer can operate without special dust-free climate control, but it also means that disk storage packs cannot easily be swapped in and out of use. Units that are part of a delivered computer are housed within the cabinet.

Diskette Drives

The computer normally comes with one integral diskette drive. Expansion kits that add another diskette drive may be available as an option. The drive accommodates 5¼ inch flexible diskettes (also known as floppy disks).

There are two capacities available: 640 Kb and 1.2Mb. If you have a 1.2Mb drive, you should change the names in **/dev[r]SA** from **diskette*** to **disketteHd***. This allows **sysadm format** to correctly format these drives.

Identifying Devices to the Operating System

Before a disk or tape device can be used on a computer running SYSTEM V/88, it must be made known to the system. Device names are already created for most of the equipment that might be attached to your computer.

The traditional way of handling the identification is through an entry in the **/dev** directory of the **root** file system. An entry in a directory is a file (or another directory), and conceptually a disk device is treated as if it were a file. There is a difference, however, that leads to the practice of referring to devices as "special" files. In the place where a regular file would show the character count for the file, a special file shows two decimal numbers called the major and minor numbers. Figure 4-1 shows excerpts from the output of **ls**(1) **–l** commands on a user's directory and the **/dev** directory structure.

```
                    (a regular file)
-rw-r-----   1 abc    dsg        1050 Apr 23 08:14 dm.ol
                    (hard disk device files)
brw-------   2 root   sys         2,  0 Apr 15 10:59 /dev/dsk/m323_0s0
brw-------   2 root   sys         2,  1 Apr 12 13:51 /dev/dsk/m323_0s1
crw-------   2 root   sys        21,  0 Apr 15 10:58 /dev/rdsk/m323_1s0
crw-------   2 root   sys        21,  1 Apr 12 13:51 /dev/rdsk/m323_1s1
brw-------   2 root   sys         2,  0 Apr 15 10:59 /dev/root
                    (floppy diskette device files)
brw-------   2 root   sys        87, 16 Apr 12 13:51 /dev/dsk/m327_d70s0
brw-------   7 root   sys        87, 23 Apr 22 17:56 /dev/dsk/m327_d70s0
crw-------   2 root   sys        87, 16 Apr 12 13:51 /dev/rdsk/m327_d70s0
crw-------   7 root   sys        87, 23 Apr 12 13:51 /dev/rdsk/m327_d70s0
```

Figure 4-1. Directory Listing Extracts: Regular and Device Files

The extracts from directory listings in Figure 4-1 show a regular file (indicated by the hyphen (-) in the first position of the line) with these characteristics:

- It has 1050 characters.
- The filename is **dm.ol**.
- It is owned by user **abc** who is a member of group **dsg**.
- The owner has read/write permission, group members have read permission, other users have no permissions.

There are also device files (indicated by the "b" or "c" in the first position of this line) with these characteristics:

- Major and minor numbers appear in place of the character count.

  Major is the number of the device controller or driver (an offset into a table of devices in the kernel); minor is the identifying number of the specific device.

- There can be alias names; for example, file **/dev/root** is an alias for **/dev/dsk/m323_0s0** (indicated whenever the second column's entry is 2 or greater).

- The files are owned by **root**, and no group or other user has any permission to use them. This means that only processes with the **root** ID can read from and write to these device files. (Tape devices are typically exceptions to this rule.)

Block and Character Devices

The identification as a block device or a character device has more to do with how the device is accessed than with any physical characteristics. A block device name is used when the intent is to read from or write to the device in logical, 1024-byte blocks. In the operating system, standard C language subroutines for handling file I/O work with blocks.

A character device name is used to read from or write to the device one character at a time. A character device is also referred to as a "raw" device. This is reflected in the device names or directory names shown in Figure 4-1, where the character device version of the hard disk drive is in directory **/dev/rdsk**. The character-at-a-time method is used by some file maintenance utilities.

Defining a New Special File

The need to define new special device files occurs infrequently. If you add devices, device files for them are probably already defined. When the need does occur, however, the **mknod**(1M) command is available to do it.

where:

name
specifies the *name* of the special file.

b
specifies a block device.

c
specifies a character device. The or sign (|) indicates you must specify one or the other.

major
identifies the driver type (for hardware devices, this corresponds to a controller type).

minor
identifies a specific physical device and/or type of interface.

p
specifies the special file as a first-in, first-out device. This is also known as a named pipe. (For details on pipes, see the *Programmer's Guide*.)

Formatting and Partitioning

Formatting a disk means establishing addressable areas on the medium. Partitioning means assigning file systems or other logical units to addressable areas.

Formatting Disks

Before a disk can be used for the storage of information, it must be formatted. Until the medium is formatted, its surfaces are uncharted areas treated with a substance that accepts and holds magnetic charges. Formatting imposes an addressing scheme onto these magnetic surfaces. For disks, formatting maps both sides of the disk into tracks and sectors that can be addressed by the disk controller.

A portion of a hard disk is reserved for data about the specific disk. The volume ID configuration information and bad track table reside in that area. An additional reserved area (known as the "alternate" area) is used to map portions of the disk that may not be usable. Formatting a previously used disk, in addition to redefining the tracks, erases any data that may be there.

You will have more need to format diskettes than hard disks. It is good practice to format an entire box of diskettes at the time the box is first opened. By formatting diskettes on a box basis, you avoid the problem of keeping track of which ones are formatted i.e, if the box is open, it means all the diskettes are formatted). Diskettes are formatted by the **fmt**(1) command or the **sysadm format**(1) subcommand.

Hard disks are shipped from the factory already formatted. The **dinit**(1M) command is used for formatting and for (re)mapping areas that develop defects after use.

Hard Disk Partitioning

Partitions on the hard disk devices on your computer are allocated in a standard arrangement. The arrangement varies depending on the size of the disk drives. Appendix A describes the default partitioning.

If you have a single hard disk device, it is partitioned to accommodate the root and usr file systems, swap space, and a small partition for the boot program.

If you have a multi-disk system, usr can be put on a separate disk, while root must remain on the primary disk.

The default partitions are fundamentally a compromise. After your system has been in operation for a few months, you may decide that a different assignment of file systems to partitions would better serve the needs of your users.

If you change the usr location or size (especially if you shrink it and use its previous space for another purpose), be sure to preserve this new information for future upgrades or in the event release software must be reinstalled on your disk.

Planning to Reallocate Hard Disk Partitions

The basic question in reaching a decision about repartitioning hard disk devices is whether it is better to have many smaller file systems or a few larger file systems. Other questions to consider include the following:

- What group-IDs are defined? Do we have the right number of groups and are users assigned to them appropriately?

- What kind of processing is done by the user groups? Does their work require temporary data storage? Is there a big difference between the type of processing done by different groups?
- Have we added, or are we planning to add, system software that changes our thinking about space requirements?

Helpful information about the performance of your existing file system arrangement can be obtained with the **sadp**(1M) command. The use of **sadp** is described in Chapter 6, *Performance Management*.

If you decide that repartitioning is needed, it can be done with a full system installation, moving the usr location (see the *Software Release Guide*) and/or with **dcopy**(1M). Moving the root file system is not possible.

Reallocating Partitions to Increase Swap Space

If you frequently get console messages warning of insufficient memory, it may mean that the system's current configuration of main memory and swap area is insufficient to support user demands. Before adding more main memory, an alternate solution is to allocate an additional swap area (on either single or multiple hard disk systems).

This is accomplished with the **swap**(1M) command. Reconfiguring the operating system kernel with a different default swap area and/or size is *not* recommended because future operating system updates could destroy data on your disk. Before running **swap** perform the following tasks:

1. Find out about your present partitions. (Check Appendix A for default partition sizes and use.)
2. Decide which part of which partition(s) should be used. If the disk(s) is already fully allocated, creating a new swap area means removing or reducing the size of an existing file system.

There are ways to shrink a file system:

- Take a full backup of the file system (Procedure 5.4); remake the file system with **mkfs**(1M) and Procedure 5.2; mount the new mount file system and restore the previous contents from the archive.
- Use **ipro**(1M) to compress the file system in place. Be sure to make a backup of it in the event **ipro**(1M) is interrupted in mid-compression.

If the usr file system is to be shrunk via **ipro**(1M), the system must be in single-user mode with usr unmounted.

Other Disk Operations

An additional operation you may need to perform from time to time is duplicating diskettes and verifying the usability of a diskettes.

Duplicating Diskettes

The contents of an existing diskette are copied to another diskette by using the **dd**(1)M command or the **sysadm cpdisk**(1) subcommand. If your computer has a single diskette drive, the contents of the diskette to be duplicated are first copied to a temporary file on the hard disk. When **dd** is used, either the character or the block device can be specified, but the same device (character or block) must be specified for the entire procedure. The source diskette is then replaced and the temporary file copied to the destination diskette.

While any file system can be used for the temporary file space, it is wise to use space in either **/tmp** or **/usr.tmp**. Files in those directories are automatically deleted during the transition to the multi-user mode (run level 2). No matter which directory is specified, a minimum of 1276 blocks (2560 blocks for a 1.2Mb drive) must be available (free) in that file system to use as the temporary file space. If you have more than one diskette drive, the temporary file on hard disk is not needed.

Verifying Usability

The integrity of the storage medium of a formatted diskette can be verified without changing the data on the disk by using the **dd**(1M) command. The technique is to copy the data on the disk to **/dev/null**. Either the character (**/dev/rSA/diskette1**) or the block (**/dev/SA/diskette1**) device can be specified as the source. On completion, the **dd** command reports the number of whole and partial data blocks processes (input and output). A normal diskette provides 19 + 1 blocks (32Kb blocks); a normal 1.2Mb diskette provides 39 +1 blocks (32Kb blocks). By directing to output to a null file (**/dev/null**), this procedure is independent of the amount of free disk space and requiries no file cleanup at the end of the copy.

Dynamic Bad Track Redirection

Introduction

Occasionally, new bad spots appear on a disk. This section describes a method of dynamically redirecting a track that contains a bad spot to an alternate track, without reformatting the entire disk. The affected track is copied block by block (512 bytes) to the alternate track with retries attempted in case of an error. In case of a soft error, chances of data loss are remote. In the case of an unrecoverable error, you can minimize the risk of data loss by using the utilities provided.

A menu interface, described in Procedure 4.3, has been developed for the functions described here. Before using Procedure 4.3, read through the following description of bad track handling at least once. This non-menu method can be used when the **sysadm** menus are unavailable e.g., when booted from the release media.

The sequence of steps is:

1. Preparing the input data
2. Generating Input File(s) for **fsck**
3. Assessing the file damage and saving readable files
4. Redirecting the bad track(s)
5. Repairing the file system(s)
6. Restoring lost files

1. Preparing the Input Data

You must have the bad track numbers in hand, e.g., from an error report, a bad read or write message. This data will be in one of three formats: 1) *track* numbers, 2) *head,cylinder* pairs, or 3) *logical device,block number* pairs (produced by **errpt**(1M)). Here are some examples of bad tracks in the different formats.

tracks:	*head,cyl* pair:	*logical device,block no.* pair:
6961	1 870	6 20
6998	6 874	6 620

The formatting program (**dinit**(1M)) accepts data in one of three formats: 1) *track*, 2) *head,cylinder*, or 3) *head,cylinder,BFI*. If the numbers are in the logical device, block number format, you can quickly convert them to a valid format with the function **xformtrk**(1M). This is a general conversion program that converts any of the above forms to either *track* or *head,cylinder* pairs. In the **errpt** output, the logical device is listed in decimal, then octal (in parentheses). Use the decimal value as input to **xformtrk**.

NOTE

If you are using a device that has HD, CYL, BFI, skip to Step 2.

Whether entering the numbers interactively within the **xformtrk** utility or using file input, all entries must be in the same format and for any type, the entries must be one number or pair per line. If you put the numbers in a file, there should be no extra text, just the data.

The output from **xformtrk** goes into the file **/tmp/xf**_*device*. If a file by this name exists, it will be overwritten. It is in a format suitable as input to **fblkgen**(1M), the next step in this procedure. The file will be removed at the next system reboot, so you must move it to a more permanent home if you want to keep it.

xformtrk discards input values less than zero or that have corresponding track numbers larger than the maximum *head,cylinder* or *track* value, as computed from entries in **/etc/dskdefs**.

You must know the device name and type for this and the subsequent utilities. These definitions are summarized here but you will find a more detailed definition of the device name in Appendix A.

The device type is a name (found in **/etc/dskdefs**) that uniquely identifies the device. For example, **m323182** is the device type for the CDC 182 megabyte ESDI drive. Devices are listed on the **dinit**(1M) manual page.

The disk device name passed to **xformtrk** or **fblkgen** must be of the form *x_y*, where *x* is the controller mnemonic and *y* is the drive logical unit number on that controller, e.g., **m323_0**.

The command format is:

xformtrk -I *opt* **-O** *opt* [**-t** *file*] *devtype device*

where:

-I *opt*
input format: **t** for track, **c** for *head,cylinder* pairs or **d** for *logical device,block number* pairs.

-O *opt*
output format: **t** for *track* or **c** for *head,cylinder* pairs. There is no *device* output format.

-t *file*
input file name. If you leave this off, you will be prompted to enter the numbers interactively.

For example:

/etc/xformtrk -I d -O c -t badtracks m323182 m323_0

Convert input for file **badtracks** that is in *logical device,block number* format, to *cylinder,head* format; leave in **/tmp/xf.m323_0.**

/etc/xformtrk -I d -O t m323182 m323_0

Prompt for input in *logical device,block number* format; convert to track numbers.

2. Generating Input File(s) for fsck

Not every track is within a file system, but if it is, you must be concerned about file system damage. The utility **/etc/fblkgen** will generate files containing the affected file system logical block numbers for each file system that contains any of the given tracks. These files are placed in **/etc/badtracks**. The names are of the form **F.***x*_*y***s***p*, where *x* is the controller, *y* is the drive logical unit number on that controller, and *p* is the partition in which this file system begins. No files are created for partitions that do not have bad blocks or contain file systems. These files are of the correct format to be used as input for **fsck** (**-r** option).

Examples:

/etc/badtracks/F.m323_0s5
/etc/badtracks/F.m323_1s2

The following files contain file system logical block numbers only, one per line; for example:

```
1501
2839
```

You must remove any existing bad block files for the device before you run **fblkgen** or you will have the new numbers appended to the old list. To see if any files exist, type the following command (the names, if any, will be printed):

```
ls /etc/badtracks/F.*
```

To remove the files for a particular device, type the following command:

```
rm /etc/badtracks/F.device*
```

You are now ready to start the **fblkgen** utility. The device type and device name are as described above for the **xformtrk** utility. The command format is:

fblkgen [**-T**] [**-t** *file*] *devtype device*

where:

-T
is data and is in single track number format.

-t *file*
is the input file name. If you leave this off, you are prompted to enter the numbers interactively.

NOTE

fblkgen accepts input in track or head, cylinder format only. Do not enter BFI.

For example:

```
/etc/fblkgen -t badtracks m323182 m323_0
/etc/fblkgen -T m323182 m323_0
```

If file systems appear to overlap (due to an obsolete file system that has not had its superblock overwritten), you will be prompted to indicate which is the current file system.

fblkgen generates multiple block numbers for each track affected. This is done because it does not know the extent of the damage, and because redirection can only be done on a track basis. On a 182 megabyte CDC, for example, there are 36 512-byte blocks.

3. Assessing the File Damage and Saving Readable Files

Before you do the track redirection, it is strongly recommended that you pass through this procedure once in a no-write mode to see what is going to happen and minimize your losses by backing up any files you can.

After the track-to-block conversion of the previous step, for any file system that will be affected, the file names will appear in **/etc/badtracks**. You can identify the device and the partition in which the file system begins by the file name (see *Generating Input Files for fsck). For example, you entered two track numbers and the files* **F.m323_0s5** and **F.m323_0s6** are created. The device is **m323_0** and the file systems affected are in partitions 5 and 6.

To see if any files exist indicating affected file systems, type in the following command and the names, if any exist, will be printed as shown in the following sections:

ls /etc/badtracks/F.*

When using the -r option of **fsck**, you may check only one file system at a time since only one instance of **-r** is permitted. The file system must be unmounted for the check. For details, see **mount(1M)** in the *System Administrator's Reference Manual*.

To begin the file system check, enter the command:

fsck -n -r <*file*> *partition*

where:

-n
do not write on this pass.

-r *file*
file contains the file system logical block numbers that may be corrupted (zeroed) by the redirection. List the files/directories whose data blocks match an entry in *file*. Also list i-node blocks that will be corrupted, and the i-nodes affected.

partition
the full device and partition number to be checked. Examples: **/dev/dsk/m323_0s5** and **/dev/dsk/m323_0s6**.

For example:

/etc/fsck -n -r /etc/badtracks/F.m323_0s5 /dev/dsk/m323_0s5

If a file is found that may be corrupted due to lost data from the track redirection or that may already be corrupt due to the bad track itself, a message like the following appears:

```
FILE/DIR  CORRUPT   I=47  OWNER=bin MODE=100755
  SIZE=35497 MTIME=Jun 24 10:07 1986
  FILE/DIR = [/dev/rdsk/m323_0s2]/bin/create
```

Note that the file name is relative to the file system described by the device. In the above example, **m323_0s2** is the usr file system so the corrupt file is **/usr/bin/create**.

If any of the file system blocks in **/etc/badtracks/F.m323.0s5** are in the I-list, a message like the following appears:

```
I-BLOCK CORRUPT - BLOCK=100
  INODES 1569-1584 UNRECOVERABLE
```

This is followed in Phase 2 with an `UNALLOCATED` message for each of the files affected (16 files for a 1K-block file system).

You should write down the names of the files affected (denoted in the **fsck** output as `FILE/DIR CORRUPT` or `UNALLOCATED`) because once the redirection is done, some of that information may be lost.

It may now be possible to copy most of the listed files elsewhere before the redirection (since only a few of the blocks listed by **fblkgen** may actually be unreadable). After the redirection, they can be restored to their original location. The files directly affected by the bad spot on the track may be unreadable; these must be restored from a previous backup.

4. Redirecting the Bad Track(s)

If everything has gone as expected until this point, you are ready to begin the redirection. If not, no changes have been made and you can quit at this time.

The **dinit** command used is:

dinit [-T] -n *devtype rawdev*

Example:

dinit -n m323182 /dev/rdsk/m323_0s7

dinit prompts the user for bad spot information. The input is taken from **stdin**. Only one bad spot is accepted per invocation of **dinit** when the -n option is used.

5. Repairing the File System(s)

Before you begin, make sure that **/lost+found** directories exist for each file system you are going to check. Recall that you can identify the file systems by the names beginning with **F.** in **/etc/badtracks**. Depending on the kind of damage, some files can be reconnected. (The files you generated for **fsck** should still be in **/etc/badtracks**. If not, repeat step 2.)

Repeat step 3, leaving off the **-n** (no-write) and **-r** options. You may select either automatic repair (**-y**) or interactive repair (no option). The command is:

fsck *partition*

Repeat steps 4 and 5 as required.

See **fsck**(1M) in the *System Administrator's Reference Manual.*

6. Restoring Lost Files

Follow the procedures for Backup and Restore to restore files and directories that have been corrupted or lost due to the track redirection. The readable files saved in the previous step can be restored. Any that could not be accessed because of the damage to the track must be recovered from a previous backup.

In certain cases, files may have been reconnected to **/lost+found** of the file system. Check there for any files that a user has changed since the last backup but were lost in the redirection.

5 File System Administration

Introduction

How the File System Is Organized

A primary function of the operating system is to support file systems. In the operating system, a file is a one-dimensional array of bytes with no other structure implied. Files are attached to a hierarchy of directories. A directory is another type of file that the user is permitted to use, but not to write; the operating system itself retains the responsibility for writing directories. The combination of directories and files make up a file system. Figure 5-1 shows the relationship between directories and files in a SYSTEM V/88 file system. The circles represent directories.

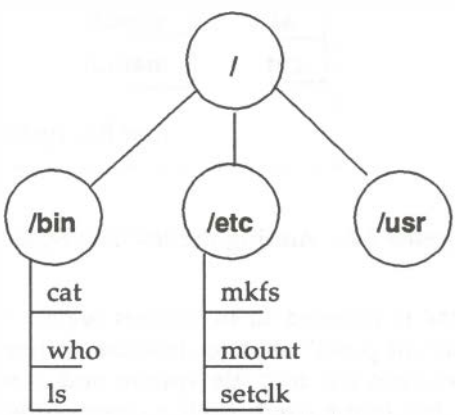


Figure 5-1. A SYSTEM V/88 File System

The starting point of any SYSTEM V/88 file system is a directory that serves as the root. In the operating system, there is always one file system that is itself referred to by that name, **root**. Traditionally, the root directory of the root file system is represented by a single slash (*/*). The file system diagrammed in Figure 5-1, then, is a root file system. If we graft another file system onto root at a directory called, for example, usr, the result can be illustrated by the diagram in Figure 5-2.

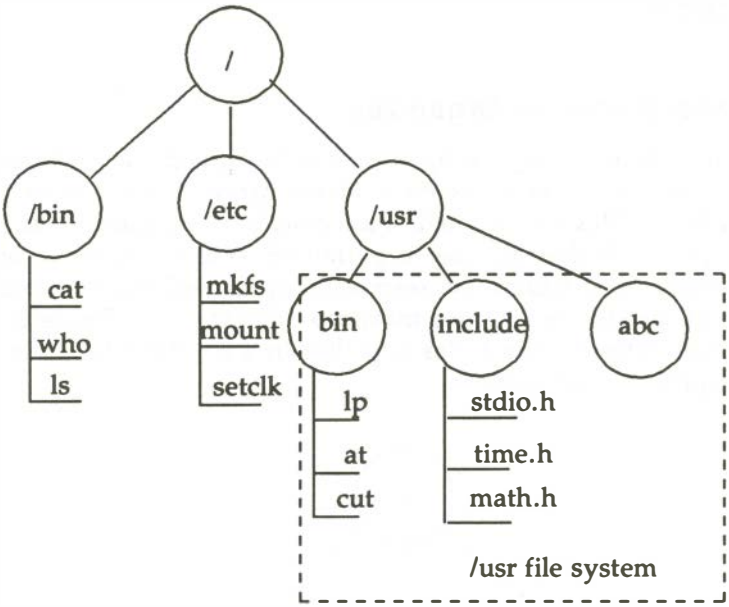


Figure 5-2. Adding the **usr** File System

A directory such as **/usr** is referred to in various ways. You sometimes see the terms "leaf" and "mount point" used to describe a directory that is used to form the connection between the root file system and another mountable file system. Regardless of the terms used, such a directory is the root of the file system that descends from it. The name of that file system is, coincidentally, the name of the directory. In our example, the file system is usr.

The diagrams in Figures 5-1 and 5-2 may be a convenient representation of the file and directory structure of file systems, but they are not particularly accurate, or helpful, ways of illustrating how a file system is known to the operating system. The operating system views a file system as an arrangement of addressable blocks of disk space that can be classified in four categories:

- block 0
- block 1: the super-block
- a variable number of blocks comprising the i-list
- a variable number of storage blocks: most contain data, some contain the freelist and indirect addresses

This scheme is illustrated in Figure 5-3.

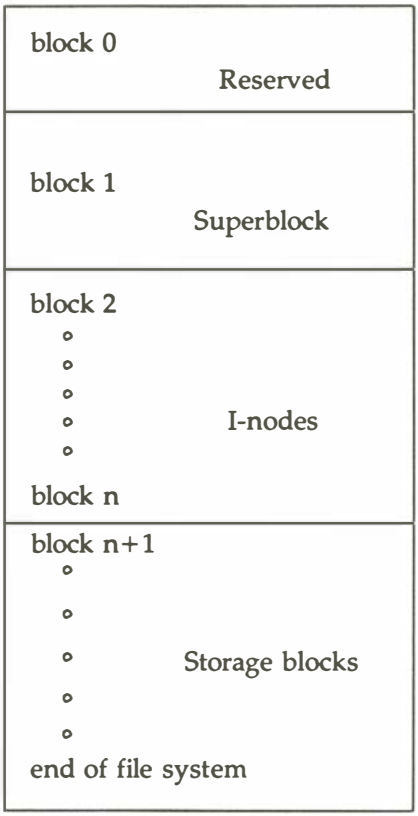


Figure 5-3. The SYSYEM V/88 View of a File System

Block 0

Block 0, although considered to be part of the file system, is not used by it. is reserved for storing booting procedures. Not all file systems are involv in booting. For a file system that is not, block 0 is left unused.

Block 1: Super-Block

Much of the information about the file system is stored in the super-block, including the following:

- file system size and status
  - label, file system name
  - size in physical and logical blocks
  - read-only flag
  - super-block modified flag
  - date and time of last update
- i-nodes
  - total number of i-nodes allocated
  - number of i-nodes free
  - array containing 100 free i-node numbers
  - an index into the free i-node number array
- storage blocks
  - total number of free blocks
  - array containing 50 free-block numbers
  - an index into the free-block number array

A diagram of the fields in the super-block is shown in Table 5-1. Note that the super-block does not maintain complete lists of free i-nodes and free blocks, but only enough to meet current demands as the file system is used. At almost any time, unless the file system is close to running out of i-nodes and storage blocks, there are sure to be more free i-nodes and blocks than are listed in the super-block. The information about them is kept in one of the storage blocks.

Table 5-1. The Super-Block

Miscellaneous Information - logical/physical disk sizes - superblock modified flag - read-only system flag - current date of last update - label or name
I-node Information - total number of i-nodes - total free i-nodes - array of free i-node numbers - index into i-number array
Block Information - total number of free blocks - array of free block numbers - index into array of free block numbers

i-nodes

The term "i-node" stands for information node. (You will often see it spelled with no hyphen: inode.) The same formulation is used in other references to things associated with i-nodes. For example, the list of i-nodes is referred to as the i-list (or ilist); an i-number is the position of an i-node in the i-list.

The i-node contains all the information about a file except for its name, which is kept in a directory. An i-node is 64 bytes long, so there are 8 i-nodes to a physical block. There is no set number of blocks occupied by the i-node list; it depends on how many i-nodes are specified at the time the file system is created. An i-node contains:

- type and mode of file: type is regular (–), directory (d), block (b), character (c), or FIFO, also known as named pipe, (p); mode is the set of read-write-execute permissions
- number of links to the file
- owner's user-id number
- group-id number to which the file belongs
- number of bytes in the file
- an array of 13 disk block addresses
- date and time last accessed
- date and time last modified
- date and time created

The array of 13 disk block addresses is the heart of the i-node. The first 10 are direct addresses; i.e., they point directly to the first 10 logical storage blocks of the contents of the file. If the file is larger than 10240 characters, the 11th address points to an indirect block that contains more block addresses; the 12th address points to a double indirect block that contains the addresses of 256 storage blocks. Finally, for files larger than 67,381,248 bytes, the 13th address in the array is the address of a triple indirect block that contains the addresses of 256 double indirect blocks. The theoretical maximum size of an operating system is beyond the limits of the amount of disk storage on your computer. Figure 5-4 illustrates this chaining of address blocks stemming from the i-node.

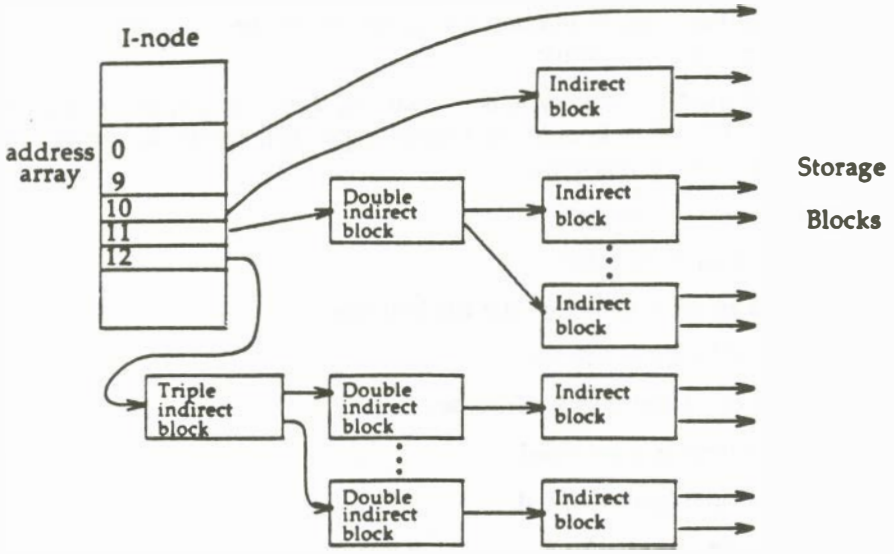


Figure 5-4. The File System Address Chain

The following table shows the number of bytes addressable by the different levels of indirection in the inode address array. These numbers are calculated using the logical block size of the file system and the number of bytes used to hold an address (4).

Logical Block Size	Maximum Number of Bytes Addressable by Direct Blocks	Single Indirect Blocks	Double Indirect Blocks	Triple Indirect Blocks
512 bytes	5120	64K	8M	1G
1024 bytes	10240	256K	64M	16G

The table shows the number of bytes addressable using the level of indirection in the column header, plus all lower levels of addressing. For example, the table values for single indirect blocks also include bytes addressable by direct blocks; the table values for triple indirect blocks include bytes addressable by direct blocks and single and double indirect blocks.

The theoretical maximum size of a SYSTEM V/88 system file is the same as the size of a file addressable with triple indirection (shown in the last column of the table). In practice, however, file size is limited by the size field in the inode. This is a 32-bit field, so file sizes are limited to 4 gigabytes.

Storage Blocks

The remainder of the space allocated to the file system is taken up by storage blocks, also called data blocks. For a regular file, the storage blocks contain the contents of the file. The contents are undefined. For a directory, the storage blocks contain 16-byte entries (64 to a physical block). Each entry represents a file or subdirectory that is a member of the directory. An entry consists of 2 bytes for the i-number and 14 bytes for the filename of the member file or subdirectory.

Free Blocks

Blocks not currently being used as i-nodes, as indirect address blocks, or as a storage blocks are chained together in a linked list. Each block in the list carries the address of the next block in the chain.

Summary

What has been described thus far is an abstract view of a SYSTEM V/88 file system, the components of a file system, and something of the way they relate to each other. Later sections of this chapter describe how file systems are stored on disks, and what happens to a file system when it is in use.

The Relationship Between the File System and the Storage Device

In the operating system, file systems reside on random-access disk devices. (You can put a file system on tape, but that is almost always for backup security instead of for live access.) Before you can install a file system on a disk there are some preliminaries that must be done. The material in this part of the chapter is a summary of material described in more detail in Chapter 4, *Disk Management*.

Disk Format

Before a disk can be used by the operating system, it must be formatted into addressable sectors. A disk sector is a 256- or 512-byte portion of the storage medium that can be addressed by the disk controller. The number of sectors is a function of the size and number of surfaces of the disk device.

NOTE

Hard disk units on the computer are formatted when installed. The only time they might need to be reformatted is after a catastrophic hardware failure. Contact your Field Service Representative if such an event occurs.

The command **dinit**(1M) is used to define a volume table of contents (VTOC) for a formatted hard disk (see the *Partitions* sections).

Diskettes are made to be usable in more than one machine. Manufacturers produce diskettes diskettes unformatted, leaving it to customers to format them for the particular machine on which they are to be used. The commands to use are:

fmt(1M)
to format a diskette

sysadm format(1)
to format a diskette using the System Administration Disk Management menu

Partitions

The next level in disk formatting are partitions or slices. Up to 8 partitions can be defined on a hard disk device and up to 2 on a floppy diskette. The MVME842 ESDI CDC devices contain 8 partitions, numbered **0** through **7** (partition **7** is defined to be the whole disk). The **sledit**(1M) command can be used to view and edit the slice table on your hard disk.

An example slice table is shown in Table 5-2:

Table 5-2. Slice Table for MVME842 ESDI CDC 182Mb Drive

/dev/rdsk/m323_0s0 (8 slices) (512-byte blocks)					
Slice Table			**File System Table**		
Slice	**Offset**	**Size**	**Size**	**FSname**	**Vol-ID**
0	648	55080	55080	root	DO
1	55728	45036	45036		
2	100764	207684	207684	usr	DO
3	0	0			
4	0	0			
5	0	0			
6	0	0			
7	0	313632			

The illustration shows two file systems defined on this drive: the root file system and one called usr. Space has also been set aside for the boot file and for swap space. Tables showing the default partitioning for all supported devices are in Appendix A.

Size Limitations

The maximum number of blocks that can be allocated to a file system is close to the total number of sectors on the disk device. This maximum may be reduced by space set aside for swapping or paging.

The maximum number of i-nodes that can be specified is 65,534.

The size of a block on a disk is 512-bytes, the same as a disk sector. The size of a logical block is set by the administrator with the **mkfs**(1M) command. As shipped, the size of a logical, or file system, block is 1024 bytes. It is possible, however, to create file systems with 512- or 8192-byte blocks (**omkfs, mkfs8k**). Subroutines that handle file I/O work with logical blocks instead of with the 512-byte physical blocks.

How the File System Works

So far, the organization of a SYSTEM V/88 file system on paper and on the physical storage disk has been discussed. This section describes what the operating system does with a file system when it is being used.

Tables in Memory

When a file system is identified to the operating system through a **mount**(1M) command, an entry is made in the mount table, and the super-block is read into an internal buffer maintained by the kernel. Parts of the super-block that are most needed in memory are the lists of free i-nodes and free storage blocks, and the flags and time fields that are constantly being modified.

The System i-node Table

The operating system maintains a structure known as the system i-node table. Whenever a file is opened, its i-node is copied from the secondary storage disk into the system i-node table. If two or more processes have the same file open, they share the same i-node table entry. The entry includes, among other things:

- name of the device from which the i-node was copied
- i-node number or i-number
- a reference count of the number of pointers to this entry (A file can be open for more than one process.)

A diagram of an entry in the system i-node table is shown in Figure 5-5.

i-node chain pointers
Free-list chain
Flag
Waiting count for i-node
Reference count
Device where i-node resides
i-number
Mode
Number of links
User-ID of owner
Group-ID of owner
Size of file

Figure 5-5. An Entry in the System i-node Table

The System File Table

The system maintains another table called the system file table. Because files may be shared among related processes a table is needed to keep track of which files are accessible by which process. For each file descriptor, an entry in the system file table contains:

- flag to tell how the file was opened (read/write)
- count of the processes pointing to this entry
  (When the count drops to zero, the system drops the entry.)
- pointer to the system i-node table
- pointer that tells where in the file the next I/O operation will take place

The Open File Table

The last table that is used to provide access to files is the open file table. It is located in the user area portion of memory. There is a user area for each process and, consequently, an open file table for each process. An entry in the open file table contains a pointer to the appropriate system file table entry. Figure 5-6 shows how these tables point to each other.

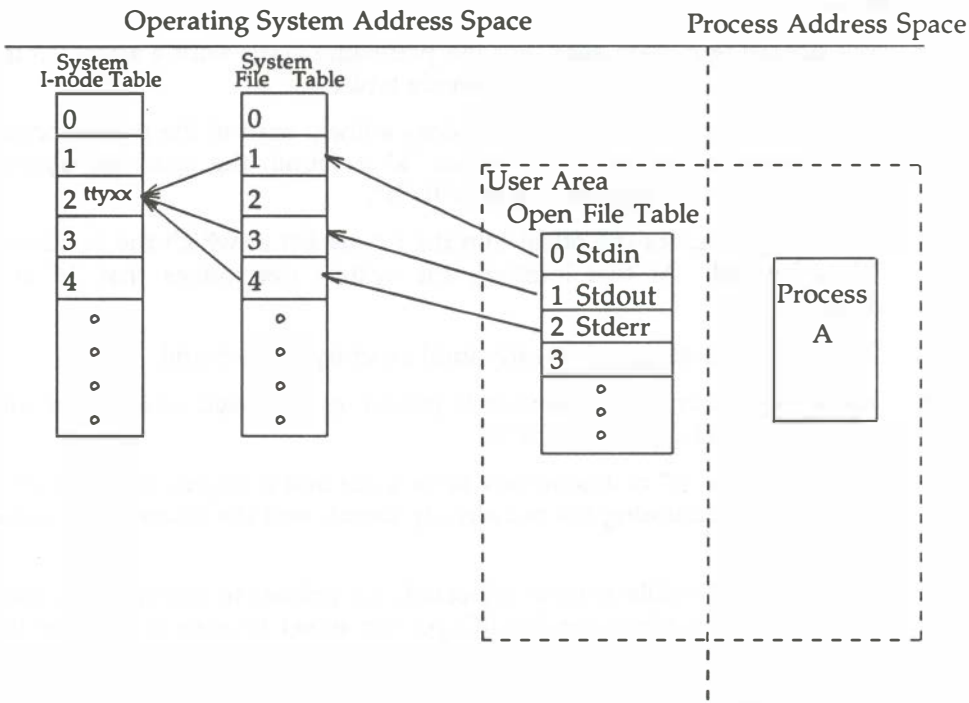


Figure 5-6. File System Tables and Their Pointers

System Steps in Accessing a File

In the next few paragraphs the steps followed by the operating system in opening, creating, reading, or writing a file are described.

Open

Suppose the pathname **/a/b** is given to the **open**(2) system call. (Our program probably uses the **fopen**(3) subroutine from the standard I/O library, but that in turn invokes the system call.)

1. The operating system sees that the pathname starts with a slash, so the root i-node is obtained from the i-node table.
2. Using the root i-node, the system does a linear scan of the root directory file looking for an entry "a". When "a" is found, the operating system picks up the i-number associated with "a".
3. The i-number gives the offset into the i-node list at which the i-node for "a" is located. At that location, the system determines that "a" is a directory.
4. Directory "a" is searched linearly until an entry "b" is found.
5. When "b" is found, its i-number is picked up and used as an index into the i-list to find the i-node for "b".
6. The i-node for "b" is determined to be a file and is copied to the system i-node table (assuming it's not already there), and the reference count is initialized.
7. The system file table entry is allocated, the pointer to the system i-node table is set, the offset for the I/O pointer is set to zero to indicate the beginning of the file, and the reference count is initialized.
8. The user area file descriptor table entry is allocated with a pointer set to the entry in the system file table.
9. The number of the file descriptor slot is returned to the program.

The linear scan algorithm for locating the i-node of a file illustrates why it is advisable to keep directories small. Search time is also speeded up by keeping subdirectory names near the beginning of a directory file (use **dcopy**(1M) to do this).

Create

Creating a file (the **creat**(2) system call) has these additional steps at the beginning:

1. The super-block is referenced for a free i-node number.
2. The mode of the file is established (possibly **and**-ed with the complement of a **umask** entry; see **umask**(2)) and entered in the i-node.
3. Using the i-number, the system goes through a directory search similar to that used in the **open** system call. The difference is that in the case of **creat**, the system writes the last portion of the pathname into the directory that is the next to last portion of the pathname. The i-number is stored with it.

Reading and Writing

Both the **read**(2) and **write**(2) system call follow these steps:

1. Using the file descriptor supplied with the call as an index, the user's open file table is read and the pointer to the system file table obtained.
2. The user buffer address and number of bytes to read(write) are supplied as arguments to the call. The correct offset into the file is read from the system file table entry.
3. **Reading**

   The i-node is found by following the pointer from the system file table entry to the system i-node table. The operating system copies the data from storage to the user's buffer.

   Writing

   The same pointer chain is followed, but the system writes into the data blocks. If new direct or indirect blocks are needed, they are allocated from the file system's list of free blocks.
4. Before the system call returns to the user, the number of bytes read(written) is added to the offset in the system file table.
5. The number of bytes read or written is returned to the user.

Files Used by More Than One Process

If related processes are sharing a file descriptor (e.g., after a **fork**(2)) they also share the same entry in the system file table. Unrelated processes that access the same file have separate entries in the system file table, because they may be reading from or writing to different places in the file. In both cases, the entry in the i-node table is shared; the correct offset at which the read or write should take place is tracked by the offset entry in the system file table.

Pathname Conversion

The directory search and pathname conversion takes place only once as long as the file remains open. For subsequent access of the file the system, supplies a file descriptor that is an index into the open file table in your user process area. The open file table points to the system file table entry where the pointer to the system i-node table is picked up. Given the i-node, the system can find the data blocks that make up the file.

Synchronization

The above description, while complex, may seem neat and orderly. The situation is complicated, however, by the fact that SYSTEM V/88 is a multi-tasking system. To give some tasks prompt attention, the system may make the decision that other tasks are less urgent. In addition, the system keeps a buffer cache and a cache of free blocks and i-nodes in memory together with the super-block to provide more responsive service to users. The stability that comes from having every byte of data in a file immediately written to the storage disk is traded for the gain of being able to provide more service to more users.

In normal processing, disk buffers are flushed periodically to the disk devices. This is a system process that is not related directly to any reads or writes of user processes. The process is called "synchronization." It includes writing out the super-blocks in addition to the disk buffers. The **sync** command can be used to cause the writing of super-blocks and updated i-nodes and the flushing of buffers. It is worth noting, however, that the return from the command simply means that the writing was scheduled, not necessarily completed. For this reason, many people enter the command twice in a row to introduce delay.

Search Time

There are two things that have a bearing on the amount of time the system needs to spend in looking for and reading in a file:

- the size of the directories being searched
- the size of the file itself

As described above, when the operating system is locating a file to be opened, it searches linearly through all the directories in the pathname. Search time can be reduced in two ways:

1. Keep the number of entries in a directory low. Unless compressed with **dcopy**(1M), a directory retains it largest size. If you have a directory that has had more than 640 entries, you have reached the point of indirect addresses.
2. Move subdirectory names to the start of the directory. The **dcopy**(1M) utility does this by default. When subdirectory names are grouped at the start of a directory, the system finds a match with less searching.

Large files are slower to read because of the need to chain through indirect or double indirect addresses.

Holes in Files

When a file is created by a program, instead of by a person using an editor, the program may seek to locations in the file that are blocks away from where previous data in the file were stored. For example, a program has written to blocks 1 and 2 of a file. Those storage blocks are pointed to from the i-node. If the program seeks to block 5 and writes data there, the fifth address pointer in the i-node will hold a live address, but the third and fourth will still be zeros pointing to nothing. The file is now said to have a hole in it. There is nothing serious about this; the blocks used by the file are the ones that have been written to. If the file system is reorganized (e.g., by **dcopy**(1M)), storage blocks will be assigned to blocks 3 and 4; addresses will be added to the i-node; the storage blocks will be taken off the free list and initialized to zeros.

Summary

This section has discussed how the operating system controls file systems. Seeing how things are supposed to work can give us an appreciation of the steps that must be taken to keep a file system consistent.

Administering the File System

Creating a File System and Making it Available

Once a disk is formatted, the next step is to define the file system. The **mkfs**(1M) command is used for this purpose. The **sysadm makefsys**(1) command can be used to define a file system on a diskette.

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 in neither format is the file system actually given a name; 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. The format of the minor is feature-dependent.

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. Figure 5-7 shows the recommended values to use with the **mkfs** command for devices supported on the system. 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 (check Appendix A). 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, rounding up if necessary to fill the final inode block.

Device	Gap Size	Blks/Cyl	Mfg
Floppy Disk Low Density	1	16	
High Density Floppy Disk	1	30	
182Mb ESDI Hard Disk	1	324	(CDC)
300Mb SCSI Hard Disk	1	414	(CDC)
390Mb ESDI Hard Disk	1	540	(CDC)
600Mb SCSI Hard Disk	1	765	(CDC)

Figure 5-7. Interrecord Gap and Blocks/Cylinder Recommendations

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 *System Administrator's Reference Manual*. Note that the **sysadm makesys** subcommand has no provision for the use of prototype files.

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.

Choosing Logical Block Size

Logical block size is the size of the chunks the operating 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** (1m) 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.

Summary: Creating and Converting File Systems

The following is a summary of the steps in creating a new file system or converting an old one to a new logical block size:

1. If the new file system is to be created on a disk partition where an old file system resides, backup the old system.
2. If the new file system is to be created from an old file system, run the **labelit** command, which reports the mounted file system name and the physical volume name of the old file system; see **volcopy** (1M). These labels are destroyed when you make the new file system, so you must restore them.
3. If the new file system is to be created from an old file system and the new file system will have a larger logical block size, then, because of fragmentation, the new file system will allocate more disk blocks for data storage than the old. Use the **fsba** (1M) command to find out the space requirements of the old file system with the new block size.

   Use the information you get from the **fsba** command to make sure that the disk partition to be used for the new file system is large enough. Use the **sledit**(1M) command to find out the size of your current disk partitions. If the new file system requires a disk repartition, see *Formatting and Partitioning* in Chapter 4.

4. Use the **mkfs** (1M), command with the **–b** option to make the new file system with the appropriate logical block size. The **mkfs** (1M) command is described in the section *Administering the File System* in this chapter. Also see Procedure 5.1.

5. Run the **labelit** command to restore the file system and volume names.

Relating the File System Device to a File System Name

A SYSTEM V/88 file system is generally referred to by the name of the highest level directory in its hierarchy. The file system shown in Figure 5-2 at the beginning of this chapter is called usr because that is the directory to which it is tied. Similarly, the root file system is called that because its first directory is **/root** (represented in the operating system parlance by a slash (**/**)). However, in the previous sections when the file system was created, the only name on the command line (other than the name of a prototype file, if you used that option) was the name of a special device file. There are several ways in which the file system name and the directory name can be tied together.

The first, and most explicit, is through the **labelit**(1M) command. **labelit** makes the connection between the device special file and the mounted name of the file system. It writes the name of the file system. i.e., highest level directory, into a field in the super-block. When **labelit** is used for removable file systems, such as those on diskette, one command line argument can be the identifying number of a volume. This number, too, is stored in a field in the super-block, but it is common practice to write it on a self-adhesive label that is attached to the diskette or tape that holds the file system.

The connection between the device and the file system name is also made by the **mount**(1M) command. This step is mandatory if the file system is to be available to users.

Mounting and Unmounting File Systems

For a file system to be available to users, the operating system has to be told to "mount" it. The root file system is always mounted as part of the boot procedure. The usr file system, which may be on the same disk device as root, is also automatically mounted as the system is being brought up to multi-user mode. The issuing of the **mount** command that brings these two file systems online is hidden in start-up shell procedures. Regardless of whether the **mount** command is hidden or not, its execution causes the mounted disk device and the mounted-on directory to be paired in an internal operating system table, called the mount table, and in the file **/etc/mnttab**. For example, the following command tells the system that **/dev/dsk/m323_0s2** contains a file system that begins at directory **/usr**:

```
# mount /dev/dsk/m323_0s2 /usr
```

NOTE

The **mount** command has other arguments. See the *System Administrator's Reference Manual* for details.

If you try to change directories (**cd**(1)) to a directory in **/usr** before the **mount** command is issued, the **cd** command will fail. Until the **mount** command completes, the system does not know about any of the directories beneath usr. True, there is a directory **/usr**. (it must exist at the time the **mount** command is issued), but the file system below that remains inaccessible until the **mount**.

It is common practice for small file systems to be contained completely on one diskette. A diskette can holds as many as 1276 512-byte blocks: close to 3/4 of megabyte. You can define file systems on diskette and use them either for storage or for live access. Using a diskette for live access has the following two disadvantages:

1. The access time is not as fast as the hard disk.
2. It ties up the diskette device.

It is more common for users to copy in such a file system to a directory on the hard disk. To do that, the file system must first be mounted. A user who plans to establish a file system that can be brought in from diskettes needs first to create two directories on the hard disk: one to serve as a mount point, one to be the target directory of the file system being brought in. For example, you have created the directories; the mount point is named **/hk**, and the target is named **/myfs**. You could bring a file system from diskette to hard disk with the following command sequence:

 # mount —r /dev/SA/diskette1 /hk

(*The* **—r** means read-only.)

 # cd/hk

find . —print | cpio —pdm /myfs

(*See* **find**(1) and **cpio**(1) in te *User's Reference Manual* for an explanation of the options used.)

The command for unmounting a file system requires only the name of the special device. After you have copied in a file system from a diskette, for example, you would issue the command:

#umount /dev/SA/diskette1

Unmounting is frequently a first step before using other commands that operate on file systems. For example, **fsck**(1M), which checks and repairs a file system, and **dcopy**(1M), which copies and compresses a file system, work on unmounted file systems. Unmounting is also an important part of the process of shutting the system down.

Summary

Thus far, the file systems have been described in the abstract; something of the way they are created, stored on disks, made available to the system, or removed from the system. The next portion of this chapter discusses how you maintain the integrity of an active file system.

Maintaining a File System

Once a file system is created and made available to users, it is always necessary to monitor how it is being used by the people in the organization. The distinction between a file system and its files may result in some confusion. The administrators view is more likely to be of the file system, while users tend to think and work in terms of files. When talking about the tasks involved in keeping file systems working smoothly for users, you must be ready to deal with users individual files as well as with the entire system.

Once a file system has been created and made available, there are several tasks routinely done to make certain that the file systems in regular use on a computer are providing the level of service and stability they should. They can be grouped into procedures for:

- checking for file system consistency
- monitoring disk usage
- compressing and reorganizing file systems
- backing up and restoring file systems

The Need for Policies

As with most other aspects of administering a computer, file system administration should be based on establishing a set of policies that are appropriate for your organization. There can be no hard-and-fast rules for such things as the size of file systems, the number of users in a file system, the way in which backups are done, the extent to which users can be allowed to keep inactive files in the system, or the amount of disk space a single user is entitled to occupy. These questions can only be resolved within the context of the organization. The number of users, the type of work they are doing, the number of files needed —all are variables. The responsible administrator must determine what best meets the needs of the organization.

Shell Scripts for File System Administration

Once policies have been agreed on, many of the routine tasks connected with file system administration can be incorporated in shell scripts. Monitoring disk usage, for example, can be handled through shell scripts that do the monitoring for you and transmit messages to the system console when exceptions are detected.

Here are a few ideas:

- Use a shell script running under **cron**(1M) control to investigate free blocks and free i-nodes and to report on file systems that fall below a given threshold.
- Use a shell script to do automatic clean-ups of files that grow.
- Use a shell script to highlight cases of excessive use of disk space.

Checking for File System Consistency

There is a separate section later in this chapter that describes **fsck**(1M), the file system checking utility. However, it is mentioned here because file system checking is central to the whole problem of normal file system maintenance.

Monitoring Disk Usage

You need to monitor the level of usage of a file system for the following reasons:

- If not watched regularly, the percentage of disk space used increases until the allocated space is used up.
- When the allocated space is used up, processes run very slowly, or not at all; the system spends its time putting out a message about being out of file space.
- There is a natural tendency for users to forget about files they no longer use so that the files just sit there taking up space.
- Some files grow larger as a result of perfectly normal use of the system. It is an administrative responsibility to keep them under control.

- Some directories, notably **/tmp**, accumulate files during the day. When the system is first brought up, **/tmp** needs to have enough free blocks to carry it through to **shutdown**(1M).

There are four tasks that are part of keeping disk space uncluttered:

1. monitoring percent of disk space used
2. monitoring files and directories that grow
3. identifying and removing inactive files
4. identifying large space users

The tasks mentioned here are shown in Chapter P5, *File System Administration Procedures*. In the procedures, the tasks are done through the System Administration menus. This section supplements the procedures in Part 1 and shows some operating system commands that you may use.

Monitoring Percent of Disk Space Used

Monitoring disk space may be done at any time to see how close to capacity your system is running. Until a pattern has emerged, it is advisable to check every day. In this example, the **df**(1M) command is used.

```
$ df –t
```

The **–t** option causes the total allocated blocks and i-nodes to display as well as free blocks and i-nodes. When no file systems are named, information about all mounted file systems displays.

Monitoring Files and Directories that Grow

Almost any system that is used daily has several files and directories that grow through normal use. Some examples are:

File	Use
/etc/wtmp	history of system logins
/usr/adm/sulog	history of **su** commands
/usr/lib/cron/log	history of actions of **/etc/cron**
/usr/lib/help/HELPLOG	actions of **/usr/bin/help**
/usr/lib/spell/spellhist	words that **spell**(1) fails to match

The frequency with which you should check growing files depends on how active your system is and how critical the disk space problem is. A good technique for keeping them down to a reasonable size uses a combination of **tail**(1) and **mv**(1):

```
$ tail –50 /usr/adm/sulog > /tmp/sulog
$ mv /tmp/sulog /usr/adm/sulog
```

This sequence puts the last 50 lines of **/usr/adm/sulog** into a temporary file, and then it moves the temporary file to **usr/adm/sulog**, thus effectively truncating the file to the 50 most recent entries.

See Appendix B for a description of files that grow and methods for controlling their growth.

Identifying and Removing Inactive Files

Part of the job of cleaning up heavily loaded file systems involves locating and removing files that have not been used recently. The commands you might use to do this work are shown below; the policy decisions involved are:

- How long should a file remain unused before it becomes a candidate for removal?
- Should users be warned that old files are about to be purged?
- Should the files be permanently removed or archived?

The **find**(1) command locates files that have not been accessed recently. **find** searches a directory tree beginning at a point named on the command line. It looks for files that match a given set of expressions, and when a match is found, performs a specified action on the file. This example barely begins to suggest the full power of **find**.

```
$ find /usr –type f –mtime +60 –print > /tmp/deadfiles &
```

Here is what the example shows:

/usr
: specifies the pathname where **find** is to start. Presumably, your machine is organized in such a way that inactive user files will not often be found in the root file system.

–type f
tells **find** to look only for regular files and to ignore special files, directories, and pipes.

–mtime +60
says you are interested only in files that have not been modified in 60 days.

–print
means that when a file is found that matches the **–type** and **–mtime** expressions, you want the pathname to be printed.

> /tmp/deadfiles &
directs the output to a temporary file and indicates that the process is to run in the background. This is a sensible precaution if your experience tells you to expect a substantial amount of output.

The **sysadm fileage**(1) command can be used to produce similar information (see Procedure 5.3).

Identifying Large Space Users

Here again the most important questions are not what commands to use to learn who is occupying excessive amounts of disk space, but instead policy questions concerned what the limits should be:

- On our system, what constitutes a reasonable amount of disk space for a user to need?
- If a user exceeds the normal amount by 25%, say, is it possible the users job requires extraordinary amounts of disk space?
- Is our system as a whole running short of space? Do our existing limits need to be reviewed?

Two commands produce useful information in this area: **du**(1M) and, once again, **find**(1).

du produces a summary of the block counts for files or directories named in the command line. For example, the following displays the block count for all directories in the usr file system:

```
$ du /usr
```

Optional arguments allow you to refine the output somewhat. For example, **du –s** may be run against each user's login to monitor individual users.

The **find** command can be used to locate specific files that exceed a given size limit:

```
$ find /usr –size +10 –print
```

This example produces a display of the pathnames of all files (and directories) in the usr file system that are larger than 10 (512-byte) blocks. Similar information can be produced by the **sysadm**(1) command (see Procedure 5.3).

File System Backup and Restore

The importance of establishing and following a file system backup plan is not appreciated until data is lost and cannot be recovered. Backing-up a file system takes time. Trying to recover lost or damaged data from paper records and best-guess-work takes even more time. The value of an effective system backup plan lies in the ability to recover lost or damaged data easily. It is important to consider and evaluate your system backup needs when forming this plan. You should reevaluate the plan as the use of the machine changes.

System Administration menus are provided to help you with complete and incremental file system backups to cartridge tape and diskette, and for restoring backup data to the hard disk (see Procedure 5.4). The capability to copy selected directories and files to tape is provided by using the **sysadm store**(1) command, the **find**(1) command with **cpio**(1), or using the **sysadm backup_rest**(1) command. The directories and files are read back to the hard disk by using **sysadm rest**(1) or the appropriate **cpio** command.

Overview of Backup and Restore

The backup and restore system is a collection of menus listing items that present possible actions. You choose one of the available actions by entering its item number. If more information is needed, additional menus or questions are presented until the action is finally done.

The menus are organized in an inverted tree structure similar to the SYSTEM V/88 directory structure. Each level has a QUIT menu item that returns you to the previous level. The topmost QUIT exits the backup and restore system.

The menus use no special terminal features, working equally well on video or print-oriented devices. The only requirement is about 20 visible lines of text and about 64 columns in each line. This allows you to use any kind of terminal to perform backups.

Each menu is a list of numbered items. Generally, item 0 is labeled QUIT and is the last item in the menu list. Select an item by entering its number. Non-numeric input or numbers out of range are rejected. The "interrupt" key can also be used to quit.

Some actions request a yes or no response before continuing. In these cases, "YES" is any input that begins with **Y** or **y**. Any other response is interpreted as "NO". An empty line causes the question to be asked again until a response is entered.

Occasionally, you are asked to enter a string or name for future use. Type the entry as requested, then press **<RETURN>**. Any special meaning of the input is explained by the program.

Before actually using backup and restore, try taking a "test-drive". Invoke backup and restore (see *Starting Up the System*) and work through this section of the manual and Procedures 5.4 and 5.5, which contain important details about **backup_rest**. Try out everything that does not require a functioning archive device; change some preferences, use different base directories. You can even choose a BACKUP item, since they always ask for confirmation; just enter **no** when asked. This preliminary exercise lets you see what the menus really look like while giving you an understanding of the overall backup and restore system. You can always choose the QUIT menu item to come back to the previous level or to cancel some requested action. You can do no harm unless you actually perform a backup or restore, or you change the archive description to senseless values and then save it (although you can always recover a default description from the library).

Starting Up the System

To invoke backup and restore, type:

br

If you are not logged in as root, you are prompted for a password, since this command changes you to the user ID **br**. You can log in directly as the user **br** or use **su - br** if you are already logged in.

You can also enter **br** via the **sysadm** program or the **sysadm** login. **backup_rest** is in the File Management Menu. To go directly to the program, execute or log in as **sysadm backup_rest**.

A prerequisite for running **br** is system run level S. This run level should have all TTY ports disabled except for the system console. This prevents backups from occurring while the file systems are active, which could invalidate or confuse some activities, like incremental backups. If **br** detects that the system is currently at run level S, it automatically executes **/etc/mountall** to be sure that all necessary file systems are mounted before continuing. **br** checks the run level and warns you if it is not at run level S. You are not prevented from proceeding, however, since it is presumed that you know the consequences. Other actions may reiterate the warning prior to an action that would be affected by nonquiescent file systems.

Preparing for File System Backup and Restore

Preferences

Some features of **backup_rest** can be changed by the user. These features are listed under menu item 6, PREFERENCES in the **backup_rest** main menu:

```
Select new preferences:
  1. Disable WRITE-VERIFY of new archive
  2. Enable  LOGGING of full and incremental backups
  3. Enable  INCREMENTAL verbosity
  4. Enable  FULL verbosity
  5. Enable  ESTIMATE verbosity
  6. Enable  VERIFY verbosity
  7. Change time for incrementals from:  02:00
  8. Save new preferences
  0. QUIT this level
-- Your choice? [0-8]
```

Examine this menu and make necessary changes before performing a backup or restore. This is especially important for menu item `1, Disable WRITE-VERIFY of new archive`. This pass can nearly double the time required to make a backup, although it greatly enhances the reliability of the new archive. By default, WRITE-VERIFY passes are made; type `1` to disable this feature.

The verbosity flags (`3` through `6`) affect the particular functions by telling **bru**(1) to give a running update of its progress. Specifically, it gives **bru** the **-v** flag. Only one level of verbosity is specified.

The LOGGING preference (`2`) causes a copy of **bru**s output to be saved in a log file. This can only be done for full backups and attended or unattended incremental backups, and only if the corresponding verbosity has been enabled. Unattended scheduled backups also mail their error output to root. The log files are:

/backups/files/LogIncr for incremental backups
/backups/files/LogFull for full backups

If LOGGING is enabled, each incremental backup overwrites any existing incremental log file. The log file for full backups is overwritten only if you start over with the complete list of file systems. As each file system is backed up, any output is appended to the log file. If the FULL verbosity flag (`4`) is also enabled, a very large log file (large enough to run out of space on "/") can result. If you only want to log a few of the backups, quit the full backup, change the preference, then resume the full backup. The state of the full backup is saved when you quit in the middle, so you do not lose any work.

Menu item `7` lets you change the time at which incremental backups occur. If you choose this item, you are asked to enter a new time in 24-hour format. In this format, 12 midnight is 00:00, 12 noon is 12:00, 5 p.m. is 17:00, 10 p.m. is 22:00. Note that you must enter a leading zero for times between 00:00 and 09:59. A certain amount of range checking is done and illegal values discarded, but it is still possible to enter an invalid time. If you do this, the scheduled backups may not occur when you expect, or they may not occur at all.

Menu item `8` saves the changes that you make. Until you save them, the changes are only kept as potential new values.

Describing Your System

The **backup_rest** subcommand needs a description of your system and a range of possible archive devices (see Procedure 5.5) to execute a proper backup or restore. Before using **backup_rest** to actually back up or restore your system, you must change these descriptions to accurately reflect your system configuration.

Information about your system's disk configuration is located in the description file **/backups/files/DiskInfo**. **DiskInfo** is an ordinary text file with one file system described per line. Its initial contents describe a minimal system as it arrives from the factory. If you add, change, or remove file systems from this initial configuration, you must make the corresponding changes in **DiskInfo**.

The **DiskInfo** file should describe every file system and every partition on every disk in the system. However, its most important use is to describe those file systems which will be backed up during a full backup. Each line in the file describes one file system or partition (device). Although you need only include those which you want to back up during a full backup, you should describe all disks, even if they are not currently in use. The order of lines in the file is not significant, but it is convenient to order them by increasing device number.

```
dev-name        mount-point    backup
/dev/root       /       f
/dev/usr        /usr    f
/dev/01s0       /1.2/src       f
mktg.RECORDS    /mktg   x
/dev/01s2       /src    -
```

The three columns in a **DiskInfo** file are:

dev-name
The block-special device name of the file system or partition. For remote resources this is the remote resource name.

mount-point
The "mount point" or directory where the file system is normally mounted.

backup
The indication of when and how to back up a file system.

f
The item is backed up for both full and incrementals.

x
The item is backed up during full backups, but *is not* backed up during incrementals. This is typically done for remote resources that you want to fully back up from a single machine but are incrementally backed up individually on their host machine.

-
The item is not backed up for either fulls or incrementals. This is usually used for read-only file systems.

Each line of a **DiskInfo** file is made up of words separated by white space (blanks or tabs). Each word is composed of letters, digits, and symbols. Any span of characters uninterrupted by white space is considered a word. Words are parsed from the lines by finding intervening white space; therefore, any ignored or irrelevant fields must still be present. Normally, this is done by entering a hyphen (-) to serve as a place holder.

The **DiskInfo** file that comes with your system describes only the minimal system. After you have decided which file systems will be mounted and have described these in **/etc/fstab**, enter the equivalent descriptions into **DiskInfo**. You could write a simple **awk** script that examines **fstab** and produces a corresponding **DiskInfo** file. Since the root file system "/" is never described in **fstab**, make sure it is in **DiskInfo**.

Archive Devices

The **backup_rest** facility is capable of using nearly any archive device. The archive device is the place where backups are written and restorals are read. The archive device is often a tape drive but could be any other device or connection attached to your computer system, perhaps even a network. Most archive devices have several common characteristics:

device name
: the SYSTEM V/88 path name of the device

media size
: the number of bytes that can be written on each unit of media. For devices without removable media, this is the largest archive size that the device can hold. For removable media, it is the size of each volume in a multivolume archive.

"blocking size"
: the size of chunks written to or read from the media. The smallest acceptable value is 2K; the largest is limited only by the address space of a process. You may need to try several blocking sizes before finding the right combination that maximizes your system's throughput.

The choice of blocking size affects the usable space on the archive medium. Since data is read or written in chunks, only an integral number of chunks are allowed on a single archive. For example, if the archive media size was 5.2Mb, and you chose a blocking size of 1Mb, then .2Mb of every archive would be unused. This is automatically calculated and accounted for, so you need only be concerned with tuning for optimal speed vs. media usage.

As stated previously, all portions of backup and restore use a single designated archive device set from the **backup_rest** main menu. The description of this device is found in the **/backups/files/Archive**, an ordinary text file in a special format. To see this file, select menu item 5, ARCHIVE description change from the main menu. You can examine the archive description without changing it.

The Archive Description Library

Because you can use many different kinds of archive devices with **backup_rest**, it is convenient to describe them as a group and allow selection of one. This selection is done by choosing menu item 2, **READ all-new values from library** from the ARCHIVE Description change menu. The library descriptions are found in the archive description library file, **/backups/files/ArchiveLib**. This file is an ordinary text file with a single archive device description on each line. Depending on your exact configuration, you may not have some of the devices described in the library. *Do not* choose a device that you do not have.

Each description refers to a particular combination of device name and control parameters. For example, a 9-track tape unit can be given in several different descriptions with the same device name but different media sizes for various-sized tape reels.

The following is a sample description library:

```
Floppy /dev/rSA/diskette1    -b 128k-s 640K
Slow-Tape      /dev/rmt0     -b 64k -s 36m -p
Faster-Tape    /dev/rst0     -b 128k-s 59m -p
```

As in the **DiskInfo** description, items are separated by white space. The first item is a descriptive name for the device. It is used when presenting certain menus including the **READ all-new values from library** menu. The description cannot contain embedded blanks, so use hyphens (or underscores) to separate words, as shown in the example. The second item is the actual SYSTEM V/88 device name of the device. To determine this name, you must be familiar with SYSTEM V/88 devices. To avoid unwanted side effects, the device should be the "raw" device, if possible. The remaining items describe the control parameters for the device. They are unaltered **bru**(1) options and can be in any order. Recall that **-b** introduces the blocking size, **-s** introduces the media size, and **-p** means that the device is "tape-like".

The default library value describes the normal archive devices available on a minimal system. You should examine the library file to see what is there. If you add device descriptions to the library, be sure they are well tested; otherwise, you will be unable to make meaningful backups.

NOTE

The check for proper run level is not limited to run level S; **/backups/cmds/common/SiteValues** defines the expected run level, which defaults to S. You can edit the **SiteValues** file if necessary. The shell variable defining the expected run level is named **SITE_RUN**.

Also defined in **SiteValues** is a default archive device to use if the archive description file is destroyed. You can change this value if necessary.

Full Backup

The full backup provided by the System Administration backup facilities copies directories and files of a specified mounted file system to cartridge tape or diskette (see Procedure 5.4). Full backups plus incremental backups combine to form a unified backup strategy. Full backups can be done without intervening incremental backups, but incremental backups must be preceded by at least one full backup to provide the base.

If the backup medium is diskettes, you must have enough formatted diskettes to contain the directories and files being backed up. Each file system is backed up to a separate (possibly multivolume) archive. As a rule of thumb, the number of blocks of storage needed for a full backup is equal to the number of data blocks used in the file system multiplied by 1.5. For example, a file system that occupies 13,294 blocks requires 16 diskettes for a full backup (at 1200 blocks per diskette).

The time required to do a full backup of a file system is about three minutes per diskette. The 13,294 block file system is estimated to require 16 diskettes, thus, takes about one hour to back up.

If the backup medium is cartridge tape, one tape can contain a file system of about 100,000 (512-byte) blocks. The time required for backups to tape is less than for diskettes; a full tape can be written in about 35 minutes.

Deciding when to do a new full backup is critical in a system backup plan. The following questions should be considered:

- If the backup medium is diskettes, how many diskettes do you want to dedicate to incremental backups? To illustrate, if incremental backups are made daily and full backups once a month, over 80 diskettes could be required for backing up the 13,294 block file system described previously.
- How much time do you want to devote to full backups? Less media is required to maintain a backup library the more frequently a full backup is done. A full backup, however, may take 8 or 10 times as long as any single increment.
- How much time do you want to devote to incremental backups? Each incremental is relative to the last full backup, and thus each incremental takes longer to perform as the volume of changes increases until the next full backup.
- What is the approximate rate of change in the file system under scrutiny? Do changes occur throughout the file system or in a small percentage of the files? If change is frequent and widespread, it may be best to schedule full backups more often (reducing the media required for incrementals). If change is concentrated in a few files, perhaps selective backup should be a part of the overall plan.

It is strongly recommended that a full backup of the system be performed as soon as you have configured it to your satisfaction (added users, enabled TTYs, set up **uucp**, set up printers). It is wise to store this backup in a safe place so that it (instead of the release media) can be used to completely restore the base system should all other backups be unavailable.

Incremental Backup

The incremental backup provided by the System Administration backup facilities copies files that have changed since the last full backup to cartridge tape or diskettes (see Procedure 5.4).

Files for incremental backup are found by comparing their modification date with the date of the last full backup. Files with modification dates more recent than the last full backup are included in the incremental. Changing the modes of a file with **chmod**(1) does not affect the modification date. Similarly, renaming or moving a file (**mv**(1)) anywhere on the same file system does not change its modification date. To force such files to be found for incremental backup, use the **touch**(1) command.

The `ESTIMATE archive requirements` selection in the INCREMENTAL backup menu may take some time to complete because it searches the entire selected directory tree for files to be backed up (even though they are not written to the archive). By default, there is no screen indication of progress, but you can set a personal preference flag so that estimates tell you of their progress (see *Preferences* in this chapter).

Choose item 3 and an incremental backup will be scheduled to occur automatically at a preset time in the future (changed under PREFERENCES). If the archive device is properly set up, an incremental backup will occur at the future time. In this way you can arrange for a backup, then have it actually occur when the system is quiescent. Such backups must be individually scheduled.

Before the scheduled time occurs, you must be sure that the archive device is ready to write an archive. Once scheduled, the incremental backup will proceed without any user confirmation or waiting for the archive device to become ready. Errors and messages that would normally appear on the display are mailed to root after the backup process completes. If an error prevents the backup from actually taking place, this information is also mailed to root.

If no incremental estimate has been done since the last incremental backup, an estimate is made before the backup is scheduled. If the estimate requires more than one archive volume, then the unattended backup is not scheduled, and you must have to perform the incremental backup yourself, using the indicated number of archive volumes. If this happens frequently, you should probably do full backups more often.

Unattended backups are always scheduled for the same time of day, by default, 2:00 a.m. The **backup_rest** main menu item 6, PREFERENCES allows you to change this time (see *Preferences* in this chapter). Changing the preferred time only affects backups that are scheduled after the change, not backups that have already been scheduled but have yet to occur.

When you schedule an unattended backup, it occurs at the next opportunity for the given time, i.e., if the scheduled time of day is less than the current time, it means tomorrow. If the scheduled time is greater than the current time, it means today. Time comparisons are based on the 24-hour format.

NOTE

Not all changed directories and files are copied in an incremental backup. The contents of **/etc/save.d/except** specifies files and directories that *are not* copied. A typical **/etc/save.d/except** file is shown in Figure 5-8.

The incremental file system backup is faster than a than a full backup because only files that have changed since the last full backup (complete or incremental) are being collected. However, while incremental backups save time, restoring from incremental backups can result in your giving back all the time saved and tie up a sizable number of diskettes.

```
#  Patterns of filenames to be excluded from saving by savefiles.
#  These are ed(1) regular expressions.
/.news_time$
/.yesterday$
/a.out$
/core$
/dead.answer$
/dead.letter$
/ed.hup$
/nohup.out$
/tmp/
^/etc/mnttab$
^/etc/save.d/timestamp/
^/etc/utmp$
^/etc/wtmp$
^/usr/adm/
^/usr/at/
^/usr/crash/
^/usr/dict/
^/usr/games/
^/usr/learn/
^/usr/news/
^/usr/spool/
^/usr/tmp/
```

Figure 5-8. Sample **/etc/save.d/except** File

Selected Backup

The selected backup function lets you back up files and directories of your choice. Basically, you provide a base directory and selection criteria, then any descendant files of that directory that match the criteria are saved in the archive. The selection criteria can be a name-matching pattern (in the style of the shell), a file modification date more recent than a certain point in time, a specific owner, or any of a variety of other criteria (see *bru Options*).

bru Options

The heart of **backup_rest** is the **bru**(1) command. It reads and writes all archives. Normally, the only time you directly interact with **bru** is when you are doing a multivolume backup or restore, and **bru** asks you to insert the next volume. The rest of the backup and restore system keeps track of **bru**s actions, remembering what happened (or did not happen), as well as such information as the current archive description, lists of file systems.

At this time, you must directly enter **bru** options to perform selected backups or restores (other than full restores). Future versions will present a more descriptive menu interface. The **bru**(1) entry in the *User's Reference Manual* lists the available file selection options (including many not described here) and gives some examples of their use. The following lists describe the most useful options. For options which require an argument, the argument must be separated from the option flag by one or more blanks. Control options are:

-w

Wait for your confirmation from the terminal. For each file, **bru** prints the file name and the action to be taken, then waits for your confirming input. Any entry beginning with **y** causes the action to occur. Any other input (including blank lines) prevents the action, and **bru** will move to the next file. This option is most useful when files to be saved or restored are scattered around the directory hierarchy. Using **-w** permits you to save or restore only the needed files, instead of a full directory subtree.

File selection options are:

-n *date*
Select only those files newer than *date*. The *date* is a string given in one of the following forms:

String	Example
DD-MMM-YY[,hh:mm:ss]	**12-Mar-84,12:45:00**
MM/DD/YY[,hh:mm:ss]	**3/12/84**
MMDDhhmm[YY]	**0312124584**
pathname	**/usr/lib/LastBackup**

The time of day is optional in the first two forms (as shown by the square brackets, which are never actually entered). In the third form, the year is optional and is assumed to be the current year if absent. In the fourth form, the named file's modification date is used as the date, allowing you to save or restore only those files modified since another file was changed.

-o *user* Select only those files owned by *user*. The user string may be:

- A user login name, as found in the password file.
- A decimal number indicating the user ID number.
- The pathname of a file, in which case the owner of that file is used.

-u *flags* When restoring files, unconditionally extract the files specified by flags, regardless of their modification times. Normally, if an archived file is older than its corresponding existing file, the archived version is *not* extracted. This option overrides this behavior. The flags select those files to be unconditionally restored:

b block-special files
c character-special files
d directories
p FIFOs (named pipes)
r regular files

Selection of directories (**d**) only means that their attributes are modified to reflect those saved on the archive. Existing directories are never overwritten.

If more than one file selection criterion is given, a file is saved or restored only if all criteria are met. For example, the following restores from the archive all files owned by 'gg' that are also newer than March 12, 1984:

```
-n 3/12/84 -o gg -u r
```

These files are unconditionally restored, even if they would overwrite an existing file that had been modified more recently.

In addition to the selection options, you can specify file name patterns that must be matched before a file is saved or restored. These patterns are in the same style as the shell, described in **sh**(1). Do not use quotes around the patterns unless you really intend for the quotes to be a literal part of the pattern. The shell is not allowed to expand any of the patterns you enter as file selection criteria. If the pattern matches a directory, the directory and all of its descendants are saved to or extracted from the archive according to the other selection criteria.

Backup Schedule Reminder

A feature of the System Administration FILE MANAGEMENT menu is a way to create reminders for yourself about your backup schedule. The backup schedule reminder feature lets you do the following:

1. Schedule reminder messages that say, in effect, "If the machine is shut down within this time range, send a message to the console to do a backup of this or that file system."
2. Schedule reminder checks that say, "If it gets to be this time and the machine has not been shut down, take a look to see if any reminder messages would have been sent had the machine been shut down."

This reminder mechanism does not relieve you of the job of working out a reasonable schedule of backups. It merely nudges you about the schedule you have set up. Procedure 5.4 shows the steps you follow to use the reminder feature. You can, if you prefer, set up reminder notices using **cron**(1M) and **crontab**(1).

Restoring a File System from Backup

You can restore directories and files from system backups for a single file, a directory structure, or all files of a system backup. Use the **sysadm restore**(1) command for all three types (see Procedure 5.4 for examples).

To restore a file system using operating system commands, change to the appropriate directory and execute to take input from the cartridge tapes:

```
$ cpio –icdumv < /dev/rmt/ctape
```

The tapes used must be the same ones that were used for backup, and they must be presented in the same order (see the discussion on *Selected Backup*).

What Can Go Wrong With a File System

Most of the things that can corrupt a file system have to do with the failure of the correct pointer and count information to make it out to the storage medium. This can be caused by:

- hardware failure
- program interrupts
- human error
- a combination of hardware/program failures and incorrect procedures

Hardware Failure

There is no effective way of predicting when hardware failure will occur. The best way to avoid it is to be sure that recommended diagnostic and maintenance procedures are followed conscientiously. There is a utility that reports bad blocks on a hard disk and procedures for assigning alternate areas for these blocks (see Chapter 4 *Disk Management*).

Program Interrupts

It is possible that errors that cause a program to fail might result in the loss of some data. It is not easy to generalize about this because the range of possibilities is so large. Perhaps the best thing to say is that programs should be exhaustively tested before they are put into production with valuable data.

Human Error

Probably, the main cause of file system corruption falls under this heading. There are four rules that should be followed by anyone who manages file systems:

1. *always* check a file system before mounting it. Nothing complicates the problem of cleaning up a corrupt file system so much as allowing it to be used when it is bad.
2. *never* remove a file system physically without first unmounting it.
3. *always* use the **sync** command before shutting the system down and before unmounting a file system.
4. *never* physically write-protect a mounted file system, unless it is mounted "read only."

The random nature of all these mishaps simply underscores the importance of establishing and observing good backup practices. It is the most effective form of insurance against data loss.

How to Check a File System for Consistency

When the operating system is brought up, a consistency check of the file systems should always be done. This check is automatically done as part of the power-up process. Included as part of that process is the command **fsstat**(1M). **fsstat** returns a code for each file system on the hard disk indicating whether the consistency checking and repair program, **fsck**(1M), should be run.

These same commands or **sysadm checkfsys**, should be used to check file systems not mounted routinely as part of the power-up process. If inconsistencies are discovered, corrective action must be taken before the file systems are mounted. The remainder of this section is designed to acquaint you with the command line options of the **fsck** utility, the type of checking it does in each of its phases, and the repairs it suggests.

File system corruption, while serious, is not very common. Most of the time a check of the file systems finds everything all right. So much emphasis is placed on file system checking because if errors are allowed to go undetected, the loss can be substantial.

fsck Utility

The file system check (**fsck**) utility is an interactive file system check and repair program. Procedure 5.3 shows the steps required to run **fsck** with the **sysadm checkfsys** command. **fsck** uses the information carried in the file system itself to perform consistency checks. If an inconsistency is detected, a message describing the inconsistency displays. You may elect to have **fsck** either make the repair or not. The reason you might choose to have **fsck** ignore an inconsistency is that you judge the problem to be so severe that you want either to fix it yourself using the **fsdb**(1M) utility, or you plan to go back to an earlier version of the file system. The decision to have **fsck** ignore inconsistencies and then do nothing about them yourself is dangerous. File system inconsistencies do not repair themselves. If they are ignored, they only get worse.

fsck Command

The **fsck** command is used to check and repair inconsistencies in a file system. With the exception of the root file system, a file system should be unmounted while it is being checked. The root file system should be checked only when the computer is in run level S and no other activity is taking place in the machine.

The following is the general format of the **fsck** command:

fsck [**–y**] [**–n**] [**–s***x*] [**–S***x*] [**–t***file*] [**–q**] [**–D**] [**–f**] [**–b**] [*fsdevice*

where:

–y
specifies a "yes" response for all questions. This is the normal choice when the command is being run as part of a shell procedure. It generally causes **fsck** to correct all errors.

–n
specifies a "no" response for all questions. **fsck** will not write the file system.

–b
reboots the system if the file system being checked is the root (*/*) file system and changes have been made by **fsck**. If only minor, fixable damage is found, the file system is remounted.

–s*X*

specifies an unconditional reconstruction of the free list. Following the reconstruction of the free list, the system should be rebooted so that the in-core copy of the super-block is updated (see the **–b** option). The *X* argument specifies the number of blocks-per-cylinder and the number of blocks to skip (rotational gap). The default values are those specified when the file system was created. The formats for some common disk drives are as follows for 1K file systems. See the *Using mkfs* section of this chapter for more information.

Device	**–s***Blocks/Cylinder:Gap*
182M Hard Disk	-s324:1

–S*X*

specifies a conditional reconstruction of the free list, to be done only if corruption is detected. The format of the *X* argument is the same as described above for the **–s** option.

–t *file*

specifies a scratch file for use in case the file system check requires additional memory. If this option is not specified, the process asks for a filename when more memory is needed.

–q

specifies a "quiet" file system check. Output messages from the process are suppressed.

–D

checks directories for bad blocks. This option is used to check file systems for damage after a system crash.

–f

specifies that a fast file system check be done. Only Phase 1 (check blocks and sizes) and Phase 5 (check free list) are executed for a fast check. Phase 6 (reconstruct free list) is run only if necessary.

–r *file*

specifies a file containing file system block numbers (one per line) that have been redirected (because the track they are on has been redirected) and are now corrupted (have been zeroed).

fsdevice
names the special device file associated with a file system. If no device name is specified, **fsck** checks all file systems named in **/etc/checklist**.

Sample Command Use

The command line in the following example shows **fsck** being entered to check the root file system. The system response means that no inconsistencies were detected. The command operates in phases, some of which are run only if required, or in response to a command line option. As each phase is completed, a message displays. At the end of the program a summary message displays showing the number of files (i-nodes), blocks, and free blocks.

```
# fsck /dev/dsk/m323_0s0
  /dev/dsk/m323_0s0
  File System: root Volume: root

  ** Phase 1 - Check Blocks and Sizes
  ** Phase 2 - Check Pathnames
  ** Phase 3 - Check Connectivity
  ** Phase 4 - Check Reference Counts
  ** Phase 5 - Check Free List
  289 files 6522 blocks 3220 free
#
```

File System Components Checked by fsck

Before discussing the **fsck** phases and the messages that may appear in each, it is important to review the components of a SYSTEM V/88 file system and to describe the kinds of consistency checks that are applied to them.

Super-Block

The super-block is vulnerable because every change to the file system blocks or i-nodes modifies the super-block. If the CPU is halted, and the last command involving output to the file system is not a **sync** command, the super-block is almost certainly corrupted. The super-block can be checked for inconsistencies involving:

- file system size
- i-node list size
- free-block list
- free-block count
- free i-node count

File System Size and i-node List Size

Total file system size must be greater than the number of blocks used by the super-block plus the blocks used by the list of i-nodes. The number of i-nodes must be less than 65,535. While there is no way to check these sizes, **fsck** can check that they are within reasonable bounds. All other checks of the file system depend on the reasonableness of these values.

Free-Block List

The free-block list starts in the super-block and continues through the free-list blocks of the file system. Each free-list block can be checked for:

- list count out of range
- block numbers out of range
- blocks already allocated within the file system

A check is made to see that all the blocks in the file system were found.

The first free-block list is in the super-block. The **fsck** program checks the list count for a value of less than 0 or greater than 50. It also checks each block number to make sure it is within the range bounded by the first and last data block in the file system. Each block number is compared to a list of already allocated blocks. If the free-list block pointer is not 0, the next free-list block is read and the process is repeated.

When all the blocks have been accounted for, a check is made to see if the number of blocks in the free-block list plus the number of blocks claimed by the i-nodes equals the total number of blocks in the file system. If anything is wrong with the free-block list, **fsck** can rebuild it leaving out blocks already allocated.

Free-Block Count

The super-block contains a count of the total number of free blocks within the file system. The **fsck** program compares this count to the number of blocks it found free within the file system. If the counts do not agree, **fsck** can replace the count in the super-block by the actual free-block count.

Free i-node Count

The super-block contains a count of the number of free i-nodes within the file system. The **fsck** program compares this count to the number of i-nodes it found free within the file system. If the counts do not agree, **fsck** can replace the count in the super-block by the actual free i-node count.

i-nodes

The list of i-nodes is checked sequentially starting with i-node 1 (there is no i-node 0). Each i-node is checked for inconsistencies involving:

- format and type
- link count
- duplicate blocks
- bad block numbers
- i-node size

Format and Type

Each i-node contains a mode word. This mode word describes the type and state of the i-node. I-nodes may be one of five types:

- regular
- directory

- block special
- character special
- fifo (named pipe)

If an i-node is not one of these types, it is illegal. I-nodes may be in one of three states: unallocated, allocated, and partially allocated. This last state means an incorrectly formatted i-node. An i-node can reach this state if, for example, bad data are written into the i-node list through a hardware failure. The only corrective action **fsck** can take is to clear the i-node.

Link Count

Each i-node contains a count of the number of directory entries linked to it. The **fsck** program verifies the link count of each i-node by examining the total directory structure, starting from the root directory, and calculating an actual link count for each i-node.

If the link count stored in the i-node and the actual link count determined by **fsck** do not agree, the reason may be:

- stored count not 0, actual count 0

  No directory entry appears for the i-node.

  fsck can link the disconnected file to the **lost+found** directory.

- stored count not 0, actual count not 0, counts unequal

  Directory entry possibly removed without i-node update.
  fsck can replace the stored link count with the actual link count.

Duplicate Blocks

Each i-node contains a list of all the blocks claimed by the i-node. The **fsck** program compares each block number claimed by an i-node to a list of allocated blocks. If a block number claimed by an i-node is on the allocated-blocks list, it is put on a duplicate-blocks list. If the block number is not on the allocated-blocks list, it is put there. If a duplicate-blocks list develops, **fsck** makes a second pass of the i-node list to find the other i-node that claims the duplicated block. While there is not enough information available to determine which i-node is in error, most of the time the i-node with the latest modification time is correct. This condition can occur by using a file system with blocks claimed by both the free-block list and by other parts of the file system.

A large number of duplicate blocks in an i-node may be caused by an indirect block not being written to the file system. The **fsck** program prompts the user to clear both i-nodes.

Bad Block Numbers

The **fsck** program checks each block number claimed by an i-node for a value lower than that of the first data block or greater than the last block in the file system. If the block number is outside this range, the block number is bad.

If there are many bad block numbers in an i-node, it may be caused by an indirect block not being written to the file system. The **fsck** program prompts the user to clear the i-node.

NOTE

A certain amount of semantic confusion is possible here. A bad block number in a file system is not the same as a bad (that is, unreadable) block on a hard disk.

i-node Size

Each i-node contains a 32-bit (4-byte) size field. This size shows the number of characters in the file associated with the i-node. A directory i-node within the file system has the directory bit set in the i-node mode word. The directory size must be a multiple of 16 because a directory entry contains 16 bytes (2 bytes for the i-node number and 14 bytes for the file or directory name).

If the directory size is not a multiple of 16, **fsck** warns of directory misalignment. This is only a warning because not enough information can be gathered to correct the misalignment.

For a regular file, a rough check of the consistency of the size field of an i-node can be performed by using the number of characters shown in the size field to calculate how many blocks should be associated with the i-node, and comparing that to the actual number of blocks claimed by the i-node.

The Algorithm

The **fsck** program calculates the number of blocks that should be in a file by dividing the number of characters in an i-node by 512 (the number of characters per block) and rounding up. One block is added for each indirect block associated with the i-node.

If the actual number of blocks does not match the computed number of blocks, **fsck** warns of a possible file-size error. This is only a warning. Logical blocks can be created in random order, and the operating system does not fill them in. A check of the file would be required to tell if the error is real or not (see the section on *Holes in Files*, earlier in this chapter).

Indirect Blocks

Indirect blocks are owned by an i-node. Therefore, inconsistencies in an indirect block directly affect the i-node that owns it. Inconsistencies that can be checked are:

- blocks already claimed by another i-node
- block numbers outside the range of the file system

The consistency checks described under *Duplicate Blocks* and *Bad Block Numbers* above are performed for indirect blocks as well as for the direct blocks of an i-node.

Directory Data Blocks

Directories are distinguished from regular files by an entry in the mode field of the i-node. Data blocks associated with a directory contain the directory entries. Directory data blocks are checked for inconsistencies involving:

- directory i-node numbers pointing to unallocated i-nodes
- directory i-node numbers greater than the number of i-nodes in the file system
- incorrect directory i-node numbers for "." and ".." directories
- directories disconnected from the file system

If **–r** is specified, **fsck** can identify directories which have been corrupted due to known damage to a particular block(s).

Directory Unallocated

If a directory entry i-node number points to an unallocated i-node, **fsck** can remove that directory entry. This condition occurs if the data blocks containing the directory entries are modified and written out while the i-node is not yet written out.

Bad i-node Number

If a directory entry i-node number is pointing beyond the end of the i-node list, **fsck** can remove that directory entry. This condition occurs if bad data is written into a directory data block.

Incorrect "." and ".." Entries

The directory i-node number entry for "." should be the first entry in the directory data block. Its value should be equal to the i-node number for the directory data block.

The directory i-node number entry for ".." should be the second entry in the directory data block. Its value should be equal to the i-node number for the parent of the directory entry (or the i-node number of the directory data block if the directory is the root directory). If the directory i-node numbers for "." and ".." are incorrect, **fsck** can replace them with the correct values.

Disconnected Directories

The **fsck** program checks the general connectivity of the file system. If directories are found not to be linked into the file system, **fsck** links the directory back into the file system in the **lost+found** directory. This condition can be caused by i-nodes being written to the file system with the corresponding directory data blocks not being written to the file system. When a file is linked into the **lost+found** directory, the owner of the file needs to be told about it.

Regular Data Blocks

Data blocks associated with a regular file hold the file's contents. **fsck** does not attempt to check the validity of the contents of a regular file's data blocks. It can, however, identify files that have been corrupted due to known damage to a particular block or blocks (**–r**). At present, **fsck** cannot identify the byte offset or extent of the corruption in the file.

Running fsck

The **fsck** program runs in phases. Each phase reports errors it detects. If an error is one that **fsck** can correct, the user is asked if the correction should be made. This section describes the messages that are produced by each phase.

The following abbreviations are used in the **fsck** error messages:

BLK	block number
DUP	duplicate block number
DIR	directory name
MTIME	time file was last modified
UNREF	unreferenced

Table 5-3 shows the single-letter abbreviations, used in the messages shown in the pages that follow, are replaced by the corresponding value when the message appears on your screen:

Table 5-3. Error Message Abbreviations in **fsck**

B	block number
F	file (or directory) name
I	i-node number
M	file mode
O	user-id of a file's owner
S	file size
T	time file was last modified
X	link count,
or	number of BAD, DUP, or MISSING blocks
or	number of files (depending on context)
Y	corrected link count number
or	number of blocks in file system (depending on context)
Z	number of free blocks

Initialization Phase

Command line syntax is checked. Before the file system check can be performed, **fsck** sets up some tables and opens some files. The **fsck** program terminates on initialization errors.

General Errors

Three error messages may appear in any phase. While they seem to offer the option to continue, it is generally best to regard them as fatal, end the run, and investigate what may have caused the problem.

`CANNOT SEEK: BLK` *B* `(CONTINUE?)`
The request to move to a specified block number *B* in the file system failed. The occurrence of this error condition indicates a serious problem (probably a hardware failure) that may require additional help.

`CANNOT 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 (probably a hardware failure) that may require additional help.

`CANNOT WRITE: BLK` *B* `(CONTINUE?)`
The request for writing a specified block number *B* in the file system failed. The disk may be write-protected.

Meaning of Yes/No Responses

An n(no) response to the **CONTINUE?** prompt says:

> Terminate program.
> (This is the recommended response.)

A y(yes) response to the **CONTINUE?** prompt says:

> Attempt to continue to run file system check.
> Often, however, the problem persists. The 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.

Phase 1: Check Blocks and Sizes

This phase checks the i-node list. It reports 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
- checking i-node format

Types of Error Messages —Phase 1

Phase 1 has three types of error messages:

1. information messages
2. messages with a **CONTINUE?** prompt
3. messages with a **CLEAR?** prompt

There is a connection between some information messages and messages with a **CONTINUE?** prompt. The meaning of the **CONTINUE?** prompt generally is that some limit of tolerance has been reached.

Meaning of Yes/No Responses —Phase 1

In Phase 1, an n(no) response to the `CONTINUE?` prompt says:

Terminate the program.

In Phase 1, a y(yes) response to the `CONTINUE?` prompt says:

Continue with the program.
This error condition means that a complete check of the file system is not possible. A second run of **fsck** should be made to recheck this file system.

In Phase 1, an n(no) response to the `CLEAR?` prompt says:

Ignore the error condition.
A *no* response is only appropriate if the user intends to take other measures to fix the problem.

In Phase 1, a y(yes) response to the `CLEAR?` prompt says:

Deallocate i-node *I* by zeroing its contents.
This may invoke the `UNALLOCATED` error condition in Phase 2 for each directory entry pointing to this i-node.

Phase 1 Error Messages

`UNKNOWN FILE TYPE I=`*I* `(CLEAR?)`
The mode word of the i-node *I* suggests that the i-node is not a pipe, special character i-node, regular i-node, or directory i-node.

`LINK COUNT TABLE OVERFLOW (CONTINUE?)`
An internal table for **fsck** containing allocated i-nodes with a link count of zero has no more room.

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 BLOCKS I=*I* (CONTINUE?)
There is more than a tolerable number (usually 10) of blocks 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 associated with i-node *I*.

B DUP I=*I*
I-node *I* contains block number *B*, which 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 DUP BLKS I=*I* (CONTINUE?)
There is more than a tolerable number (usually 10) of blocks claimed by other i-nodes.

DUP TABLE OVERFLOW (CONTINUE?)
An internal table in **fsck** containing duplicate block numbers has no more room.

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 is not printed.

DIRECTORY MISALIGNED I=*I*
The size of a directory i-node is not a multiple of 16. This is only a warning. If the **–q** option is used, this message is not printed.

PARTIALLY ALLOCATED INODE I=*I* (CLEAR?)
I-node *I* is neither allocated nor unallocated.

I-BLOCK CORRUPT - BLOCK=*B*
INODES I - UNRECOVERABLE
This is an informational message that occurs only when the **–r** option is used. The recently redirected block, *B*, listed in the bad block file given by the **–r** option, was found to be in the i-node list. As a result, the information for the INODES contained therein (and, therefore, the files they described) is lost. The list of file/dir names affected appears in Phase 2 as UNALLOCATED.

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. When the duplicate block is found, the following information message is printed:

B `DUP I=`*I*
I-node *I* contains block number *B*, which 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.

Check Path Names

This phase removes directory entries pointing to bad i-nodes found in Phase 1 and Phase 1B. It reports error conditions resulting from:

- root i-node mode and status
- directory i-node pointers out of range
- directory entries pointing to bad i-nodes

Types of Error Messages —Phase 2

Phase 2 has 4 types of error messages:

1. information messages
2. messages with a `FIX?` prompt
3. messages with a `CONTINUE?` prompt
4. messages with a `REMOVE?` prompt

Meaning of Yes/No Responses —Phase 2

In Phase 2, an n(no) response to the `FIX?` prompt says:

Terminate the program since **fsck** will be unable to continue.

In Phase 2, a y(yes) response to the **FIX?** prompt says:

Change the root i-node type to "directory."
If the root i-node data blocks are not directory blocks, a very large number of error conditions are produced.

In Phase 2, an n(no) response to the **CONTINUE?** prompt says:

Terminate the program.

In Phase 2, a y(yes) response to the **CONTINUE?** prompt says:

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.

In Phase 2, an n(no) response to the **REMOVE?** prompt says:

Ignore the error condition.
A *no* response is only appropriate if the user intends to take other measures to fix the problem.

In Phase 2, a y(yes) response to the **REMOVE?** prompt says:

Remove duplicate or unallocated blocks.

Phase 2 Error Messages

ROOT INODE UNALLOCATED. TERMINATING
The root i-node (always i-node number 2) has no allocate mode bits. The occurrence of this error condition indicates a serious problem. The program stops.

ROOT INODE NOT DIRECTORY (FIX?)
The root i-node (usually i-node number 2) is not directory i-node type.

DUPS/BAD IN ROOT INODE (CONTINUE?)
Phase 1 or Phase 1B found duplicate blocks or bad blocks in the root i-node (usually i-node number 2) for the file system.

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.

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 *O*, mode *M*, size *S*, modify time *T*, and filename *F* are printed. If the file system is not mounted and the **-n** option was not specified, the entry is removed automatically if the i-node it points to is character size 0.

DUP/BAD I= *I* **OWNER=** *O* **MODE=** *M*
SIZE= *S* **MTIME=** *T* **DIR=** *F* **(REMOVE?)**
Phase 1 or Phase 1B found duplicate blocks or bad blocks associated with directory entry *F*, directory i-node *I*. The owner *O*, mode *M*, size *S*, modify time *T*, and directory name *F* are printed.

DUP/BAD I= *I* **OWNER=** *O* **MODE=** *M*
SIZE= *S* **MTIME=** *T* **FILE=** *F* **(REMOVE?)**
Phase 1 or Phase 1B found duplicate blocks or bad blocks associated with file entry *F*, i-node *I*. The owner *O*, mode *M*, size *S*, modify time *T*, and filename *F* are printed.

BAD BLK IN DIR I= *I* **OWNER=** *O* **MODE=** *M*
SIZE= *S* **MTIME=** *T*
This message only occurs when the **-D** 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 means 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.

FILE/DIR CORRUPT I=*I* **OWNER=***O* **MODE=***M* **SIZE=***S* **MTIME=***T*
FILE/DIR=[<device>]/*F*
This message is informational and only occurs when the **-r** option is used. One of the blocks in the bad block file given by the **-r** option was found to be a direct or indirect data block of the indicated FILE portion of the file/dir has been zeroed by the redirection process.

Phase 3: Check Connectivity

This phase concerns itself with the directory connectivity seen in Phase 2. It reports error conditions resulting from:

- unreferenced directories
- missing or full **lost+found** directories

Types of Error Messages —Phase 3

Phase 3 has 2 types of error messages:

1. information messages
2. messages with a `RECONNECT?` prompt

Meaning of Yes/No Responses —Phase 3

In Phase 3, an n(no) response to the `RECONNECT?` prompt says:

> Ignore the error condition.
> This invokes the `UNREF` error condition in Phase 4.
> A *no* response is only appropriate if the user intends to take other measures to fix the problem.

In Phase 3, a y(yes) response to the `RECONNECT?` prompt says:

> Reconnect directory i-node *I* to the file system in directory for lost files (usually **lost+found**).
> This may invoke a **lost+found** error condition if there are problems connecting directory i-node *I* to **lost+found**. This invokes `CONNECTED` information message if link was successful.

Phase 3 Error Messages

`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.

`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. Possible problem with 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 (see Procedure 5.2).

`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 checks the link count information seen in Phases 2 and 3. It reports error conditions resulting from:

- unreferenced files
- missing or full **lost+found** directory
- incorrect link counts for files, directories, or special files
- unreferenced files and directories
- bad and duplicate blocks in files and directories
- incorrect total free-i-node counts

Types of Error Messages —Phase 4

Phase 4 has 5 types of error messages:

1. information messages
2. messages with a `RECONNECT?` prompt
3. messages with a `CLEAR?` prompt

4. messages with an **ADJUST?** prompt
5. messages with a **FIX?** prompt

Meaning of Yes/No Responses —Phase 4

In Phase 4, an n(no) response to the **RECONNECT?** prompt says:

> Ignore this error condition.
> This invokes a **CLEAR** error condition later in Phase 4.

In Phase 4, a y(yes) response to the **RECONNECT?** prompt says:

> 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 this phase if there are problems connecting i-node *I* to **lost+found**.

In Phase 4, an n(no) response to the **CLEAR?** prompt says:

> Ignore the error condition.
> A *no* response is only appropriate if the user intends to take other measures to fix the problem.

In Phase 4, a y(yes) response to the **CLEAR?** prompt says:

> Deallocate the i-node by zeroing its contents.

In Phase 4, an n(no) response to the **ADJUST?** prompt says:

> Ignore the error condition.
> A *no* response is only appropriate if the user intends to take other measures to fix the problem.

In Phase 4, a y(yes) response to the **ADJUST?** prompt says:

> Replace link count of file i-node *I* with *Y*.

In Phase 4, an n(no) response to the **FIX?** prompt says:

> Ignore the error condition.
> A *no* response is only appropriate if the user intends to take other measures to fix the problem.

In Phase 4, a y(yes) response to the **FIX?** prompt says:

> Replace count in super-block by actual count.

Phase 4 Error Messages

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 are cleared automatically. Nonempty files are not cleared.

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 later in Phase 4. Possible problem with 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 later in Phase 4. Check size and contents of **lost+found**.

(CLEAR)
The i-node mentioned in the immediately previous **UNREF** error condition cannot be reconnected.

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.

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.

LINK COUNT F I=*I* OWNER=*O* MODE=*M*
SIZE=*S* MTIME=*T* COUNT=*X* SHOULD BE *Y* (ADJUST?)
The link count for *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.

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 are cleared automatically. Nonempty directories are not cleared.

UNREF DIR *I* **OWNER=** *O* **MODE=** *M*
SIZE= *S* **MTIME=** *T* **COUNT=** *X* **SHOULD BE** *Y* **(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 directories are cleared automatically. Nonempty directories are not cleared.

BAD/DUP FILE I= *I* **OWNER=** *O* **MODE=** *M*
SIZE= *S* **MTIME=** *T* **COUNT=** *X* **SHOULD BE** *Y* **(CLEAR?)**
Phase 1 or Phase 1B found duplicate blocks or bad blocks associated with file i-node *I*. The owner *O*, mode *M*, size *S*, and modify time *T* of i-node *I* are printed.

BAD/DUP DIR I= *I* **OWNER=** *O* **MODE=** *M*
SIZE= *S* **MTIME=** *T* **COUNT=** *X* **SHOULD BE** *Y* **(CLEAR?)**
Phase 1 or Phase 1B 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.

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.

Phase 5: Check Free List

This phase checks the free-block list. It reports 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
- total free-block count incorrect

Types of Error Messages —Phase 5

Phase 5 has 4 types of error messages:

1. information messages
2. messages that have a `CONTINUE?` prompt
3. messages that have a `FIX?` prompt
4. messages that have a `SALVAGE?` prompt

Meaning of Yes/No Responses —Phase 5

In Phase 5, an n(no) response to the `CONTINUE?` prompt says:

> Terminate the program.

In Phase 5, a y(yes) response to the `CONTINUE?` prompt says:

> Ignore rest of the free-block list and continue execution of **fsck**.
> This error condition will always invoke `BAD BLKS IN FREE LIST` error condition later in Phase 5.

In Phase 5, an n(no) response to the `FIX?` prompt says:

> Ignore the error condition.
> A *no* response is only appropriate if the user intends to take other measures to fix the problem.

In Phase 5, a y(yes) response to the `FIX?` prompt says:

> Replace count in super-block by actual count.

In Phase 5, an n(no) response to the `SALVAGE?` prompt says:

> Ignore the error condition.
> A *no* response is only appropriate if the user intends to take other measures to fix the problem.

In Phase 5, a y(yes) response to the **SALVAGE?** prompt says:

> Replace actual free-block list with a new free-block list.
> The new free-block list will be ordered according to the gap and cylinder specs of the **-s** or **-S** option to reduce time spent waiting for the disk to rotate into position.

Phase 5 Error Messages

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.

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.

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 later in Phase 5.

X **BAD BLKS IN FREE LIST**
X blocks in the free-block list have a block number lower 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 later 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 later 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 later 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.

BAD FREE LIST (SALVAGE?)
This message is always preceded by one or more of the Phase 5 information messages. If the **–q** option is specified, the free-block list will be salvaged automatically.

Phase 6: Salvage Free List

This phase reconstructs the free-block list. It has one possible error condition that results from bad blocks-per-cylinder and gap values.

Phase 6 Error Messages

DEFAULT FREE-BLOCK LIST SPACING ASSUMED
This is an advisory message indicating the blocks-to-skip (gap) is greater than the blocks-per-cylinder, the blocks-to-skip is less than 1, the blocks-per-cylinder is less than 1, or the blocks-per-cylinder is greater than 500. The values of 7 blocks-to-skip and 400 blocks-per-cylinder are used.

Because the values used are incorrect for the computer, care must be taken to specify correct values with the **–s** option on the command line. See the **fsck**(1M) and **mkfs**(1M) manual pages for further details.

Cleanup Phase

Once a file system has been checked, a few cleanup functions are performed. The cleanup phase displays advisory messages about the file system and status of the file system.

Cleanup Phase Messages

X **files** *Y* **blocks** *Z* **free**
This is an advisory message indicating that the file system checked contained *X* files using *Y* blocks leaving *Z* blocks free in the file system.

******* BOOT SYSTEM V/88 (NO SYNC!) *******
This is an advisory message indicating that a mounted file system or the root file system has been modified by **fsck**. If the operating system is not rebooted immediately without **sync**, the work done by **fsck** may be undone by the in-core copies of tables the operating system keeps. If the **–b** option of the **fsck** command was specified and the file system is **root**, a reboot is automatically done.

******* FILE SYSTEM WAS MODIFIED *******

This is an advisory message indicating that the current file system was modified by **fsck**.

6 Performance Management

Introduction

This chapter describes ways to monitor and enhance the performance of your computer system.

- General approach to performance management

  Finding and fixing performance problems

- Improving performance

  Tuning the kernel for minimum overhead and tuning the disk subsystem for maximum throughput

  Workload analysis and housekeeping techniques for reducing peak load

  Estimating capacity

  Samples of typical procedures

- Performance tools

  Description of the performance tools.

- Tunable parameters

  Extensive definition of the tunable parameters.

General Approach to Performance Management

Performance management is an activity that may need your attention when you first set up your computer. When you bring the computer up for the first time, the system is automatically set to a basic configuration that is satisfactory for most applications. This configuration, however, cannot take into account the usage patterns and the behavior of your particular applications. For this reason, the structure of the system allows you to reconfigure it to enhance the performance for your particular application over that of the standard configuration.

It is possible that you may never have to do any special fine tuning of your system, and that your only experience with reconfiguration will be adding new memory and peripherals.

Finding Problems

The system is automatically configured to its default configuration the first time it is booted. After the system has been running several days, you may receive signals from a variety of sources that the system needs tuning. In particular, you may see that the response time at the console is frequently slow. Your job of performance management really begins then. To proceed, you will use the tools described later in this chapter (e.g., the **sar** command) to pinpoint the problem.

Fixing Problems

At this point, you need to take some corrective action. Some of the major areas for action are:

- Modifying the tunable configuration parameters.

  This is usually referred to as tuning the kernel, since you are adjusting the essential control structures at the heart of the system (the kernel). Many of these parameters are described in detail later in this chapter.

  When you change any of these parameters you must reconfigure the system, which is the way to recreate a new bootable version of the operating system that incorporates the new parameter definitions.

- Removing optional kernel packages not needed by your applications.

  This procedure makes disk and memory space available for user programs and can benefit performance.

- Improving disk utilization.

  In addition to the allocation of system memory space defined by the tunable parameters, you have some control over how your file systems are organized on the disk and policies to cache frequently-used programs in memory.

- Defining best system usage patterns.

  Finally, you can establish the model for best system usage, e.g., encouraging users to run large non-interactive programs at night.

These areas are discussed in more detail in the remaining sections of the chapter.

Improving Performance

Modifying the Tunable Configuration Parameters

The core system tunable parameters are set with **sysgen** which modifies the parameter entries in the **sysgen** kernel file, or other files in the **sysgen** data base, e.g., **shm**, **msg**, or **sem**. For full definitions of the individual tunable parameters, and suggestions about setting them, see the section *Tunable Parameters* near the end of this chapter (see the tables in that section for the recommended initial values for the tunable parameters relative to a particular memory size). Use the **/etc/sysdef** command to see what the current values of the tunable parameters are in the present configuration of your system.

Generally, the default parameters for your configuration result in acceptable performance. If, however, you are running an application that has special performance needs, you can use the tools described in the *Performance Tools* section to measure system load and determine which parameters might be changed to improve performance. Two key parameters, NBUF (system buffers) and NHBUF (hash buffers), have a strong impact on the efficiency of disk utilization. They are discussed in the *Improving Disk Utilization* section.

See Procedure 6.1, *Reconfigure the System*, for examples of editing the tunables and reconfiguring the system.

Improving Disk Utilization

Disk input/output may cause a bottleneck in system performance. There are four steps in tuning the disk subsystem for better utilization:

- Allocate a larger buffer cache.
- Setting text-bit (sticky-bit) for frequently used executables.
- Organizing the file systems to minimize disk activity.
- Set the logical block size to suit the application.

Size of the Buffer Cache

Use the number of buffers (NBUF and NHBUF) given in the "Kernel Parameter Values" table under the section *Tunable Parameters.* These values come close to optimum for most system workloads.

The NBUF parameter controls the number of 1K buffers in the system buffer cache. These buffers are used for file systems whose logical block size is 512 bytes or 1K bytes. These buffers hold recently-used data on the chance that it will be needed again. If a read or 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 needed 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. If the value for NBUF is left null, the system calculates the values shown in the "Kernel Parameter Values" table in the section *Tunable Parameters.* The value for NHBUF is a power of 2 that is about one-quarter the value of NBUF; it must be specified using the **sysgen** utility.

Increasing NBUF and NHBUF, up to a point, improves system performance. A system with 2Mb of memory can typically devote about 250Kb of memory to buffers, while a system with 4Mb of memory can devote 600Kb 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.

After the operating system has run for several days, check for excessive swapping activity. If such activity is found, reduce the number of buffers (NBUF and NHBUF) or increase your system memory.

Setting Text-Bit (Sticky-Bit)

Setting the text-bit can reduce the disk traffic of a select group of commands. Text pages of sticky commands are kept resident in memory, even when the process terminates. Once loaded into memory, such pages usually remain. One exception is when the pages are reclaimed by the paging daemon in a tight memory situation (i.e., when the number of available pages falls below

the low-water mark as defined by the tunable parameter GPGSLO). Finding sticky pages already in memory can reduce considerably the loading time for the text pages of a process.

NOTE

Sticky text is not kept in the **swap** area, as was the case in earlier, non-paging computers.

On systems that usually run a light to medium workload and are seldom in a tight-memory situation, setting the text-bit can cause a significant improvement in performance. On systems with limited memory or a heavy workload, however, the text-bit should not be used. Follow these guidelines:

System Memory	Text-Bit Guideline
2M	Do not use the text-bit
4M to 16M	Use the text-bit if the average amount of free memory is greater than the high-water mark (GPGSHI)

Determine average amount of free memory by setting the text-bit on some test commands with the **chmod**(1) command, and using the **–r** option of **sar**(1) to determine the average FREEMEM count over a typical interval of system activity. If the **sar** report shows that it is safe to set the text-bit on for some commands, logical candidates would be frequently used, (preferably) small commands. If the average free memory for the system is low (i.e., close to the high water mark), then performance is not likely to be significantly improved and may be hurt by setting the text-bit.

File System Organization

This section describes several actions you can take to reduce the overhead of file access. As file systems are used, the blocks of individual member files tend to become physically scattered around the disk(s) and I/O becomes less efficient. This scattering yields poor ordering of blocks with files and poor directory structure.

Organization of File System Free List

The computer file systems are set up to allocate free blocks in a manner that allows the files to be read or written with efficiency. A free list array is created when a file system is created with **/etc/mkfs**(1M). The free list is set up with the rotational gap specified by **mkfs** options. (See the *Using mkfs* section of Chapter 5 for recommended values for the rotation gap.) The difference between successive block numbers in the free list is the rotational gap. For example, a file created on a system with a rotational gap of 10 may consist of blocks 510, 511, and 512. When the file is read, I/O requests are sent to the disk drive to read blocks 510, 520, and 530. As soon as the drive finishes reading block 510 and has started to process the second request, block 520 will be moving over the read/write head just as the drive is ready to read that request. This method makes for efficient I/O operation.

However, as you start changing files (changing size or removing), the efficiency starts to decrease. When several files are being created at once, they will be contending for blocks from the free list. Some of the blocks allocated to the files will be out of sequence. As you can see, the free list also becomes scattered about the disk as blocks are allocated and freed.

Directory Organization

Directory organization also affects input/output performance. The problems show up when files are removed by users. When a file is removed from a directory, the i-node number is nulled out. This leaves an unused slot for that i-node; over time the number of empty slots may become quite large. If you have a directory with 100 files in it and you remove the first 99 files, the directory still contains the 99 empty slots, at 16 bytes per slot, preceding the active slot. In effect, unless a directory is reorganized on the disk, it will retain the largest size it has ever achieved.

Restoring Good File System Organization

There is no automatic way to solve these problems; however, you can manually rearrange the file system. Here are a variety of ways to do this. Note that in all cases the file system(s) must be unmounted.

Step 1: To reorganize the free list, run **fsck**(1M) **–s**.

Step 2: To reorganize particular directory structures, use **cpio**(1) **–pdm** to copy them to a temporary location; remove the original structure; then use **cpio –pdm** to copy them back to their original location.

Step 3: To reorganize the file system, run **dcopy**(1M) **–s**. This is probably the most comprehensive way to handle file system reorganization. **dcopy** performs Steps 1 and 2 above, and it also provides the opportunity to change the file system and i-node list sizes.

Step 4: To prevent file system indirection (which causes inefficient directory searching), use the following command to find directories with more than 10 logical blocks:

find / –type d –size +10 –print

A directory entry takes 16 bytes, so a directory in a 1K file system requires indirection if it has more than 640 entries ((1024/16) * 10).

Step 5: If you have more than one disk, balance heavily used file systems across the disks.

Defining Best System Usage Patterns

After the kernel and the system activities are tuned, and the file systems organized, the next step toward improving system performance is to perform some housekeeping activities and to check whether prime time load can be reduced. The person responsible for administering the system should check for:

- less important jobs interfering with more important jobs
- unnecessary activities being carried out
- scheduling of selected jobs at times when the system is not so busy
- the efficiency of user-defined features, e.g., **.profile** and $PATH

ps

The **ps**(1) command is used to obtain information about active processes. The command gives a "snapshot" picture of what is going on, which is useful when you are trying to identify what processes are loading the system. Things will probably change by the time the output appears; however, the entries that you should be interested in are: TIME (minutes and seconds of CPU time used by processes) and STIME (time when process first started).

When you spot a "runaway" process (one that uses progressively more system resources over a period of time while you are monitoring it), you should check with the owner. It is possible that such a process should be stopped immediately via the **kill**(1) **–9** command. When you have a real runaway, it continues to use up system resources until everything grinds to a halt.

When you spot processes that take a very long time to execute, you should consider using **cron**(1M) to execute the job during off-hours.

User $PATH Variables

$PATH is searched upon each command execution. Before outputting `not found`, the system must search every directory in $PATH. These searches require both processor and disk time. If there is a disk or processor bottleneck, changes here can help performance.

Some things that you should check for in user $PATH variables are:

- Path Efficiency

  $PATH is read left to right, so the most likely places to find the command should be first in the path (**/bin** and **/usr/bin**). Make sure that a directory is not searched more than once for a command.

- Convenience and Human Factors

  Users may prefer to have the current directory listed first in the path. (Start with the :. notation).

- Path Length

  In general, $PATH should have the least number of required entries.

- Large Directory Searches

  Searches of large directories should be avoided if possible. Put any large directories at the end of $PATH.

Samples of General Procedures

This section describes a general procedure for troubleshooting performance problems.

Sample Procedure for Investigating Performance Problems

Locating the source of a performance problem can require some careful detective work. Hence, what follows is not a canned procedure, but a sample of a typical approach. It covers basic areas where problems occur and suggests actions to take to solve the problems. The most common symptom that a problem exists is consistently poor response time. Refer to Figure 6-1 for an outline of the approach. Note that if you have identified a familiar problem area, see Procedure 6.1, *Reconfigure the System*, to make the necessary system parameter changes.

Check for Excess Swapping

The first thing to look at is swapping activity, since the swapping of pages is costly in both disk and CPU overhead. Get the **sar**(1) **–qw** report. Look at the %swpocc column (percentage that the swap queue is occupied) for values greater than 5. Then look at the swap-out rate (swpot/s) for values greater than 1.00.

Check whether the "freemem" count (number of pages available to user programs), shown by **sar –r**, is consistently less than the value of the tunable parameter GPGSHI (high-water mark).

If any or all of these conditions are occurring frequently, increase your system memory or check if you can reduce memory allocated for system buffers. If a large number of buffers have been configured and the buffer cache hit ratios (**sar –b**) are 90% or more, try decreasing the number of buffers (stored in the tunable parameter NBUF). It is possible that returning memory space occupied by some buffers will solve the swapping problem by leaving more space for user programs. Additional memory for user programs can also be obtained by uninstalling optional kernel utilities that are not needed by your applications.

Check for Disk Bottleneck

If the value of %wio (from the **sar –u** report above) is greater than 10%, or if the %busy for a disk drive (obtained by **sar –d**) is greater than 50%, then the system has a disk bottleneck. Some ways to alleviate a disk bottleneck are:

1. Increase the number of buffers.

2. Organize the file system to minimize disk activity. If you have two disks, distribute the file systems for a more balanced load.
3. Consider adding more memory if the situation persists. Additional memory reduces swapping/paging traffic and allows an expanded buffer pool (reducing the number of user-level reads and writes that need to go out to disk).
4. Consider setting the text-bit on for frequently used files. See the earlier discussion of this subject under *Setting Text-Bit (Sticky-Bit).*
5. Consider adding an additional disk and balancing the most active file systems across the two disks.

Check for Modem Interrupts

Run **sar -y** to get a report of activity on terminal devices. If the number of modem interrupts per second, **mdmin/s**, is much greater than 0, your system may have faulty communications hardware. Repair it.

Check for Potential Table Overflows

To check for potential process, file, or i-node table overflows, run the **sar –v** report. Overflows in these tables are avoided by increasing NPROC, NFILE, and NINODE/NS5INODE (in the **sysgen** kernel file).

Shift Workload to Off-Peak Hours

Examine **/usr/spool/cron/crontabs/*** to see if jobs are queued up for peak periods that might better be run at times when the system is idle. Use the **ps** command to determine what processes are heavily loading the system. Encourage users to run large, non-interactive commands at off-peak hours. You may also want to run such commands with a low priority by using the **nice**(1) or **batch**(1) commands.

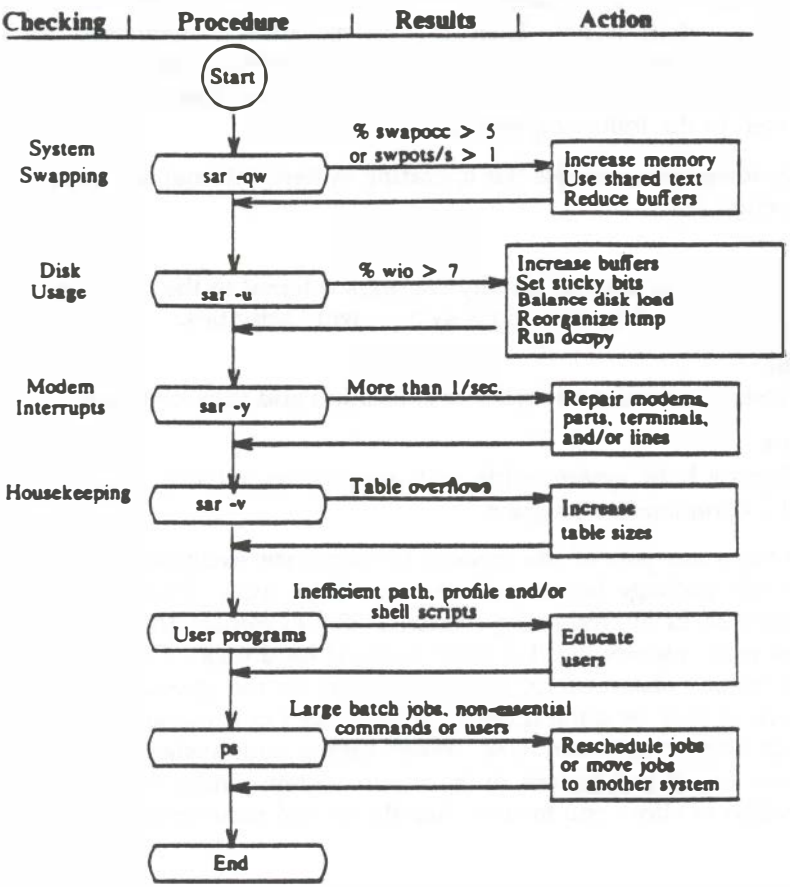


Figure 6-1. Outline of Typical Troubleshooting Procedure

Performance Tools

Internal activity is measured by a number of counters contained in the operating system kernel. Each time an operation is performed, an associated counter is incremented. **sar** and the other performance tools allow you to monitor the values of these counters. The functions monitored by **sar** are discussed in the following sections.

The performance tools for the operating system internal activities described in this section are:

sar
Samples cumulative activity counters internal to the operating system and provides reports on various system-wide activities.

sadp
Produces profiles of disk access location and seek distance.

timex
Reports both system-wide and per-process activity during the execution of a command or program.

These tools are part of the System Performance Analysis Utilities. You must install this package before you can use these tools. Examples of these tools are provided in the following sections. The examples are from a system with 2Mb of main memory and a 30Mb integral hard disk. Command outputs are typical values observed for user workloads on the operating system. Values you receive may be quite different from values in the examples, depending on your application or benchmark. When tuning your system, it is recommended that you use a benchmark or have the system under normal load for your application to allow you to tune directly toward your specific application.

sar

Throughout this section, **sar** options are described with an analysis of sample outputs of the options. **sar** can be used either to gather system activity data or to extract what has been collected in data files created by **sa1** and **sa2**. **sa1** and **sa2** are initiated by entries put in the **crontab sys** file.

sar –a

The **sar –a** option reports the use of file access operations. The operating system routines reported are:

iget/s
Number of files located by i-node entry per second.

namei/s
Number of file system path searches per second. **namei** calls **iget**, so **iget/s** is always larger than **namei/s**.

dirbk/s
Number of directory block reads issued per second.

An example of **sar –a** output, with a 30-second sampling interval, follows:

```
unix    unix     3.2  1.0  m88000 12/30/89
12:41:40  iget/s namei/s dirbk/s
12:42:10       4       1       3
12:42:40       2       1       1
12:43:10       5       2       3
Average        4       1       3
```

The larger the values reported, the more time the kernel is spending to access user files. This indicates how heavily programs and applications are using the file system(s). The **–a** option is helpful for understanding how disk-dependent the application system is; it is not used for any specific tuning step.

sar –b

The **–b** option reports the following buffer activity.

bread/s
Average number of physical blocks read into the system buffers from the disk (or other block devices) per second.

lread/s
Average number of logical blocks read from system buffers per second.

%rcache
Fraction of logical reads found in buffer cache (100% minus the ratio of **breads** to **lreads**).

bwrit/s
Average number of physical blocks written from the system buffers to disk (or other block devices) per second.

lwrit/s
Average number of logical blocks written to system buffers per second.

%wcache
Fraction of logical writes found in buffer cache (100% minus the ratio of **bwrit/s** to **lwrit/s**).

pread/s
Average number of physical read requests per second.

pwrit/s
Average number of physical write requests per second.

The entries that you should be most interested in are the cache hit ratios **%rcache** and **%wcache** which measure the effectiveness of system buffering. If **%rcache** falls below 90, or **%wcache** falls below 65, it may be possible to improve performance by increasing the number of buffers.

An example of **sar –b** output follows:

```
unix  unix  3.2  1.0  m88000 12/30/89
16:32:57 bread/s lread/s %rcache bwrit/s lwrit/s %wcache pread/s pwrit/s
16:33:07       3      39      93       1      16      91       0       0
16:33:17       4      40      90       2      16      87       0       0
16:33:27       4      41      90       3       7      64       0       0
Average        4      40      91       2      13      84       0       0
```

This example shows that the buffers are not causing bottlenecks because all data is within acceptable limits.

sar –c

The **–c** option reports system calls in the following categories:

scall/s
All types of system calls per second, generally about 30 per second on a busy 4 to 6 user system.

sread/s
Read system calls per second.

swrit/s
Write system calls per second.

fork/s
Fork system calls per second, about 0.5 per second on a 4 to 6 user system. This number increases if shell scripts are running.

exec/s
Exec system calls per second. (If (**exec/s**) / (**fork/s**) is greater than 3, look for inefficient $PATHs.)

rchar/s
Characters (bytes) transferred by read system calls per second.

wchar/s
Characters (bytes) transferred by write system calls per second.

Typically, reads plus writes account for about half of the total system calls, although this varies greatly with the activities that are being performed by the system.

An example of **sar –c** output follows:

```
unix  unix  3.2  1.0  m88000 12/30/89
18:33:04 scall/s sread/s swrit/s fork/s exec/s rchar/s wchar/s
18:33:35      38      16       6   0.03   0.03    6089    1638
18:34:05      39      16       4   0.07   0.07    6123    1602
18:34:35      38      17       5   0.17   0.17    6042    1704
Average       38      16       5   0.09   0.09    6085    1648
```

A parallel option is available for systems that have Remote File Sharing (RFS) installed. See the description of **sar –Dc** in Chapter 10, *Remote File Sharing*.

sar –d

The **sar –d** option reports the activity of block devices.

device
Name of the block device(s) that **sar** is monitoring.

%busy
Percent of time the device was servicing a transfer request.

avque
The average number of requests outstanding during the period of time (measured only when the queue is occupied).

r+w/s
Number of read and write transfers to the device per second.

blks/s
Number of 512-byte blocks transferred to the device per second.

avwait
Average time in milliseconds that transfer requests wait idly in the queue (measured only when the queue is occupied).

avserv
Average time in milliseconds for a transfer request to be completed by the device (for disks this includes seek, rotational latency, and data transfer times).

Note that queue lengths and wait times are measured while the queue had something on it. If **busy** is small, large queues and service times probably represent the periodic **sync** efforts by the system to ensure that altered blocks are written to the disk in a timely fashion.

sar –m

The **sar –m** option reports on interprocess communication activities. Message and semaphore calls are reported as follows:

msg/s
Number of message operations (sends and receives) per second.

sema/s
Number of semaphore operations per second.

An example of **sar –m** output follows:

```
unix  unix  3.2  1.0  m88000 12/30/89
15:16:58   msg/s  sema/s
15:17:32     0.00    0.00
15:18:02     0.00    0.00
15:18:32     0.00    0.00
Average      0.00    0.00
```

These figures will usually be zero (0.00) unless you are running applications that use the message or semaphore features.

sar –q

The **sar –q** option reports the average queue length while the queue is occupied and percent of time occupied.

runq-sz
Run queue of processes in memory; typically, this should be less than 2. Consistently higher values mean you are CPU-bound.

%runocc
The percentage of time the run queue is occupied; the larger this value is the better.

swpq-sz
Swap queue of processes to be swapped out; the smaller this number is the better.

%swpocc
The percentage of time the swap queue is occupied; the smaller this value is the better.

An example of **sar –q** follows:

```
unix  unix  3.2  1.0  m88000 12/30/89
11:00:56 runq-sz %runocc swpq-sz %swpocc
11:01:07     1.7      98     1.5      36
11:01:17     1.0      63     1.0      31
11:01:27     1.0      58     1.0      49
Average      1.3      74     1.2      39
```

In this example, the processor utilization (%runocc) varies between 58% and 98%, while the fraction of time the swap queue is not empty (%swpocc) is 31% to 49%. This means that memory is not causing a major bottleneck in the system throughput, but more memory would help reduce the swapping/paging activity.

If %runocc is greater than 90 and **runq-sz** is greater than 2, the CPU is heavily loaded and response is degraded. Here, additional CPU capacity may be required to obtain acceptable system response. If %swpocc is greater than 20, more memory or fewer buffers would help reduce swapping/paging activity.

sar –u

The CPU utilization is listed by **sar –u** (default). At any given moment the processor will be either busy or idle. When busy, the processor will be in either user or system mode. When idle, the processor is either waiting for input/output completion or has no work to do. The **–u** option of **sar** lists the percent of time that the processor is in system mode (%sys), user mode (%user), waiting for input/output completion (%wio), and idle time (%idle).

In typical timesharing use, %sys and %usr are about the same value. In special applications, either of these may be larger than the other without anything being abnormal. A high %wio generally means a disk bottleneck. A high %idle, with degraded response time, may mean memory constraints; time spent waiting for memory is attributed to %idle.

An example of **sar –u** follows:

```
unix  unix  3.2  1.0  m88000 12/30/89
09:20:08  %usr      %sys     %wio     %idle
09:40:12     6         7        2        86
10:00:03     7         9        3        80
10:20:07    14        16       10        61
Average      9        11        5        76
```

A parallel option is available for systems that have RFS installed. See the description of **sar –Du** in Chapter 10, *Remote File Sharing*.

If your computer is equipped with a co-processor, **sar -u** produces additional output for the co-processor, including an extra column showing the number of system calls per second being executed on the co-processor. The %wio column is empty for the co-processor because the co-processor does not do I/O. The following **sar** data were collected on a co-processor system at intervals of 20 seconds.

```
unix  unix  3.2  1.0  m88000 12/30/89
10:02:07     %usr     %sys     %wio     %idle
10:02:27       82       18        0         0
10:02:47       39       35       16        10
10:03:07        7       28       16        50
10:03:27        1       16        0        83

Average        32       24        8        36

10:02:07  %co-usr  %co-sys          %co-idle scall/s
10:02:27       98        0                 2       0
10:02:47       65        0                35       0
10:03:07       11        0                89       1
10:03:27        0        0               100       0

Average        44        0                56       0
```

sar –v

The **–v** option reports the status of process, i-node, file, shared memory record, and shared memory file tables. From this report you know when the system tables need to be modified.

proc-sz
Number of process table entries presently being used/allocated in the kernel.

inod-sz
Number of i-node table entries presently being used/allocated in the kernel.

file-sz
Number of file table entries presently being used/allocated in the kernel.

ov
Number of times an overflow occurred. (One column for each of the above three items.)

lock-sz
The number of shared memory record table entries presently being used/allocated in the kernel.

fhdr-sz
No longer applicable.

The values are given as *level/table size*. An example of **sar –v** follows:

```
unix  unix  3.2  1.0  m88000 12/30/89
17:36:05 proc-sz ov   inod-sz ov   file-sz ov   lock-sz   fhdr-sz
17:36:35  17/ 40  0    39/ 80  0    29/ 80  0     0/ 50     0/  0
17:37:05  19/ 40  0    46/ 80  0    35/ 80  0     0/ 50     0/  0
17:37:35  18/ 40  0    43/ 80  0    34/ 80  0     0/ 50     0/  0
```

This example shows that all tables are large enough to have no overflows. Sizes could be reduced to save main memory space if these are the highest values ever recorded.

sar –w

The **–w** option reports swapping and switching activity. The following are some target values and observations.

swpin/s
Number of transfers into memory per second.

bswin/s
Number of 512-byte-block units (blocks) transferred for swap-ins (including initial loading of some programs) per second.

swpot/s
Number of transfers from memory to the disk swap area per second. If greater than 1, memory may need to be increased or buffers decreased.

bswot/s
Number of blocks transferred for swap-outs per second.

pswch/s
Process switches per second. This should be 30 to 50 on a busy 4 to 6 user system.

An example of **sar –w** output follows:

```
unix  unix  3.2  1.0  m88000 12/30/89
19:53:44 swpin/s bswin/s swpot/s bswot/s pswch/s
19:53:58     0.0      0.0     0.0      0.0      37
19:54:14     0.0      0.0     0.0      0.0      39
19:54:24     0.0      0.0     0.0      0.0      39
Average      0.0      0.0     0.0      0.0      38
```

This example shows that there is sufficient memory for the currently active users, since no swapping is occurring.

sar –p

The **–p** option reports paging activity. The following page rates are recorded.

vflt/s
Number of address translation page faults per second (valid page not present in memory).

pflt/s
Number of page faults from protection errors per second (illegal access to page) or "copy-on-writes". **pflt/s** generally consists entirely of "copy-on-writes."

pgfil/s
Number of **vflt/s** per second satisfied by a page-in from the file system. (Each **pgfil** causes two **lreads**; see **sar –b**).

rclm/s
Number of valid pages per second that the system has reclaimed (added to list of free pages).

An example of **sar –p** output follows:

```
unix  unix  3.2  1.0  m88000 12/30/89
12:01:51 vflt/s pflt/s pgfil/s rclm/s
12:56:52  13.91   2.80   5.63   11.21
```

sar –r

The **–r** option records the number of memory pages and swap file disk blocks that are currently unused. The following are recorded.

freemem
Average number of 1K pages of memory available to user processes over the intervals sampled by the command.

freeswap
Number of 512-byte disk blocks available for process swapping.

An example of **sar –r** output follows:

```
unix  unix  3.2  1.0  m88000 12/30/89
12:01:51 freemem freeswap
12:56:52     208      5848
```

sar –y

The **–y** option monitors terminal device activities. If you have a lot of terminal I/O, you can use this report to determine if there are any bad lines. Activities recorded are defined as follows:

rawch/s
Input characters (raw queue) per second.

canch/s
Input characters processed by canon (canonical queue) per second.

outch/s
Output characters (output queue) per second.

rcvin/s
Receiver hardware interrupts per second.

xmtin/s
Transmitter hardware interrupts per second.

mdmin/s
Modem interrupts per second.

The number of modem interrupts per second (**mdmin/s**) should be close to 0, and the receive and transmit interrupts per second (**xmtin/s** and **rcvin/s**) should be less than or equal to the number of incoming or outgoing characters, respectively. If this is not the case, check for bad lines.

An example of **sar –y** output follows:

```
unix  unix  3.2  1.0  m88000 12/30/89
16:50:11 rawch/s canch/s outch/s rcvin/s xmtin/s mdmin/s
16:50:41     112      15     653     103     102       0
16:51:11     107       7     654     104     105       0
16:51:41      99       5     641      99     105       0
Average      106       9     649     102     104       0
```

sar –A

The **sar –A** option is equivalent to **sar –uUbdycwaqvmMprPKDS.** The report includes RFS operations (the **–D** and **–S** options) described in Chapter 10, *Remote File Sharing*. The **–A** option provides a view of overall system performance. Use it to get a more global perspective. If data from more that one time slice is shown, the report includes averages.

An example of **sar –A** follows:

```
unix  unix  3.2  1.0  m88000 12/30/89

00:00:00     %usr     %sys     %sys     %wio   %idle
                     local  remote
01:00:00        0         0        0        0     100

00:00:00  device    %busy    avque    r+w/s  blks/s  avwait  avserv
01:00:00 hdsk-0          0      1.5        0       0    10.9    21.1
         hdsk-1          0      2.1        0       0    30.6    27.9

00:00:00 runq-sz %runocc swpq-sz %swpocc
01:00:00      1.1        0

00:00:00 bread/s lread/s %rcache bwrit/s lwrit/s %wcache pread/s pwrit/s
01:00:00        0        0      97        0       0      67        0       0

00:00:00 swpin/s bswin/s swpot/s bswot/s pswch/s
  .
  .
  .
0:00:00  vflt/s  pflt/s pgfil/s  rclm/s
01:00:00    0.09     0.08    0.01     0.00

00:00:00 freemem freeswp
01:00:00      362    10196

00:00:00 serv/lo-hi   request   request    server    server
               3 - 6      %busy  avg lgth    %avail avg avail
01:00:00           0        0.0         0       0.0         0
```

timex

The **timex**(1) command times a command and reports the system activities that occurred during the time the command was executing. If no other programs are running, then **timex** can give you a good idea of which resources a specific command uses during its execution. System consumption can be collected for each application program and used for tuning the heavily loaded resources. For this example, the **date** command was used:

```
$ timex -s date
Wed Feb 19 08:32:50 EST 1986
real         0.17
user         0.01
sys          0.09
sysV88  sysV88  3.2  1.0  m88000 12/30/89
08:32:05     %usr     %sys     %sys     %wio   %idle
                     local  remote
08:32:05       10       77        0       13       0
08:32:05 bread/s lread/s %rcache bwrit/s lwrit/s %wcache pread/s pwrit/s
08:32:05        8      125       94        0       12      100        0       0
08:32:05  device   %busy   avque   r+w/s  blks/s  avwait  avserv
08:32:05 hdsk-0          12     1.0       4       8     0.0    30.0
         hdsk-1           8     1.0       4       8     0.0    20.0
08:32:05 rawch/s canch/s outch/s rcvin/s xmtin/s mdmin/s
08:32:05        0       0      56       0       4       0
08:32:05 scall/s sread/s swrit/s  fork/s  exec/s rchar/s wchar/s
08:32:05
   in            0       0       0             0.00       0       0
   out           0       0       0             0.00       0       0
   local       244      29       8     5.77    7.69   44154     879
```

```
08:32:05 swpin/s bswin/s swpot/s bswot/s pswch/s
08:32:05    0.00      0.0     0.00      0.0       33
08:32:05  iget/s namei/s dirbk/s
08:32:05      40       21       42
08:32:05 runq-sz %runocc swpq-sz %swpocc
08:32:05
08:32:05 proc-sz ov inod-sz ov file-sz ov lock-sz     fhdr-sz
08:32:05  13/ 60  0  33/150  0  20/150  0   0/100       0/  0
08:32:05   msg/s  sema/s
08:32:05    0.00    0.00
08:32:05  vflt/s  pflt/s pgfil/s  rclm/s
08:32:05   44.23   46.15    3.85    0.00
08:32:05 freemem freeswp
08:32:05     306   10196
08:32:05 serv/lo-hi   request   request    server    server
               3 - 6     %busy  avg lgth    %avail avg avail
08:32:05          0       0.0          0       0.0          0
```

While **date**, for its simplicity, was used for the preceding demonstration, it is not the best example since it is not a major user of system resources.

timex can be used in the following way:

$ **timex –s** *application program*

Your application program will operate normally. When you finish and exit, the **timex** result is printed on your screen, giving you a clear picture of system resources used by your program.

sadp

The **sadp**(1M) command reports disk access locations and seek distance in in tabular (**–t**) or histogram (**–h**) form:

sadp [**–th**] [**–d** *device*[*–drive*]] **s** [*n*]

Disk activity is sampled once every second during a specified interval *n*. An example of **sadp** output for hard disk drive 0 follows:

$ **sadp –h –d** *cntrlr*-0 3600<CR>

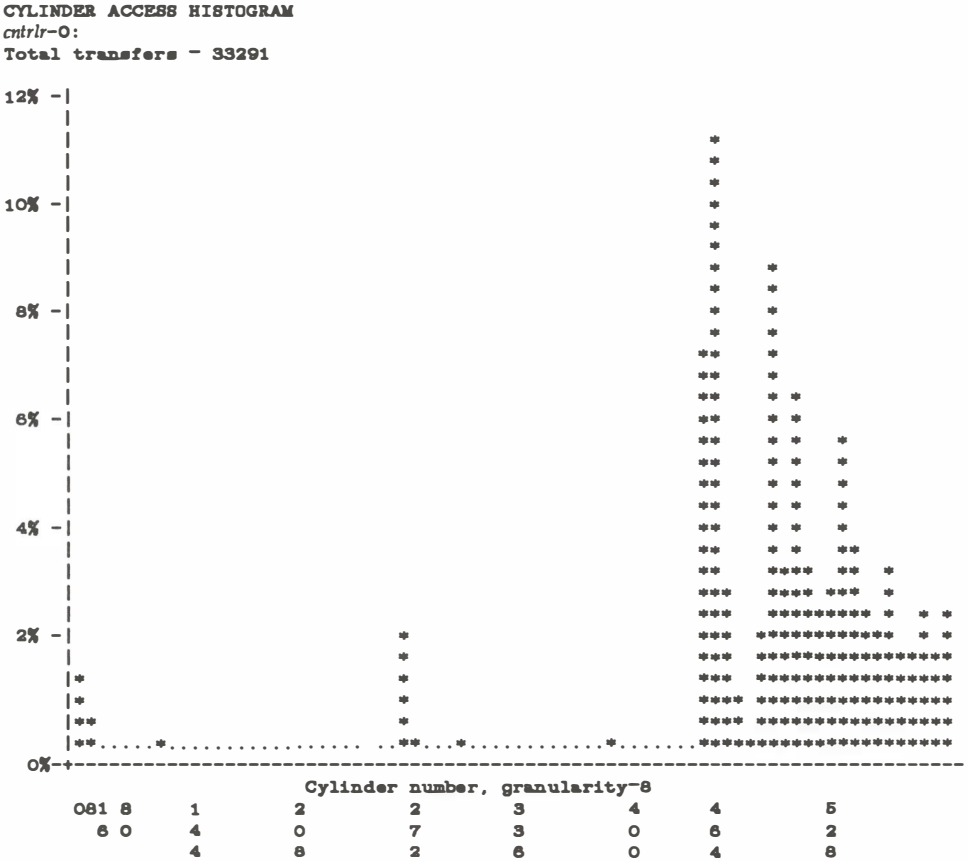


Figure 6-2. Output from **sadp**: Cylinder Access Histogram

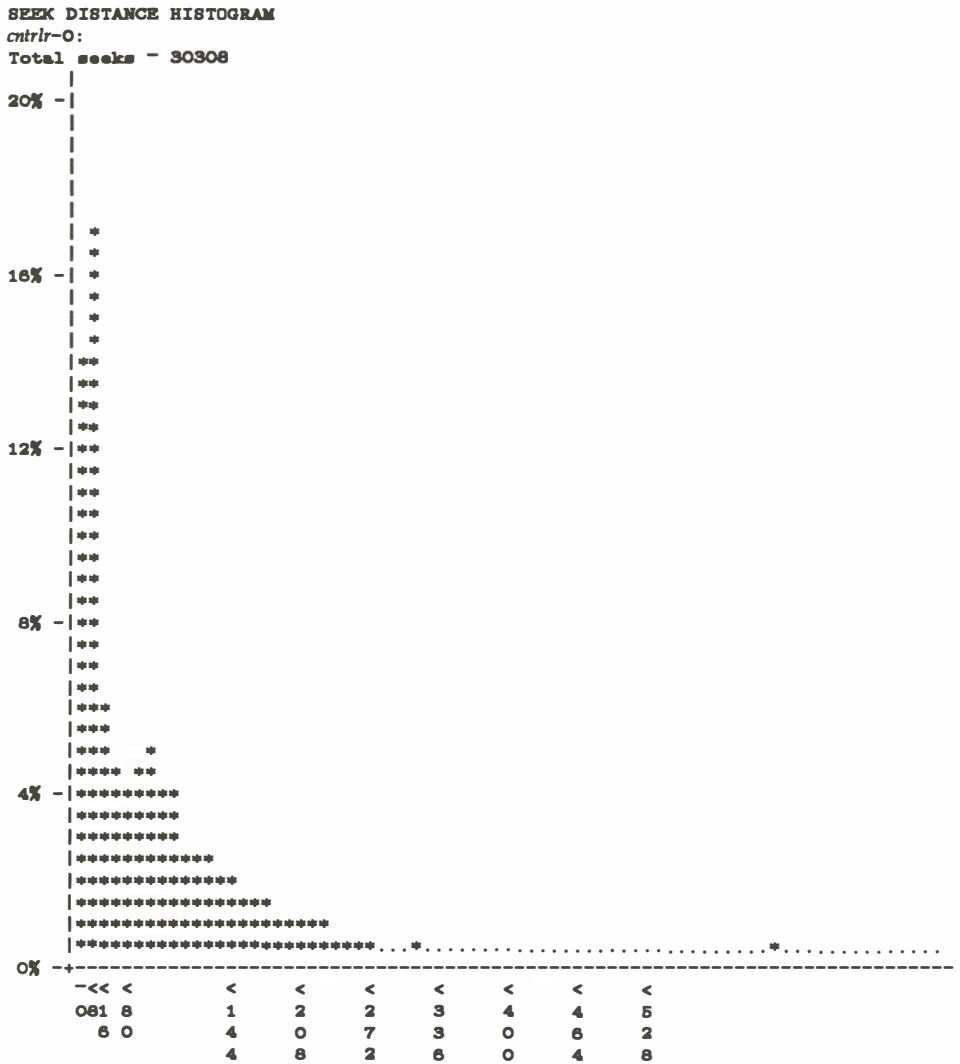


Figure 6-3. Output from **sadp**: Seek Distance Histogram

Using the **sadp** output with the output of **/etc/mount**(1M) and a table of disk sections (see Appendix A for the default partitioning of your disk(s)), you can identify the file systems with a large amount of I/O activity. In general, try to move areas of high activity close together. This will reduce the number of seeks over large distances.

The first graph (Figure 6-2) shows excellent disk cylinder locality of references. This means that every time the location of a block was referenced by the disk head (for reading or writing), it mostly occurred in the same general region of the disk. In the example, most references (as indicated by percent times referenced) occur for files near cylinders 450 to 600, with a few for files around cylinder 250. There are a few references to other files on the disk, but they occur only a small percent of the time. This graph shows that the most often used files are grouped together in the same general region of cylinders on the disk; the more clustered the stars are on the histogram, the better. This means that the disk has an excellent file system configuration.

The second graph (Figure 6-3) shows another aspect of an excellent file system configuration: the head seek distance. This refers to the distance the disk head has to move from the current cylinder to the cylinder of the next block referenced. In the example, most physical seeks are under ten cylinders. Specifically, 14% of the seeks occurred within a distance of 0 to 8 cylinders, and 17% of the seeks occurred within a distance of 8 to 16 cylinders. This means that for approximately one-third of the disk activity, the disk head was forced to move no more that 16 cylinders to reference a given block. The more to the left the stars are grouped, the better.

These two graphs give an idea of how finely you can tune your system. If, after a working period of weeks or months, you can identify which file systems are consistently the most active, you might consider repartitioning your disks to achieve the maximum from disk access activity (see Chapter 4, *Disk Management*, and Chapter 5, *File System Administration*, for more information).

Tunable Parameters

Tunable system parameters are used to set various table sizes and system thresholds to handle the expected system load. Use caution should be used when changing these variables because such changes can directly affect system performance. For the most part, the initial tunable parameter values for a new system 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.

The following tables show the recommended tunable parameter values for a Release 3.2 system equipped with different sizes of Random Access Memory (RAM). These values are recommended for the "typical user;" you may need to change (tune) the values depending on your system usage. The parameters shown in the tables are defined, using the **sysgen** utility, in the following four files:

- **kernel**
- **msg**
- **sem**
- **shm**

The default parameter settings defined in the **sysgen** kernel file are delivered with the core package. Parameters for Remote File Sharing (RFS) and Network Services Extension (NSE) are present in the core package, but should be left at their default setting of 0; non-zero settings are automatically provided when the RFS and NSE packages are installed. Additional RFS tunables are described in Chapter 10, *Remote File Sharing*. The message, semaphore, and shared memory parameters are distributed with the Interprocess Communications Utilities, in **msg, sem**, and **shm**, respectively. The following notes apply to the tables in this section:

- The parameters are set to specific values, as defined in the appropriate **sysgen** file. The default value and the size in bytes for each entry are shown in the tables.
- A dash (–) is used in the size information to indicate parameters that set flags in the kernel. Parameters that set flags do not affect the size of the kernel when their values are changed; only the values of the specific flags are changed.

Table 6-1. Kernel Parameter Values

Parameter	Default Value	Size Per Entry In Bytes
BDFLUSHR	60	-
NBUF	calculated	1080
PUTBUFSZ	2000	1
NCALL	NPROC+50	16
NCLIST	125	72
NFILE	512	12
FLCKREC	50	28
NHBUF	calculated	12
NINODE	512	64
MAXUP	75	-
MAXSLICE	12	-
MAXUMEM	4096	-
NMOUNT	50	32
NAUTOUP	30	-
NOFILES	100	-
NREGION	300	44
NS5NINODE	512	64
NPBUF	50	56
NPROC	128	130
SHLBMAX	0	12xNPROC
CDLIMIT	32768	-

Table 6-2. Paging Parameter Values

Parameter	Default Value	Size Per Entry In Bytes
GPGSHI	256	-
GPGSLO	64	-
GPGSMASK	0x408	-
MAXSWAPLIST	8	-
MINARMEM	16	-
MINASMEM	16	-
SPTMAP	200	-
(VHANDL)	-	-
VHANDR	2	-
VHANDFRAC	10	-

Table 6-3. STREAMS Parameter Values

Parameter	Default Value	Size Per Entry In Bytes
MAXSEPGCNT	1	2048
NBLK4	128	50
NBLK16	128	61
NBLK64	256	110
NBLK128	64	174
NBLK256	16	302
NBLK512	100	558
NBLK1024	100	1070
NBLK2048	50	2094
NBLK4096	4	4142
NLOG	3	12
NMUXLINK	32	12
NQUEUE	300	36
NSTREAM	32	52
NSTREVENT	64	12
NSTRPUSH	9	-
NUMTIMOD	30	?
SPCNT	64	?
STRCTLSZ	1024	-
STRLOFRAC	80	-
STRMEDFRAC	90	-
STRMSGSZ	16384	-

Table 6-4. Interprocess Parameter Values

Parameter > Default	Per Entry Value	Size In Bytes
MESG	1	-
MSGMAP	100	8
MSGMAX	2048	-
MSGMNB	4096	-
MSGMNI	50	47
MSGSEG	1024	8
MSGSSZ	8	1024
MSGTQL	40	12
SEMA	1	-
SEMAEM	16384	-
SEMMAP	10	8
SEMMNI	10	32
SEMMNS	60	8
SEMMNU	30	8x(SEMUME)
SEMMSL	25	-
SEMOPM	10	8
SEMUME	10	8xSEMMNU
SEMVMX	32767	-
SHM	1	-
SHMBRK	512	-
SHMMAX	512*1024	-
SHMMIN	1	-
SHMMNI	30	48
SHMSEG	6	12xNPROC

Kernel Parameters

The following parameters are defined in the **sysgen** kernel file:

NBUF

specifies how many 1K system buffers to allocate. The operating system buffers form a data cache. The data cache is a memory array containing disk file information. Improvement in the hit rate of this cache increases with the number of buffers. Cache hits reduce the number of disk accesses, thus, improve overall performance. The entries are normally in the range of 100 to 600. Each buffer contains 1080 bytes. Hash buffers (NHBUF) and system buffers (NBUF) are automatically increased 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 is normally greater than 2 in the range of 10 to 70. The default value is NPROC+50. 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: Time-out 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 default value is 512.

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 from 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 displays on the system console:

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

NS5INODE
NS5INODE must always be equal to or greater than NINODE. The default value is 512.

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 FILE 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 displays on the system console:

```
NOTICE: file table overflow
```

Note that this parameter does not affect the number of open files per process (see the NOFILES parameter). The default value is 512.

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**(2) system call returns the error EBUSY. Since the mount table is searched linearly, this value should be as low as possible. The default value is 50.

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). When full, the **fork**(2) system call returns the error EAGAIN. The NPROC entry is usually in the range of 50 to 200. The default value is 128.

NREGION
specifies how many region table entries to allocate. Each NREGION entry contains 44 bytes. Most processes have three 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 is 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 (the default value is 300). If the system runs out of region table entries, the following message displays 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. The default value is 125.

MAXUP

specifies how many concurrent processes a non-superuser is allowed to run. The entry is normally in the range of 15 to 25. 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. The default value is 75.

NOFILES

specifies the maximum number of open files per process; default is 100. Values higher than 20 are accessible only to processes using system calls (**open**(2), **creat**(2)). Processes using standard I/O subroutines are limited to 20, independent of the value of NOFILES. Unless an application package recommends that NOFILES be changed, leave the default setting of 100.

/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, 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 (100), 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 input/output buffers to allocate. One input/output buffer is needed for each physical read or write active. Each entry contains 56 bytes. The default value is 50.

NAUTOUP
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. The default value is 30.

BDFLUSHR
specifies the rate in seconds for checking the need to write the file system buffers to disk. The default is 1 second.

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. Each entry contains 28 bytes. The default value is 50.

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**(1M). The default value is 2000.

NODE
specifies the node name of the system. The default node name is **unix** (see Procedure 1.3, *Establish or Change System and Node Names*).

SYS
specifies the system name. The default system name is **unix** (see Procedure 1.3, *Establish or Change System and Node Names*).

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, defined in the **sysgen** kernel file, is normally one second (60 clock ticks on the computer). The default value is 12.

CDLIMIT
specifies in 512-byte blocks the size of the largest file that an ordinary user may write. The default size is 32768; i.e., the largest file an ordinary user may write is 1Mb. The super-user may write a file as large as the file system can hold. The CDLIMIT parameter does not apply to reads: any user may read a file of any size.

MAXUMEM
specifies the maximum size of a user's virtual address space in pages. This value cannot be greater than 32768 (for 1Kb pages) or 8192 (for 4Kb pages). The default is 4096.

Paging Parameters

A paging daemon, **vhand**, exists in the system, whose sole responsibility is to free up memory as the need arises. It uses a "least recently used" algorithm to approximate process working sets, and it writes those pages out to disk that have not been touched during some period of time. The page size is 1024 bytes. When memory is tight, the working sets of entire processes may be swapped out.

The following tunable parameters determine how often **vhand** runs and under what conditions. The default values in the **sysgen** kernel file should be adequate for most applications.

VHNDFRAC
used to determine 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 every VHANDR seconds. If free memory is less than VHANDL, then the paging daemon **vhand** is awakened.

The default for VHNDFRAC is 10. Decrease the value to make the daemon more active; increase the value to make the daemon less active (must be > 0 and < 25 percent of available memory).

VHANDR
specifies in seconds the maximum rate at which **vhand** can run. **vhand** will only run at this rate if free memory is less than VHANDL, as explained for VHNDFRAC. The default is 2. Increase the value to make the daemon less active (must be an integer > 2 and ≤ 300). If you have set the value higher, decreasing it makes the daemon more active.

GPGSLO
specifies the low water mark of free memory in pages for **vhand** to start stealing pages from processes. The default is 64. 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 256. Increase the value to make the daemon more active; decrease the value to make the daemon less active. (The value must be an integer > GPGSLO and < 25 percent of the number of pages of available memory.)

GPGSMSK
mask used by the paging daemon to determine the required number of aging passes before stealing a page. The default is 408. This value should not be changed.

MAXSWAPLIST
specifies the maximum number of pages that is swapped out in a single operation. The default value is 8.

MINARMEM
specifies the minimum number of memory pages reserved for the text and data segments of user processes. The default is 16.

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). The default is 16.

Streams Parameters

The following tunable parameters are associated with Streams processing. These parameters are defined in the **sysgen** kernel file. The values should be left at 0 unless the NSE package has been installed or you are planning on using STREAMS.

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 8*NSTREAM. The default value is 300.

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 application-dependent, but a value of 32-40 usually suffices for running a single transport provider with moderate traffic. The default value is 32.

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**(2) 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 is the expected number of processes to be simultaneously using **poll**(2) 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 64. Note that this number is not necessarily a hard upper limit on the number of event cells that is 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. Each new page can provide 166 event cells. 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 multiplexor link (STREAMS I_LINK **ioctl**). This number is application dependent; the default allocation equal to the number of Streams (NSTREAM) guarantees availability of links. The default value is 32.

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**(2) or **putmsg**(2) can consume an inordinate number of message blocks. The recommended 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**(2) message is not subject to the constraints of the min/max 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. The default value is 16384.

NBLK*n*

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 NSE package are intended to support a moderately loaded configuration using RFS and UUCP/CU over a network.

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 80 and there are 48 256-byte blocks, a low-priority allocation request will fail when more than 38 256-byte blocks are already allocated. The parameter is used to help prevent deadlock situations by starving out low-priority activity. The recommended value of 80 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). 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.

Log Driver Parameters

The configurable parameters for the log driver found in the **sysgen log** file are:

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**(1M)) and a trace command (**strace**(1M)), 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.

BSIZE
This number controls the number of log messages that will be kept in the log driver after being sent upstream to the error or trace logging process. Duplicates or only the last BSIZE messages are kept at the log driver; older messages are freed as new ones are submitted. The intent of this feature is to hold onto the last few log messages in case a system crash prevents their being written to the log file. A system dump analysis tool can then be used to look at these messages to determine if they point to the cause of the crash. The recommended number of 20 is more than sufficient for most burst-traffic situations.

Message Parameters

The tunable parameters in this section are associated with interprocess communication messages. These parameters are defined in the **sysgen msg** file. The order in which they are described follows the order in which they are defined in the output of the **/etc/sysdef** command.

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. The maximum size is 64 kilobytes –1.

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, i.e., 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 tunable parameters in this section are associated with interprocess communication semaphores. These parameters are defined in the **sysgen sem** file. The order in which they are described follows the order in which they are defined in the output of the **/etc/sysdef** command.

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 x (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**(2) 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*(SEMMNU) bytes.

SEMVMX
specifies the maximum value a semaphore can have. The default value is 32767. The default value 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**(2), 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 in this section are associated with interprocess communication shared memory. These parameters are defined in the **sysgen shm** file. The order in which they are described follows the order in which they are defined in the output of the **/etc/sysdef** command.

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 30. Each entry contains 48 bytes.

SHMSEG
specifies the number of attached shared memory segments per process. The default value is 6. The maximum value is 15.

Remote File Sharing Parameters

RFS parameters are discussed under *Parameter Tuning* in Chapter 10, *Remote File Sharing*. NSRMOUNT is included in that discussion because it is located in the **sysgen** kernel file. The values associated with NSRMOUNT are 0 unless the RFS package is installed. The rest of the RFS parameters are located in the **sysgen du** file.

7 LP Print Service Administration

Introduction

This chapter discusses the following

- how the LP Print Service works
- the commands used to administer the LP print service
- how to stop and start the LP Print Service
- how to configure the LP Print Service:
  - — how to set up printer configurations
  - — how to manage the printing load
  - — how to set job priority limits for users
  - — how to manage pre-printed forms
  - — how to define filters
- LP Print Service files and directories
- How to write customized filters and interface programs

Error messages issued by the LP Print Service are listed in Appendix C.

How the LP Print Service Works

The LP print service, originally called the LP spooler, is a mechanism that allows you to send a file to be printed while you continue with other work. The term "spool" is an acronym for *simultaneous peripheral output on-line*; "LP" originally stood for Line Printer, but has come to include many other types of printing devices. The LP Print Service system software:

- 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. These are shown in Table 7-1.

Table 7-1. LP Print Service User Commands

Command	Description
cancel(1)	Cancel a request for a file to be printed.
lp(1)	Send a file or files to a printer.
lpstat(1)	Report the status of the LP system.

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. 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 Table 7-2.

Table 7-2. Privileged User Commands for the LP Print Service

Command	Description
disable(1)	Deactivate the named printer(s).
enable(1)	Activate the named printer(s).

Summary of Administrative Commands

A separate set of commands available for the LP administrator is shown in Table 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 either as **root** or **lp**. **lp** is a system login (see Chapter 1, *System Security*, or Procedure 1.4 for a description of how to set up a password for a system login).

You will also need to use the commands for disabling and enabling a printer, and the commands described in the previous *Summary of User Commands* section.

Table 7-3. LP Print Service Administrative Commands

Command	Description
/usr/lib/accept(1M)	Permit job requests to be queued for a specified destination.
/usr/lib/reject(1M)	Prevent jobs from being queued for a specified destination. Described on the same manual page as **accept**(1M).
/usr/lib/lpadmin(1M)	Set up or change printer configurations.
/usr/lib/lpfilter(1M)	Set up or change filter definitions.
/usr/lib/lpforms(1M)	Set up or change preprinted forms. (Use **/usr/lib/lpadmin**(1M) to mount a form.)
/usr/lib/lpmove(1M)	Move output requests from one destination to another. Described on the same manual page as **lpsched**(1M).
/usr/lib/lpsched(1M) **/usr/lib/lpshut**(1M)	Start the LP Print Service. Stop the LP Print Service. Described on the same manual page as **lpsched**(1M).
/usr/lib/lpusers(1M)	Set or change the default priority and priority limits the users of the LP Print Service can request.

In Table 7-3 the administrative commands are listed in the order in which they appear in the *System Administrator's Reference Manual*. In the sections that follow, the commands are described 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 operating system is started, and stopped each time the operating system is stopped. However, if you need to stop the LP Print Service without stopping the operating system as well, you can do so by following the procedure described below.

Stopping the LP Print Service causes all printing to cease within seconds. Any print requests that have not finished printing are 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 are restored after it is restarted.

NOTE

To manually start and stop the LP Print Service you must be logged in as either the super-user **root** or the user **lp**.

Manually Stopping the Print Service

To manually stop the LP Print Service, enter the following command:

/usr/lib/lpshut

The following message appears and all printing ceases within a few seconds:

```
Print services stopped
```

If you try to stop the LP Print Service when it is not running, you 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 following message appears:

```
Print services started
```

It may take a minute or two for the printer configurations, forms, and filters to be reestablished, 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 must define the configuration of each printer you have. This section describes how to do that function.

Defining the Configuration of a Printer

The following 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. The more information you provide, however, the better the printer will be managed for you and the better it will be represented to the users using the LP Print Service.

The descriptions in the following sections 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 are also shown how to specify this information when adding a printer. While you can follow each section in order and correctly configure a printer in several steps, you may want to wait until you 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 in the following section) 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 users 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, e.g. **laser1**, **laser2**.

You do not 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). You also do not have to make the name precisely identify the type of printer—users who need to use a particular type of printer can specify it by type, not name (see the *Printer Type* section).

You use the printer name every time you want to refer to the printer, e.g., when adding other configuration information for the printer, changing the configuration of the printer, or referring to the status of the printer. Thus, the first thing you must do to add a printer is identify its name. You do this as shown below; but do not do it yet because you 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 must 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 (BNU) 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 BNU 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 SYSTEM V/88, 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 9, for details about setting up network connections.)

Adding a Directly Connected Printer

To add a directly connected printer enter:

/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/contty
/dev/tty11
/dev/tty12
/dev/tty13
/dev/tty14
/dev/tty15

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 enter:

/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 enter:

/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 a **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*), the alert message gives the reason for the fault. These messages are identical to the error messages produced by the BNU for similar problems. See the section called *BNU STATUS Error Messages* in Appendix C (*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, then specify it when you add a new printer. See *Customizing the Print Service* section for details on how to customize an interface program.

If you will be using the standard interface program, you do not need to 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.

NOTE

You can also specify an interface program by naming a model interface program. The models provided with the old LP Spooling Utilities have been carried over, *but they do not support all the new features.* These models will be phased out in a future release.

To identify a customized interface program by name, give the printer name and the path name of the interface program:

/usr/lib/lpadmin –p *printer-name* **–i** *path-name*

To identify a customized interface program by reference to another printer, give the printer names:

/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:

/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 —for example, **495** for the AT&T 495 Laser Printer. The *Acceptable Terminal Names* section of 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:

/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 content 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 users 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 users 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 do not, the printer type is used as the name of the content type the printer can handle. If you have not specified a printer type, the LP Print Service assumes the printer can print only files of content type **simple**. This may be sufficient if you require users to pick the proper printer and make sure the files are properly prepared for the printer before submitting them for printing.

One type of file often encountered on SYSTEM V/88 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; by default, it is spaced every eight columns on most printers

linefeed
: moves the carriage to the beginning of the next line (may require special port settings for some printers —see *Printer Port Characteristics*)

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 actions similar to those performed by a carriage 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 *do not* 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:

/usr/lib/lpadmin –p *printer-name* **–I** *content-type-list*

The *content-type-list* is a comma or space separated list of names. 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 following lists the default characteristics applied by the standard interface program:

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 lower-case to upper-case
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 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 must 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 users 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:

/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 are discussed.

When you list the print wheels or character sets available, you must assign 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 SYSTEM V/88 Release 3.2 Version 1 or a later release, 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**

where:

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 *Printer Type* section.

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*

The *print-wheel-list* is a comma or space separated list of names. 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*

The *character-set-list* is also a comma or space separated list; however, each item in the list looks like one of the following:

cs*N*=*character-set-name*
character-set-name$_1$=*character-set-name*$_2$

The *N* in the first case is a number from 0 to 63 that identifies the number of the character set in the Terminfo database. The *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, the LP Print Service assumes the following: a printer that takes print wheels has only a single, fixed print wheel, and users 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 listed the print wheels allowed on each, users can 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 stays queued and is not 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 Reference Manual* for a description of mail in SYSTEM V/88.
- You can receive an alert written to any terminal on which you are logged in. See the description of the **write** command in the *User'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 whether any print requests have not printed because the proper print wheel is not 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 choose to receive only one alert per print wheel.

To arrange to be alerted 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 the *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 directs the LP Print Service to never send you an alert when the print wheel needs to be mounted. The *integer* is the number of requests that need to be waiting for the print wheel; the *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/lpadmin –S *print-wheel-name* **–A quiet**

Once the print wheel has been mounted and unmounted again, alerts start again if too many requests are waiting. Alerts 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 uses 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 does not refuse such an attempt. The LP Print Service does, however, refuse a users 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 is 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 Reference Manual*. Briefly, the rules are:

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 that you have been denied permission to use.

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*

The *form-list* is a comma or space separated list of names of forms. 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 the use of all forms permissible, specify "**allow:all**"; to deny permission for all forms, "**deny:all.**"

If you do not add forms to the allow list or deny list, the LP Print Service considers that the printer denies the use of all forms. It does, however, allow you to mount any form. It also provides a warning message if the form is not in the allow list or if you are attempting to mount a form that does not 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 to them. Faults can range from simple problems, e.g., running out of paper or ribbon, or needing to replace the toner, to more serious faults, e.g., a local power failure or a 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: 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 Reference Manual* for a description of mail on SYSTEM V/88.
- 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 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 need a way of finding out about the faults and fixing them; the LP Print Service does 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 assumes 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 do 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 fails, you are not notified. Receiving repeated alerts per fault, or requiring manual re-enabling of the printer (see the *Fault Recovery* section), overcomes 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 the *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 directs the LP Print Service not to 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 only, by giving the following command:

/usr/lib/lpadmin –p *printer-name* **–A quiet**

If the *printer-name* is **all** in any of the commands above, the alerting condition applies to all printers.

If you do not define an alert method, you receive mail once for each printer fault. If you do define a method but do not give the **–W** option, you are 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 recovers in one of three ways:

- it continues printing at the top of the page where printing stopped
- it restarts printing at the beginning of the print request that was active when the fault occurred
- it waits 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 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 does not have this capability. If a proper filter is not being used, you are 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 tries 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 is 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 users 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 users can print sensitive information, or if use of a high quality printer incurs expenses not all users are allowed to incur.

The LP Print Service uses the list of users allowed or denied for a printer to restrict use of the printer. The LP Print Service refuses a users 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:

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 super-user **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*

The *user-list* is a comma or space separated list of names of users. 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** allows everybody, using **deny:all** denies everybody.

If you do not add user names to the allow or deny lists, the LP Print Service assumes that everybody can use the printer.

Banner Necessary

Most users want to have the output of each print request preceded by a banner page. A banner page shows who requested the printing, the request ID for it, and when the output was printed. It also allows for an optional title that the requester can use to better identify a printout. Finally, the banner page greatly eases the task of separating a sequence of print requests so that each can be given to the correct user.

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 under such circumstances 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 reverse 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 rejects all attempts to avoid a banner page when printing on the printer. This is the default action.

Description

An easy way to give users of the LP Print Service helpful information about a printer is by adding a description of it. This description can contain any message, e.g., the number of the room where the printer is found, the name of the person to call with printer problems.

Users can see the message when they 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'*

The *text* is the message. You must 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 is presented to users who ask about it.

Default Printing Attributes

When a user submits a request to print a file, page size, character pitch, and line pitch (i.e., print spacing) are normally determined from the form on which it is printed. If the user does not require a form, the user can specify the page size and print spacing to be used. However, if the user does not specify a form or the page size and print spacing, defaults are used.

You can set the defaults for each printer. Doing so can make it easier to submit print requests by allowing you to designate different printers as having different default page sizes or print spacing. A user can then simply route a file to the appropriate printer to get a desired style of output. For example, you can have one printer dedicated to printing wide (132-column) output, another printing normal (80-column by 66-line) output, and yet another printing letter quality (12 characters per inch, 8 lines per inch) output.

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, or 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 the *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 do not provide defaults, the page size and print spacing are be those available when the printer is initialized. You can find out what the defaults are by first defining the printer configuration without providing your own defaults, then using the **lpstat** program to display the printer configuration. The following command reports the default page size and print spacing:

lpstat –p *printer-name* **–l**

If you have not provided the defaults, the reported defaults are 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; the LP Print Service picks the first printer in the class that it finds free. This allows faster turn-around because printers are kept as busy as possible.

Classes are not 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 is used to print the file. The LP Print Service avoids using a filter, if possible, by choosing a printer that can print the file directly over one that would need it filtered first. See the *Filter Management* section of this chapter for more information about filters.

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 do not set a default destination, there is none. Users must explicitly name a printer or class in each print request, or set the LPDEST shell variable with the name of a destination.

Mounting a Form or Print Wheel

NOTE

See the section in this chapter for information about preprinted forms.

Before the LP Print Service can start printing files that need a preprinted form or print wheel, you must mount the form or print wheel on a printer. If alerting has been set on the form or print wheel, you are alerted when enough print requests are queued waiting for it to be mounted.

When you mount a form you may want 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 have 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 is currently printing, and because the LP Print Service continues to print files not needing the form on the printer, you must disable the printer first. The following is the proper procedure to do this:

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 have 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 the **–S** *print-wheel-name* if you are mounting just a form; 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 are not asked to press the key. You can drop the **–a** and **–o filebreak** options if you do not want to use to use 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, omit 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 the **–S none** if you just want to unmount a form; likewise, omit **–f none** if you just want to unmount a print wheel.

Until you have mounted a form on a printer, only print requests that do not require a form are sent to it. Likewise, until you have mounted a print wheel on a printer, only print requests that do not require a particular print wheel are 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 must 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 to remove a printer from a class without deleting that printer, enter the following command:

```
/usr/lib/lpadmin –p printer-name –r class-name
```

It is possible to add a new printer shown in the commands described above. You may find it easier, however, to enter one or two commands that combine all the necessary arguments. The following are some examples:

Example 1

Add a new printer called **lp1** on printer port **/dev/tty13**. 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. (The example is split into two lines for readability.)

```
/usr/lib/lpadmin –p lp1 –v /dev/tty13 –T 455 –o "width=90 length=71
stty=-onlcr"
```

Example 2

Add a new printer called **laser** on printer port **/dev/tty41**. It should use a customized interface program, it can handle three file types —**i10**, **i300**, and **impress** —and it may be used only by the users **doceng** and **docpub**. (This example is split into two lines for readability.)

```
/usr/lib/lpadmin –p laser –v /dev/tty41 –I /usr/doceng/laser_intface
        –I "I10,I300,Impress" –u "allow:doceng,docpub"
```

Example 3

When adding the **lpa1** printer in the first example, alerting was not set; do this now. You want the LP Print Service to alert you 10 minutes after a fault until you fix the problem.

```
/usr/lib/lpadmin –p lp1 –A write –W 10
```

Accepting Print Requests for a New Printer

Initially, the LP Print Service does not consider a new printer eligible for printing files. This gives you time to make sure you have defined the printer configuration the way you want. When you are ready to make the printer available to others, you must tell the LP Print Service.

There are two steps in making a printer ready for use after you have defined the printer configuration. First, the LP Print Service must be told to accept print requests for the new printer. Second, the new printer must 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.

Enabling and Disabling a Printer

When a printer is ready for use and the LP Print Service is accepting print requests for it, you must enable the printer before anything can be printed. 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, the correct print wheel or font cartridge is in place, and the printer is online.

When all is ready, 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. Do not 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 want to interrupt a print request. Disabling a printer stops further print requests from being printed, but it does not stop the LP Print Service from accepting new print requests for the printer. Normally, disabling a printer also stops the request that is currently being printed, 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 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 makes the LP Print Service wait for the current request to finish, while the third cancels the current request. The *reason* is stored and displays whenever a user asks the status of the printer. You can omit it (and the **-r**) if you don't want to specify 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 availability 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, use a standard system feature called the "setuid bit." By assigning ownership of these commands to the user **lp** (this should have been done automatically when you installed the software), and by setting the setuid bit, you can make sure that anyone will be allowed to use the **enable** and **disable** commands. Clearing the bit removes this privilege.

To allow everybody to run **enable** and **disable**, 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 step should be redundant, but it is safer to run the command than to skip it. The second command turns on the setuid bit.

To prevent others from running **enable** and **disable**, enter the following command:

```
chmod u-s /usr/bin/enable /usr/bin/disable
```

Examining a Printer Configuration

Once you have defined a printer configuration, you probably want to review it to see if it is correct. If after examining the configuration you find you made a mistake, just reenter the command that applies to the part that is wrong.

Use the **lpstat** command to examine both the configuration and the current status of a printer. The 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. The long form of the command requests a complete configuration listing.

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 tries again later.

With the long form of the command, you should see output such as the following:

```
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
```

Troubleshooting

If you are having difficulty getting your printer to work, the following provides a few suggestions of 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?

The type of connection between a computer and a printer may vary. Your printer manual should provide detailed instructions for connecting most printers and for particular applications.

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 haven't specified a printer type for the printer, omit the **–T** *printer-type* option.

Is the Baud Rate Correct?

If the baud rate (the rate at which data is transmitted) is not the same for both the computer and the printer, sometimes nothing will print (see the following sections.)

Illegible Output

The printer tries printing, but the output is not what you expected; it is not readable.

Is the Baud Rate Correct?

Usually, when the baud rate of the computer does not match that of the printer, you receive some output but it does not look at all like what you submitted for printing. Random characters appear with an unusual mixture of special characters and unlikely spacing.

Read the documentation that came with the printer to find out what is the correct baud rate. It should probably be set at 9600 baud for optimum performance, but that doesn't matter for now. If it is not 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 does not 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 received for printing has not been garbled in transmission. The parity bit can be encoded in several ways; the computer and the printer must agree on which one to use. If they do not agree, some characters either are not printed or are replaced by other characters. Generally, however, the output looks 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 the printer expects. The LP Print Service does 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 following sample command:

/usr/lib/lpadmin –p *printer-name* **–o "stty='evenp 1200'"**

Both double and single quotes are needed.

Tabs Set Correctly?

If the printer does not expect to receive tab characters, the output may contain the complete content of the file, but the text may appear in a chaotic looking format, jammed up against the right margin. (See the following sections.)

Correct Printer Type?

See the *Wrong Character Set or Font* section.

Legible Printing, but Wrong Spacing

The output contains all the expected text and is readable, but the text appears in an undesirable format: double spaced, with no left margin, run together, or zig-zagging down the page. These problems can be fixed by adjusting the printer settings (if possible) or by having the LP Print Service use settings that match those of the printer.

Double Spaced

Either the printers tab settings are wrong or the printer is adding a linefeed after each carriage return. (The LP Print Service has a carriage return added to each linefeed, so the combination causes two linefeeds.) You can have the LP Print Service not send tabs or not add a carriage return by using the **-tabs** option or **-onlcr** option, respectively.)

/usr/lib/lpadmin –p *printer-name* **–o stty=-tabs**
/usr/lib/lpadmin –p *printer-name* **–o stty=-onlcr**

No Left Margin/Runs Together/Jammed Up

The printers tab settings are not correct; they should be set every 8 spaces. You can have the LP Print Service not send tabs by using the **–tabs** option:

/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 you need to use several of these options to take care of multiple problems, you can combine them in one list as shown in the following sample command. Also include any baud rate or parity settings.

/usr/lib/lpadmin –p *printer-name* **–o "stty='–onlcr –tabs 2400'"**

Both double and single quotes are needed.

Correct Printer Type?

See the following section.

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 to be illegible, making it look like a problem described above. Another result may be that the wrong control characters cause the printer to set the wrong character set or font.

If you do not 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 SYSTEM V/88.) The output of this command will appear on your terminal: 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 do not know what names to try, you can examine the **/usr/lib/terminfo** directory to see what names are available. 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 name 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 BNU 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, they are mailed, by default, to the user **lp**.) See the *BNU STATUS Error Messages* section of Appendix C in this manual for an explanation of the failure reasons.

Idle Printers

There are several reasons why you may find a printer idle and enabled but with print requests still queued for it:

- 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. After a fault has been detected, printing resumes automatically, but not immediately. The LP Print Service waits about five minutes before trying again, continues trying until a request is printed successfully. You can force a retry immediately by enabling the printer:

  enable *printer-name*

- A dial out printer is busy or doesn'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 cannot be reached for an hour or two (depending on the reason), the LP Print Service finally alerts you to a possible problem. You can force a retry immediately by enabling the printer:

  enable *printer-name*

- Lost "child process." If the operating system process controlling the printer is killed (by operating 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 forces 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:

  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:

- printer needs periodic maintenance
- printer is broken
- printer has been removed
- you 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, e.g., stopping its ability to handle a certain form, changing the print wheels available, 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 does not care which printer is to be used. Such requests are not automatically moved; if you do not move them first, the LP Print Service cancels them.

If you decide to take a printer out of service, to change its configuration, or to lighten its load, you may want to move print requests off it and reject additional requests for it for awhile. To do so, use the **lpmove** and **reject** commands. If you do reject requests for a printer, you can accept requests for it later, by 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* displays whenever a user tries to print a file on the printer. You can omit it (and the **–r**) if you do not want to specify 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 be printed 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 must always 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 $printer\text{-}name_1$ $printer\text{-}name_2$

You can give more than one request ID before the printer name in the first command.

The first command moves the listed requests to the printer named. The latter command moves from the first printer to the second printer *all* requests currently queued for the first printer. When the latter command is used, the LP Print Service also stops accepting requests for the first printer (the same result you obtain by running the **reject** command).

Examples

The following are some examples of how you might use these three commands.

Example 1

You decided it is time to change the ribbon and perform some preventive maintenance on printer **lp1**. First, to prevent the loss of print requests, you move all requests for printer **lp1** to printer **lp2**. After the requests are moved, the LP Print Service no longer accepts requests for **lp1** (the same result you would obtain by running the **reject lp1** command before the **lpmove** command):

```
/usr/lib/lpmove lp1 lp2
```

(At this point you may disable the printer and start working on it.)

Example 2

You finished changing the ribbon and doing the other work on **lp1**; now it is time 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 *Managing Printers* in this chapter.)

Example 3

You notice that a user has queued several large files for printing on the printer **laser1**. Meanwhile, **laser2** is idle because no one has queued requests for it. 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 users can use to adjust the position of a print request in the queue. Each print request can be given a priority level by the user 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).

Thus, for example, a user who decides that their print request is of low priority can assign it a larger value when submitting the file for printing. Another user who decides that their print request is of high priority can assign it a smaller value when submitting the file for printing.

A priority scheme this simple would not work if there were no controls on how high a user can set the priority. You can define the following characteristics of this scheme:

- Each user can be assigned a priority limit. A user cannot submit a print request with a priority higher than their limit, although the user 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 (e.g., regular printing by most staff members) from interfering with higher priority printing tasks (e.g., 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 preempt the currently printing request. You can have the LP Print Service give "immediate" handling to a print request, and have it put on "hold" another print request. This allows the first request to be printed and delays the latter print request until you allow it to be "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 to move it up for immediate printing, respectively. These commands are discussed in detail in the following section.

Setting Priority Limits

To set a users 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 a 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 users for whom you have not set a personal limit, using the first **lpusers** command.

If you later decide that a user should have a different priority limit, just reenter the first command above with a new limit. Or, if you decide that the default limit is more appropriate for a user who already has a personal limit, enter the following command:

/usr/lib/lpusers –u *user-name*

Again, you can do this for more than one user at a time by including a list of names. Using the **lpusers** command with just the **–u** option puts the users in the "default limit" category.

Setting a Default Priority

To set the default priority (the priority level assigned to print requests submitted without a priority), use the following command:

/usr/lib/lpusers –d *priority-level*

Do not confuse this default with the "default limit." This default is applied when a user does not specify a priority level; the "default limit" is applied if you have not assigned a limit for a user —it is used to limit the user from requesting 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 a 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 allow other requests to be printed ahead of it
- you can put it at the head of the queue for immediate printing.

Use the **lp**(1) command to do any of these tasks.

Changing the Priority for a Request

If you want to change the priority of a particular request that is still waiting to be printed, you can assign a new priority level to it. By doing so, you can move it in the queue so that it is ahead of lower priority requests, and behind requests at the same level or of higher priority. The priority limit assigned to the user (or the default priority limit) has no effect because, as the administrator, you can override this limit.

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.

Putting a Request on Hold

Any request that has not finished printing can be put on hold. This will stop its printing, if it is currently 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 that has been put on hold by the administrator.

To place a request on hold, enter the following command:

lp –i *request-id* **–H hold**

Enter the following command to resume the request:

lp –i *request-id* **–H resume**

Once resumed, a request continues to move up the queue and is eventually printed. If printing had already begun when you put it on hold, it is the next request printed. Normally, printing begins on page one, but, if you prefer, you can have printing begin on a later page. Enter the following command to resume the request on 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 do so; the default filter used by the LP Print Service does not. An attempt to resume a request on a later page is rejected if an appropriate filter is not used.

Moving a Request to the Head of the Queue

You can move a print request to the head of the queue where it is the next one eligible for printing. If you want it to start printing immediately but another request is currently being printed, you can interrupt the first request by putting it on hold, 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, as the administrator, can move a request in this way; regular users cannot use the **–H immediate** option.

NOTE

If you set more than one request for immediate printing, the requests are printed in the reverse order set; i.e., the request moved to the head of the queue most recently is printed first.

Forms

This section tells you how you can manage the use of preprinted forms with the LP Print Service:

- 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, the following discusses what a form means in the context of the LP Print Service.

What is a Form?

A preprinted form is a blank form that you can load into your printer. Common examples of forms include:

- blank checks
- vouchers

- receipts
- labels
- company letterhead
- special paper stock

Typically, several copies of a blank form are loaded into a printer, either as a tray of single sheets or as a box of fan-folded paper. An application is used to generate a file that is printed on the form, thereby filling it out.

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, keeps track of which print requests need special forms mounted and which forms are currently mounted. It can alert you to the need to mount a new form.

Of course, if you do not use special forms for printing, you can skip this section.

Defining a Form

When adding a new form, the first thing you have to do is to define its characteristics. To do so, enter information about each of the nine required characteristics (e.g., page length, page width) as input to the **lpforms** command (see below for details). The LP Print Service uses this information for two purposes: to initialize the printer so that printing is done properly on the form, and to send you reminders about how to handle that form. Before running the **lpforms** command, gather the following information about your new 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 a length of one form. (See the description of alignment patterns.) If no filter is available, the LP Print Service does not truncate the output.

Line pitch
A measurement that shows how closely together separate lines appear on the form. It can be expressed in either lines per inch or lines per centimeter.

Character pitch
A measurement that shows how closely together separate characters appear on the form. It can be expressed in either cpi or cpc.

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 their print request when 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 users to see so they can understand what the form is, when it should be used.

Alignment pattern
A sample file that the LP Print Service uses 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 is not, you can adjust the printer to get it lined up.

NOTE

The LP Print Service does not try to mask sensitive information in an alignment pattern. If you do not want sensitive information printed on sample forms, (e.g., when you align checks), you should mask the appropriate data. The LP Print Service keeps the alignment pattern stored in a safe place, where only you (i.e., the user **lp** and the super-user **root**) can read it.

When you gather this information about the form, enter it as input to the **lpforms** command. You may want to record this information first in a separate file so you can edit it before entering it with **lpforms**. You can then use the file as input instead of typing each piece of information separately after a prompt. Whichever method you use, enter the information in the following format:

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

With two exceptions, the information can appear in any order. The exceptions are the alignment pattern (which must always appear last) and the *comment* (which must always follow the line with the `Comment:` prompt). If the *comment* contains a line beginning with a key phrase (e.g., **Page length**, **Page width**), precede that line with a ">" character so the key phrase is hidden. Be aware that any initial ">" is stripped from the comment when it displays.

Not all the information has to be given. When you do not specify values for the items listed below, the values shown beside them are assigned by default:

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. A *form-name* can be anything you choose, as long as it contains 14 or fewer letters, digits, and underscores.

If you need to change a form, just reenter one of the same commands. You need only provide information for items that must be changed; items for which you don't specify new information will stay the same.

Removing a Form

The LP Print Service imposes no fixed limit on the number of forms you can define. It is a good idea, however, to remove forms that are no longer appropriate. If you do not, users see a long list of obsolete forms when choosing a form, and may be confused. In addition, because the LP Print Service must occasionally look through all the forms listed before performing certain tasks, the failure to remove obsolete forms may require extra, unnecessary processing by the print service.

To remove a form, enter the following command:

/usr/lib/lpforms –f *form-name* **–x**

Restricting User Access

If your system has a form that you do not want to make available to all users, you can limit its availability to a subset of users on your computer. For example, you may want to limit access to checks to the users in the payroll department or accounts payable department.

The LP Print Service restricts the availability of a form by using the list of users allowed or denied access to that form. If a user is not allowed to use a particular form, the LP Print Service refuses the users request to print a file with it.

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 Reference Manual.)* Briefly, the rules are:

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 super-user **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 comma or space separated list of names of users. 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** allows all users; using **deny:all** denies all users.

If you do not add user names to the allow or deny lists, the LP Print Service assumes that everybody can use the form.

Alerting to Mount a Form

If you define more forms than printers, you cannot print files on all the forms simultaneously. This means that some print requests may be held in a queue until you mount the forms they need. How will you know when to mount a particular type of form? One method would be to periodically monitor the number of print requests pending for a particular form. The LP Print Service, however, provides an easier way: You can ask to be alerted when the number of requests waiting for a form has exceeded a specified 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 Reference Manual* for a description of mail on SYSTEM V/88).
- You can receive an alert written to any terminal on which you are logged in. (See the description of the **write** command in the *User'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 have not printed because the proper form is not 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 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 the *command* for each alert. The shell environment 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 form needs to be mounted. The *integer* is the number of requests that need to be waiting for the form; the *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, use the third command listed, specifying 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 issuing the following command:

/usr/lib/lpforms –f *form-name* **–A quiet**

Once the form has been mounted and unmounted again, alerts resume if too many requests are waiting. Alerts also start again if the number of requests waiting falls below the **–Q** threshold and then rises up to the **–Q** threshold again. This happens when waiting requests are canceled, and when the type of alerting is changed.

If *form-name* is **all** in any of the previous commands, the alerting condition applies to all forms.

If you do not define an alert method for a form, you do not receive an alert to mount it. If you do define a method but do not give the **–W** option, you are alerted once for each occasion.

Examining a Form

Once you've added a form definition to the LP Print Service, you can examine it with one of two commands, depending on the type of information you want to check. The **lpforms** command displays the definition of the form. (The display produced by **lpforms** can be used as input, so you may want to save it in a file for future reference.) The **lpstat** command displays 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 be used later 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**:

```
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
```

To protect potentially sensitive content, the alignment pattern is not shown if the **lpstat** command is used.

Filter Management

This section explains how you can manage the use of filters with the LP Print Service:

- 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 describes 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 performs one or more of three related roles:

- it may convert a users file into a data stream that can be printed properly on a given printer
- it may handle the special modes of printing that people may request with the **–y** option to the **lp** command (e.g., two-sided printing, landscape printing, draft or letter quality printing)
- it may detect printer faults and notify the LP Print Service of them, so that the print service can alert you.

Not every filter performs all three roles. Given the printer-specific nature of these three roles, the LP Print Service has been designed so that these roles can be implemented separately. This separation allows you, a printer manufacturer, or another source to provide 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 or handle any of the special modes. It may, however, be adequate for your needs.

The following examines the three roles of a filter more closely.

Role 1: Converting Files

The LP Print Service allows you to classify each printer you add to the system as a particular "type," and allows a user to identify each file submitted for printing as a "type." The type information is used to match a file with the printer that will best reproduce the file. Because many applications can generate data for various printers, this is often sufficient. However, some of the applications you use may not be able to generate output that can be printed by your printers.

By defining and creating a filter that converts 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, e.g., **nroff** to data streams that can be printed properly on some printers.

For each filter that is added to the system, the type of input it can accept and the type of output it can produce are listed. Now the LP Print Service can be more sophisticated in its attempt to match a users file with a printer. If it cannot make a direct match, it consults the table of filters to find one that can convert the files type into the printers type. The following are some examples.

Example 1

The user Chris ran a spreadsheet program and generated a file containing a 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 knows it is necessary to request output that can be handled by the Model 455. When Chris submits the file for printing, the LP Print Service queues 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 large document. Forgetting the fact that the **nroff** program understands how to generate output for several printers, Marty requests the default output type (call it type **nroff35**) which cannot be reproduced well on the Model 455. Fortunately, you have foreseen this situation and have added the **450** filter from the Terminal Filters Utilities to the filter table, marking it as a filter that takes standard **nroff** output (i.e., **nroff35**) and produces output for the Model 455 (call it type **455**). Because you have 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.

Role 2: Handling Special Modes

Another important role that filters can perform is the handling of special printing modes. Each filter you add to the filter table can be registered to handle special modes and other aspects of printing:

Special modes
Input type
Output type
Printer type
Character pitch
Line pitch
Page length
Page width
Pages to print
Character set
Form name
Number of copies

A filter is required only to handle special modes; the LP Print Service provides a default handling for 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 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 printed pages. As another example, some printers can handle multiple copies more efficiently than the LP Print Service, so a filter that can control the printer can use the information about the number of copies to skip the LP Print Services default handling of multiple copies.

The following describes how you can register special printing modes and other aspects of printing with each filter.

Role 3: Detecting Printer Faults

Just as converting a file and handling special printing modes is a printer-specific role, so is detecting printer faults. The LP Print Service attempts to detect faults in general, and for most printers it can do so properly. The range of faults that the print service can detect, however, is limited. It can check for "hang-ups" (loss of carrier, signal that indicates the printer is online) and excessive delays in printing (receipt of an XOFF flow-control character to shut off the data flow, with no matching XON to turn the flow back on). However, the print service cannot determine the cause of a fault, so it cannot tell you what to look for.

A properly designed filter can provide better fault coverage. Some printers can send a message to the host describing the reason for a fault. Others indicate a fault by using signals other than the dropping of a carrier or the shutting off of data flow. A filter can serve you by detecting more faults and providing more information about them than you would otherwise receive.

Another service a filter can provide is to wait for a printer fault to clear and then resume printing. This service 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 has knowledge of the control sequences used by a printer, can "know" where a file breaks into pages; thus, only such a filter can find the place in the file where printing should resume.

The LP Print Service has a simple interface that allows a filter to send you fault information and to restart printing 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 in an alert message that can be sent to you. Thus, you see any fault descriptions generated by the filter. If you set the printer configuration so that printing should automatically resume after a fault is cleared, the interface program keeps the filter active, so that printing can pick up where it left off.

Will Any Program Make a Good Filter?

It is tempting to use a program such as **troff**, **nroff**, or a similar word-processing program as a filter. However, the **troff** and **nroff** programs have a feature that allows references to be made in a source file to other files, known as "include files." The LP Print Service does not recognize include files; it will not enqueue any that are referenced by a source file when that file is in a queue to be printed. As a result, the **troff** or **nroff** program, unable to access the include files, may fail. Other programs may have similar features that limit their use as filters.

Here are a few guidelines for evaluating a filter:

1. Examine the kinds of files users will submit for printing that require processing by the filter. If they stand alone (i.e., 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 files other than those submitted by a user for printing.
2. If referenced files are permitted in the files submitted for printing, or if the filter needs, files other than those submitted by a user, then the filter, unable to access the additional files, is likely to fail. We suggest you do not use the program under consideration as a filter; instead, have users run the program before submitting files for printing.

Referenced files that are always specified by 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 exist only on the remote machine.

Defining a Filter

When adding a new filter, the first thing you must do is to define the characteristics of its use. There are seven characteristics you must define: input types, output types, printer type, printers, filter type, command, and options. Each is described in the following sections. This is the list of file types that the filter can process. The LP Print Service does not impose a limit on the number of input types that can be accepted by a filter, but most filters can take only one. Several file types may be similar enough so that the filter can deal with them. You can use whatever names you like here, subject to a limit of 14 letters, digits, and dashes (not underscores).

Because the LP Print Service uses 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 identify the type of a file when submitting that file for printing. This is the list of file types that the filter can produce as output. For each input type the filter will produce a single output type, of course; the output type may vary, however, from job to job. 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 match the input types handled by other filters. The LP Print Service groups 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 is unlikely that you need this level of sophistication, but the LP Print Service allows it. Try to find a set of filters that takes (as input types) all the different files your users may want printed, and converts those files directly into types your printers can handle. This is a list of printer types into which the filter can convert files. For most filters, this list is identical to the list of output types, but it can be different.

For example, you may have a printer that is given a single type for purposes of initialization (see the *Printer Type* section under *Printer Management* in this chapter), but can recognize several different types of files. In essence, this printer has an internal filter that converts the various types into one with which it 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 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, e.g., 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. You may have some printers that, although they are 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 are sent to that printer. Other printers, of identical type, you allow to be used for files that may need extensive filtering before they can be printed. In this case, you would label the filter as working with only the latter group of printers.

In most cases, the filter should be able to work with all printers that accept its output, so you can usually skip this part of the filter definition. 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 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 do not require access to the printer should be labeled "slow."

The LP Print Service runs slow filters in the background, without tying up a printer. This allows files that do not need slow filtering to move ahead; printers are not left idle while a slow filter works on a file if other files can be printed simultaneously. This is the full path name of the program run as the filter. If there are any fixed options that the program always needs, include them here. Options that the filter program needs, depending on the special modes and other aspects of printing, can be registered with the filter. This is discussed in more detail in the following section.

When you have gathered this information about the filter, enter it as input to the **lpfilter** command. You may want to record this information first in a separate file, so you can edit it before entering it with **lpfilter**. You can then use the file as input, instead of typing each piece of information separately, after a prompt. Whichever method you use, enter the information in the following format:

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. When you do not specify values for the items listed below, the values shown beside them are assigned by default.

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 default values define a very flexible filter, so you probably have to supply at least the input and output type(s). When you enter a list, you can separate the items in it with blanks or commas, unless it is a *templates-list*; items in a *templates-list* must be separated by commas.

Templates

The "*templates-list*" is a comma-separated list of templates of the following form.

keyword pattern = *replacement*

where:

keyword
labels the template as registering a particular characteristic of the printing.

pattern
is either a value of the characteristic or an asterisk (*), which serves as a placeholder for a value.

Only the *keywords* shown in the following table may be used.

Characteristic	Keyword	Possible *Patterns*
Special modes	MODES	*mode*
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*

The sources of the values for these templates are:

- The value of the MODES template comes from the **–y** option of the **lp** command (the command used to submit a print request). Because a user can specify several **–y** options, there may be several values for the MODES template. The values are applied in the left-to-right order given by the user.
- The values for 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. Each is 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 users request, the form being used, or the default values for the printer.
- The value for the PAGES template is a list of pages that should be printed. Typically, it is a comma separated list of page ranges, each of which consists of a dash separated pair of numbers or a single number (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 that should be made of the file. If the filter uses this template, the LP Print Service will reduce to 1 the number of copies of the filtered file *it* will have printed, since this "single copy" will really be the multiple copies produced by the filter.

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 place-holder * included to show where the value goes.

NOTE

If a comma or an equals sign (=) is included in a *pattern* or a *replacement*, escape its special meaning by preceding it with a backslash. A backslash in front of either character is removed when the *pattern* or *replacement* is used; all other backslashes are left alone.

Command to Enter

Once a filter definition is 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; the second command gets the filter definition from you, through the standard input. A *filter-name* can be anything you choose, as long as it contains 14 or fewer letters, digits, and underscores.

If you need to change a filter, just re-enter one of the same commands. You need only provide information for those items that must be changed; items for which you do not specify new information stay the same.

Removing a Filter

The LP Print Service imposes no fixed limit on the number of filters you can define. It is a good idea, however, 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.

To remove a filter, enter the following command:

/usr/lib/lpfilter –f *filter-name* **–x**

Examining a Filter

Once you have added a filter definition to the LP Print Service, you can examine it by running the **lpfilter** command. The output of this command is the filter definition displayed in a format that makes it suitable as input. You may want to save this output in a file that you can use later to redefine the filter if you inadvertently remove the filter from the LP print service.

To examine a defined filter, enter one of the following commands:

/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 for future reference.

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 all print requests 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 delays in the responses 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. These delays can become noticeable if there is a large number of requests that need to be filtered.

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, there may be 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 more or less frequently, or to condense the information into a report. (See *Cleaning Out the Request Log*.)

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	sys	**/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 table that starts on the following page. 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	sys	admins
drwxrwxr-x	lp	sys	bin
-rw-r--r--	lp	bin	* default
drwxrwxr-x	lp	sys	fifos
drwxrwxr-x	lp	sys	logs
drwxrwxr-x	lp	sys	model
drwxrwxr-x	lp	sys	requests
drwxrwxr-x	lp	sys	system
drwxrwxr-x	lp	sys	temp
-rw-r--r--	lp	bin	* users
/usr/spool/lp/admins:			
drwxrwxr-x	lp	sys	lp
/usr/spool/lp/admins/lp:			
drwxrwxr-x	lp	sys	classes
-rw-rw-r--	lp	bin	* filter.table
-rw-rw-r--	lp	bin	* filter.table.i
drwxrwxr-x	lp	sys	forms
drwxrwxr-x	lp	sys	interfaces
drwxrwxr-x	lp	sys	logs
drwxrwxr-x	lp	sys	printers
drwxrwxr-x	lp	sys	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*

Permissions	Owner	Group	Directory or File
/usr/spool/lp/admins/lp/forms/*formK*:			
-rwxrwx---	lp	bin	* alert.sh
-rw-rw----	lp	bin	* alert.vars
-rw-rw----	lp	bin	* align_ptrn
-rw-rw-r--	lp	bin	* allow
-rw-rw-r--	lp	bin	* comment
-rw-rw-r--	lp	bin	* deny
-rw-rw-r--	lp	bin	* describe
/usr/spool/lp/admins/lp/interfaces:			
-rwxrwxr-x	lp	bin	* *printer1*
-rwxrwxr-x	lp	bin	* *printer2*
.			
.			
.			
-rwxrwxr-x	lp	bin	* *printerN*
/usr/spool/lp/admins/lp/printers:			
drwxrwxr-x	lp	bin	* *printer1*
drwxrwxr-x	lp	bin	* *printer2*
.			
.			
.			
drwxrwxr-x	lp	bin	* *printerN*
/usr/spool/lp/admins/lp/printers/*printerK*:			
-rwxrwx---	lp	bin	* alert.sh
-rw-rw----	lp	bin	* alert.vars
-rw-rw-r--	lp	bin	* comment
-rw-rw-r--	lp	bin	* configuration
-rw-rw-r--	lp	bin	* forms.allow
-rw-rw-r--	lp	bin	* forms.deny
-rw-rw-r--	lp	bin	* users.allow
-rw-rw-r--	lp	bin	* users.deny
/usr/spool/lp/admins/lp/pwheels:			
drwxrwxr-x	lp	bin	* *printwheel1*
drwxrwxr-x	lp	bin	* *printwheel2*
.			
.			
.			
drwxrwxr-x	lp	bin	* *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`
`-r-xr-xr-x`	`lp`	`bin`	`drain.output`
`-r-xr-xr-x`	`lp`	`bin`	`lp.cat`
`-r-xr-xr-x`	`lp`	`bin`	`lp.page`
`-r-xr-xr-x`	`lp`	`bin`	`lp.set`
`-r-xr-xr-x`	`lp`	`bin`	`lp.tell`
`-r-xr-xr-x`	`lp`	`bin`	`lpsched.jr`
`-r-xr-xr-x`	`lp`	`bin`	`slow.filter`
`/usr/spool/lp/fifos`:			
`p-w--w--w-`	`root`	`root`	`* FIFO`
`drwxrwx--x`	`lp`	`sys`	`private`
`drwxrwx-wx`	`lp`	`sys`	`public`
`/usr/spool/lp/fifos/private`:			
`pr--------`	*user*	*group*	`*` *machIPID*
.			
.			
.			
`/usr/spool/lp/fifos/public`:			
`pr--------`	*user*	*group*	`*` *machIPID*
.			
.			
.			
`/usr/spool/lp/logs`:			
`-rw-rw----`	`lp`	`bin`	`* lpsched`
`-rw-rw----`	`lp`	`bin`	`* requests`
`-rw-rw----`	`lp`	`bin`	`*` *requests1*
`-rw-rw----`	`lp`	`bin`	`*` *requests2*
.			
.			
.			
`-rw-rw----`	`lp`	`bin`	`*` *requestsN*

Permissions	Owner	Group	Directory or File
/usr/spool/lp/model:			
-r-xr-xr-x	lp	bin	1640
-r-xr-xr-x	lp	bin	5310
-r-xr-xr-x	lp	bin	dqp10
-r-xr-xr-x	lp	bin	dumb
-r-xr-xr-x	lp	bin	f450
-r-xr-xr-x	lp	bin	hp
-r-xr-xr-x	lp	bin	lqp40
-r-xr-xr-x	lp	bin	ph.dups
-r-xr-xr-x	lp	bin	pprx
-r-xr-xr-x	lp	bin	prx
-r-xr-xr-x	lp	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
.			
.			
.			
-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 is 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 changes during the life of a print request, alternating between the user who submitted the request and the **lp** ID.

The two 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**, and **lp**. These directories must have the permission flags and ownership shown if 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 users 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 all users, including the submitting user.

These files remain in their directories only as long as the request is in 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 Reference Manual*.)

You can use the **crontab** entry shown below as a model:

```
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 the **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 keeps 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 to be saved more often, or if you want the file to be cleaned out more often than once a day, you can change the time when the **crontab** entry is run by changing the first two numbers. The current values, 13 and 3, cause cleaning up to be done at 3:13 A.M. each day.

The default **crontab** entry supplied is sufficient to keep the old print request records from accumulating in the spooling file system. 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 clean up procedure.

The request log has a simple structure that makes it easy to extract data from it using common operating system shell commands. Requests are listed in the order they are printed, and are separated by lines showing their request IDs. Each line below the separator line is marked with a single letter that identifies the kind of information contained in that 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. It contains 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 words **uid**, **gid**, and **size**, respectively.
C	The number of copies printed.
D	The printer or class destination or the word **any**.

Letter	Content of Line
F	The name of the file printed. This line is repeated for each file printed; files were printed in the order given.
f	The name of the form used.
H	One of three types of special handling: **resume, hold,** and **immediate.** The only useful value found in this line is **immediate.**
N	The type of alert used when the print request was 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 their terminal.
O	The **-o** options.
P	The priority of the print request.
p	The list of pages printed.
r	This single letter line is included 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.
s	The outcome of the request, shown as a combination of individual bits expressed in hexadecimal form. While several bits are used internally by the print service, the most important bits are: **0x0004** Slow filtering finished successfully. **0x0010** Printing finished successfully. **0x0040** The request was canceled. **0x0100** The request failed filtering or printing.

Letter	Content of Line
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 differs 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 has been designed to be flexible enough to handle most printers and printing needs, it does not handle every possible situation. You may buy a printer that does not fit into the way the LP Print Service handles printers, or you may have a printing need that the standard features of the LP Print Service do not accommodate.

You can customize the LP Print Service in a few ways:

- adjust the printer port characteristics
- adjust the Terminfo database
- write an interface program
- write a filter

The diagram in Figure 7-1 gives an overview of the processing of a print request from an AT&T Model 495 printer.

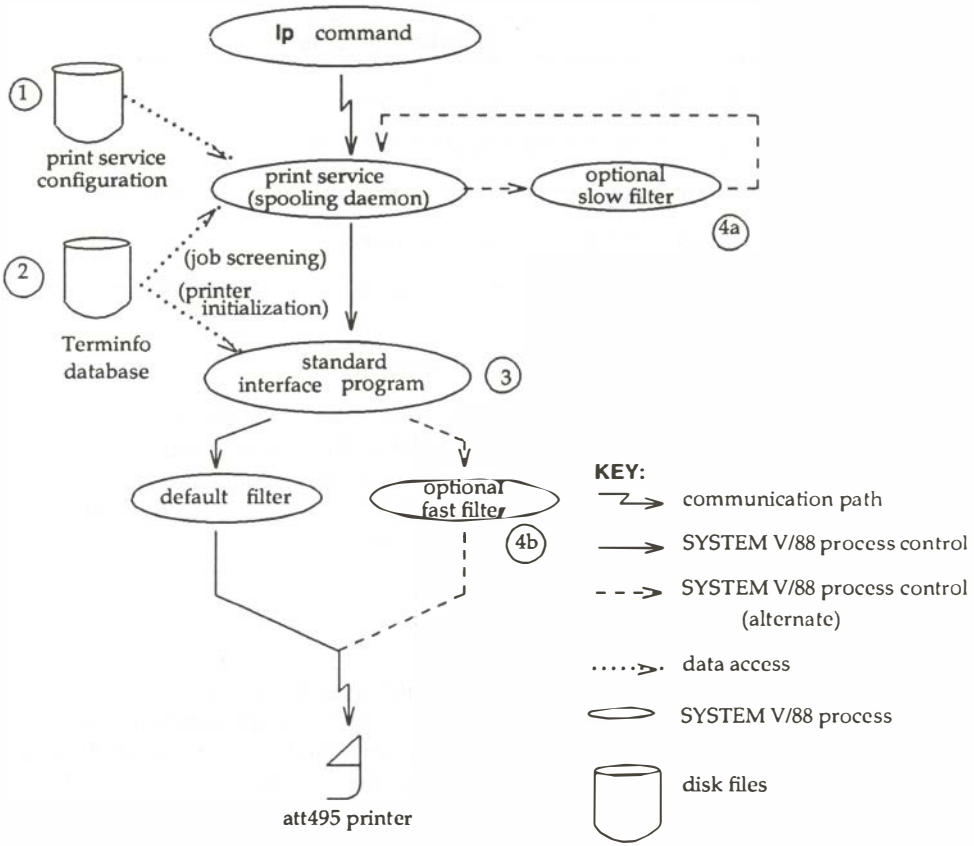


Figure 7-1. How LP processes print request **lp –d att495** *file*

Each print request is sent to a "spooling daemon" that tracks all requests. The daemon is created when you start the LP Print Service. This operating system process is also responsible for keeping track of the status of printers and slow filters; when a printer finishes printing a users file, the daemon starts it printing another request (if there is one queued).

To customize the print service, adjust or replace some of the pieces shown in Figure 7-1 (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 explain how to do this. Configuration data that is relatively dependent on the printer include the printer port characteristics, e.g., baud rate, parity.

2. For a printer that is not represented in the Terminfo database, you can add a new entry that describes its capabilities. The Terminfo 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 a request.

   For instance, if the Terminfo database does not contain an entry for a printer capable of setting a page length requested by a user, the spooling daemon will reject the request. If it does contain an entry for such a printer, 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 a users 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 are designed to work with typical files and many printers, but they do not work with all files and printers. This is not really a customizing step because a standard feature of the LP Print Service is to allow you to specify the port settings for each printer. However, it is an important step in getting your printer to work with the LP Print Service, so it is 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 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 following lists the ones that should be of interest to you (but you should still consult the **stty**(1) manual page for others):

stty Option	Meaning
evenp	Send even parity in the 8th bit
oddp	Send odd parity in the 8th bit
-parity	Do not generate parity; send all 8 bits unchanged
110 - 38400	Set the communications speed to this baud rate
ixon	Enable XON/XOFF (also known as START/STOP or DC1/DC3) flow control
-ixon	Turn off XON/XOFF flow control
-opost	Do not do any "output post-processing"
opost	Do "output post-processing" according to the settings listed below
onlcr	Send a carriage return before every linefeed

stty Option	Meaning
-onlcr	Do not send a carriage return before every linefeed
ocrnl	Change carriage returns into linefeeds
-ocrnl	Do not change carriage returns into linefeeds
-tabs	Change tabs into an equivalent number of spaces
tabs	Do not 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 section *How to Define Printer Ports and Printer Port Characteristics* 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 establish 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 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 in Appendix F in the *User's Guide* describes how to determine a correct TERM variable for a users terminal; you can use it 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 do not, you may still be able to use the printer with the LP Print Service, but you won't have the option of 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 fewer than 50 of these. 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 # 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
***cud1**	Down one line
***cuf1**	Carriage right

Terminfo Item	Meaning
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 SYSTEM V/88. The rest can be added only if you are using SYSTEM V Release 3.2 Version 1 or a later release.

To construct a database entry for a new printer, see details about the structure of the Terminfo database in the **terminfo**(4) manual page (*Programmer's Reference Manual*).

Once you have made the new entry, you need to compile it into the database using the **tic**(1M) program (available in the Terminal Information Utilities). Enter the following command:

tic *file-name*

file-name
is the name of the file containing the Terminfo entry you have crafted for the new printer.

NOTE

The LP Print Service gains 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 used with the LP Spooling Utilities before SYSTEM V/88 Release 3.2 Version 1, it should still work with the LP Print Service. Note, though, that several **-o** options have been "standardized," and are passed to every interface program. These may interfere with similarly named options used by your interface.

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, instead of 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 is responsible for doing the following tasks:

- Initializing the printer port, if necessary. The generic interface program uses the **stty** command to do this.
- Initializing the physical printer. The generic interface program uses the Terminfo database and the TERM shell variable to get the control sequences to do this.
- Printing a banner page, if necessary.
- Printing 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 one is used to connect the printer). The printer port connection is given to the interface program as standard output, and the printer is identified as 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 "uninitialize" the printer in any way.

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 ...*

where:

P
printer name

id
request ID returned by the **lp**(1) command

user
logname of the user who made the request

title
optional title specified by the user

copies
number of copies requested by the user

options
blank-separated list of options specified by the 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 blank-separated list in *options*:

nobanner
used to skip the printing of a banner page; without it, a banner page is printed.

nofilebreak
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*$_2$
specifies a format of *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 *decimal-number*$_1$ and are synonyms, respectively, for **10** columns per inch, **12** columns per inch, and as many columns per inch as possible.

length=*decimal-number*$_1$
width=*decimal-number*$_2$
specifies 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 a set of 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 may be specified by the user when issuing a print request. Alternatively, they may be specified by the LP Print Service from defaults given by the administrator either 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 the following shell variables:

TERM=*printer-type*
specifies the type of printer. The value is used as a key for getting printer capability information from the Terminfo database.

FILTER=*'pipeline'*
specifies the filter to use to send the request content to the printer; the filter is given control of the printer.

CHARSET=*character-set*
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 recognize them and treat them in a consistent manner.

Customizing the Interface Program

Make sure that the custom interface program sets the proper **stty** modes (terminal characteristics such as baud rate and 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 either the LP Print Service or the user with a line like:

stty *mode options* **0<&1**

This command line takes the standard input for the **stty** command from the printer port. The following is an example of an **stty** command line that sets the baud rate at 1200 and sets some of the option modes:

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 this is set varies, 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.
```

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 users title, which may be longer). Look in the code for the standard interface program for the section 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 shows the status of the print job. Exit codes are interpreted by the LP Print Service as follows:

Code	Meaning to LP Print Service
0	The print request has been completed successfully. If a printer fault has occurred, it has been cleared.
1 to 127	A problem has been encountered in printing this particular request (for example, too many non-printable characters, or the request exceeds the printer capabilities). The LP Print Service notifies the person who submitted the request that there was an error in printing it. This problem will not affect future print requests. If a printer fault had 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 has been encountered in printing the request. This problem affects future print requests. If the fault recovery for the printer directs the LP Print Service to wait for the administrator to fix the problem, the LP Print Service disables the printer. If the fault recovery is to continue printing, the LP Print Service does disable the printer, but tries 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 (that does not require the entire request to be reprinted) 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 the printing is finished, the interface program can give a zero exit code just as if the fault had never occurred. An added advantage is that the interface program can detect when the fault is cleared automatically, so that the administrator does not 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.
```

If the special exit code 129 or the **lp.tell** program is used, there is no longer a 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 "kills" the interface program with a signal 15 (see **kill**(1) and **signal**(2) in the *User's Reference Manual* and *Programmer's Reference Manual*, respectively). If the interface program dies from receipt of any other signal, the LP Print Service assumes that future print requests will not be affected, and continues to use the printer. The LP Print Service notifies the person who submitted the request that the request has not been finished successfully.

When the interface is first invoked, the signals SIGHUP, SIGINT, SIGQUIT, and SIGPIPE (trap numbers 1, 2, 3, and 13) are ignored. The standard interface changes this so that these signals are trapped at appropriate times. The standard interface interprets receipt of these signals as warnings that the printer has a problem; when it receives one, it issues a fault alert.

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. 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 users 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 are 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 normally would not be accessible to a regular user; if a filter fails when run directly by a user, it will fail when run by the LP Print Service.

The *Filter Management* section earlier in this chapter describes how to add a filter to the LP Print Service.

If you want your filter to detect printer faults, you must also fulfill the following requirements:

- If possible, the filter should wait for a fault to be cleared before exiting. In addition, it should continue printing at the top of the page where printing stopped after the fault clears. If the administrator does not want this contingency followed, 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. 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 users file is finished (even if errors in the file have prevented it from being printed correctly).
- It should exit with a non-zero exit code *only* if a printer fault has prevented i from finishing a file.
- When added to the filter table, it must be added as a "fast" filter. (See th *Defining a Filter* section under *Filter Management* in this chapter for details.)

8 TTY Management

Introduction

This chapter covers the following topics:

- The terms used in discussing TTY management
- How the TTY system works
- How to tell what line settings are defined
- How to create new line settings and hunt sequences
- How to modify TTY line characteristics
- How to set terminal options

Definition of Terms

The following terms are used in this chapter:

TTY
Derived from the near-classic abbreviation for teletypewriter, the term covers the whole area of access between the operating system and peripheral devices, including the system console. It shows up in commands, e.g., **getty**(1M) and **stty**(1), in the names of device special files, e.g., **/dev/tty01**, and in the names of files, e.g., **/etc/gettydefs** that is used by **getty**.

TTY line
The physical equipment through which access to the computer is made.

port
A synonym for TTY line.

line settings
A set of line characteristics.

baud rate
The speed at which data is transmitted over the line. A part of line settings.

mode
The characteristics of the terminal interface. A part of line settings. The TTY line and the terminal must be working in the same mode before communication can take place. Described in **termio**(7).

hunt sequence
A circular series of line settings such as different baud rates. During the login sequence, a user looking for a compatible connection to the computer can go from one setting to the next by sending a **BREAK** signal.

terminal options
Selectable settings that define the way a given terminal operates. Described in **termio**(7).

TTY System

The remaining sections in this chapter describe how the TTY system operates, and how you can administer it.

How the TTY System Works

A series of four processes (**init**(1M), **getty**(1M), **login**(1), **sh**(1)) connects a user to the operating system. **init** is a general process spawner that is invoked as the last step in the boot procedure. It spawns a **getty** process for each line that a user may log in on, guided by instructions in **/etc/inittab**. An argument required by the **getty** command is **line**. The TTY line argument is the name of a special file in the **/dev** directory. For a description of other arguments that may be used with **getty** see the *System Administrator's Reference Manual*.

A user attempting to make a connection generates a request-to-send signal that is routed by the hardware to the **getty** process for one of the TTY line files in **/dev**. **getty** responds by sending an entry from file **/etc/gettydefs** down the line. The **gettydefs** entry used depends on the **speed** argument used with the **getty** command. (In the SYNOPSIS of the **getty**(1M) command the argument name is **speed**, but it is really a pointer to the **label** field of a **gettydefs** entry.) If no **speed** argument is provided, **getty** uses the first entry in **gettydefs**. Among the fields in the **gettydefs** entry (described later in this chapter) is the login prompt.

On receiving the login prompt, the user enters a login name. **getty** starts **login**, using the login name as an argument. **login** issues the prompt for a password, evaluates the user's response, and assuming the password is acceptable, calls in the user's shell as listed in the **/etc/passwd** entry for the login name. If no shell is named, **/bin/sh** is furnished by default. **login** also executes **/etc/profile**.

/bin/sh executes the user's **.profile,** if it exists. **.profile** often contains **stty** commands that reset terminal options that differ from the defaults. The connection between the user and the operating system has now been made.

How to Tell What Line Settings Are Defined

There are two ways to check line settings:

1. Through the System Administration Menus, specifically the **sysadm**(1) **lineset** subcommand. **sysadm lineset** first shows the full range of line settings, then gives you the chance to examine a line in detail (see Procedure 8.1).
2. By looking directly in **/etc/gettydefs**.

The **/etc/gettydefs** file contains information used by the **getty**(1M) command to establish the speed and terminal settings for a line. The general format of the **gettydefs** file is:

label# initial-flags # final-flags #login-prompt #next-label

Figure 8-1 shows a few lines from a **gettydefs** file.

```
19200# B19200 HUPCL # B19200 SANE IXANY TAB3 HUPCL #login: #9600
9600# B9600 HUPCL # B9600 SANE IXANY TAB3 HUPCL #login: #4800
4800# B4800 HUPCL # B4800 SANE IXANY TAB3 HUPCL #login: #2400
2400# B2400 HUPCL # B2400 SANE IXANY TAB3 HUPCL #login: #1200
1200# B1200 HUPCL # B1200 SANE IXANY TAB3 HUPCL #login: #300
300# B300 HUPCL # B300 SANE IXANY TAB3 HUPCL #login: #19200
```

Figure 8-1. gettydefs Entries

The entries shown in Figure 8-1 form a single, circular hunt sequence; the last field on each line is the label of the next line. The next-label field for the last line shown points back to the first line in the sequence. The object of the hunt sequence is to link a range of line speeds. If you see garbage characters instead of a clear login prompt, entering a **BREAK** causes **getty** to step to the next entry in the sequence. The hunt continues until the baud rate of the line matches the speed of the user's terminal.

The flag fields shown have the following meanings:

B300-B19200
The baud rate of the line.

HUPCL
Hang up on close.

SANE
A composite flag that stands for a set of normal line characteristics.

IXANY
Allow any character to restart output. If this flag is not specified, only DC1 (CTL-Q) restarts output.

TAB3
Send tabs to the terminal as spaces.

For a description of all **getty** flags, see **termio**(7).

How to Create New Line Settings and Hunt Sequences

There are two ways to create new line settings and hunt sequences:

1. Use the System Administration Menus, specifically the **sysadm mklineset**(1) subcommand. **sysadm mklineset** leads you through a series of prompts. Your responses make up the information for a new **gettydefs** entry (see Procedure 8.2).
2. By using **ed**(1) or **vi**(1) to edit **/etc/gettydefs**.

Create new lines for the **gettydefs** file by following the example shown previously. Each entry in the file is followed by a blank line. After editing the file run the command:

```
# /etc/getty –c /etc/gettydefs
```

This causes **getty** to scan the file and print the results on your terminal. If there are any unrecognized modes or improperly constructed entries, they are reported.

How to Modify TTY Line Characteristics

There are two ways to do modify TTY line characteristics:

1. Use the System Administration Menus, specifically the **sysadm modtty**(1) subcommand. **sysadm modtty** leads you through a series of prompts. Your responses edit a "getty" entry in **/etc/inittab** (see Procedure 8.3).
2. By using **ed**(1) or **vi**(1) to edit **/etc/inittab**.

The **/etc/inittab** file contains instructions for the **/etc/init**(1M) command. The general format of a line entry in the **/etc/inittab** file is:

identification:level:action:process

where:

identification
A unique one- or two-character identifier for the line entry.

level
The run-level in which the entry is to be performed.

action
How **/etc/init** treats the process field (refer to the **inittab**(4) manual page for complete information).

process
The shell command to be executed.

/etc/inittab contains several entries that spawn **getty** processes. Figure 8-2 is a selection of such entries **grep**'ped from **/etc/inittab**.

```
co:1234:respawn:/etc/getty console console
ct:2:off:/etc/getty contty contty;#line not in use
21:2:respawn:/etc/getty tty21 9600
22:2:respawn:/etc/getty tty22 9600
23:2:respawn:/etc/getty tty23 9600
24:2:off:/etc/getty tty24 9600;#line not in use
25:2:off:/etc/getty tty25 9600;#line not in use
```

Figure 8-2. getty Entries from **/etc/inittab**

There are at least three things you may want to do to an **inittab** entry for a TTY line:

1. Change the action. Two actions that apply to TTY lines are "respawn" and "off" (see the **inittab**(4) manual page for complete information on this field).
2. Add or change arguments to **/etc/getty** in the process field. A frequently used argument is **–t***nn*. This tells **getty** to hang up if nothing is received within *nn* seconds. It's good practice to use the **–t** argument on dial-up lines.
3. Add or change comments. Comments can be inserted after a semi-colon (;) to end the command, and a pound sign (#) to start the comments.

How to Set Terminal Options

The TTY system described thus far establishes a basic style of communication between the user's terminal and the operating system. Once the user has successfully logged in, there may be terminal options that would be preferable to ones in the default set.

The command that is used to control terminal options is **stty**(1). Many users add an **stty** command to their **.profile** so the options they want are automatically set as part of the **login** process. The following is an example of a simple **stty** command:

```
$ stty cr0 nl0 echoe –tabs erase ^H
```

where:

cr0 nl0
No delay for carriage return or newline. Delays are not used on a video display terminal, but are necessary on some printing terminals to allow time for the mechanical parts of the equipment to move.

echoe
Erases characters as you backspace.

–tabs
Expand tabs to spaces when printing.

erase ^H
Change the character-delete character to a CTRL–H. The default character-delete character is the pound sign (#). Most terminals transmit a ^H when the **backspace** key is pressed. Specifying this option makes the **backspace** key useful.

9 Basic Networking

Introduction

The Basic Networking Utilities (BNU) let computers using SYSTEM V/88 communicate with each other and with remote terminals. These utilities range from those used to copy files between computers (**uucp** and **uuto**) to those used for remote login and command execution (**cu**, **ct**, and **uux**).

As an administrator, you must to be familiar with the administrative tools, logs, and data base files used by the BNU. Procedure 9, *Basic Network Procedures*, presents instructions to install and maintain these utilities; this chapter goes into greater detail about the BNU files, directories, daemons, and commands.

Networking Hardware

Before your computer can communicate with other computers, you must set up the hardware to complete the communications link. The cables and other hardware you will need depend on how you want to connect the computers: direct links, telephone lines, or local area networks.

Direct Links

You can create a direct link to another computer by running cables between serial ports on the two computers. Direct links are useful where two computers communicate regularly and are physically close —within 50 feet of each other. You can use a limited distance modem to increase this distance somewhat. Transfer rates of up to 19200 bits per second (bps) are possible when computers are directly linked.

Telephone Lines

Using an Automatic Call Unit (ACU), your computer can communicate with other computers over standard phone lines. The ACU dials the telephone number requested by the networking utilities. The computer it is trying to contact must have a telephone modem capable of answering incoming calls.

Local Area Network

Local Area Network (LAN) can be the communication medium for basic networking. Once your computer is established as a node on a LAN, it will be able to contact any other computer connected to the LAN. This requires the Network Services Extension (NSE).

Two LAN interfacesessss are possible:

- High speed (1 to 10 Mbit) networks requiring separate control hardware and protocol software. The NSE software must be installed to communicate over these LANs.
- Low speed (300 to 38400 baud) LAN *switches* used for terminal-to-machine and machine-to-machine connections through standard asynchronous TTY ports. A high speed LAN may or may not be used as the backbone of the *switch*.

Networking Commands

Basic networking programs can be divided into two categories: user programs and administrative programs. The following paragraphs describe the programs in each category.

User Programs

The user programs for basic networking are in **/usr/bin**. No special permission is needed to use these programs. These commands are all described in the *User's Reference Manual*.

cu
Connects your computer to a remote computer so you can be logged in on both at the same time, allowing you to transfer files or execute commands on either computer without dropping the initial link.

ct
Connects your computer to a remote terminal so the user of the remote terminal can log in. The user of a remote terminal can call the computer and request that the computer call it back. Here, the computer drops the initial link so that the remote terminal's modem will be available when it is called back.

uucp
Allows a user copy a file from one computer to another. It creates work files and data files, queues the job for transfer, and calls the **uucico** daemon, which in turn attempts to contact the remote computer.

uuto
Copies files from one computer to a public spool directory on another computer (**/usr/spool/uucppublic/receive**). Unlike **uucp**, which lets you copy a file to any accessible directory on the remote computer, **uuto** places the file in an appropriate spool directory and tells the remote user to pick it up with **uupick**.

uupick
Retrieves the files placed under **/usr/spool/uucppublic/receive** when files are transferred to a computer using **uuto**.

uux
Creates the work, data, and execute files needed to execute commands on a remote computer. The work file contains the same information as work files created by **uucp** and **uuto**. The execute files contain the command string to be executed on the remote computer and a list of the data files. The data files are those files required for the command execution.

uustat
Displays the status of requested transfers (**uucp**, **uuto**, or **uux**). It also provides you with a means of controlling queued transfers.

Administrative Programs

Most of the administrative programs are in **/usr/lib/uucp**, along with basic networking data base files and shell scripts. The only exception is **uulog**, which is in **/usr/bin**. These commands are described in the *System Administrator's Reference Manual*.

You should use the **uucp** login ID when you administer the BNU because it owns the basic networking and spooled data files. The home directory of the **uucp** login ID is **/usr/lib/uucp**. (The other basic networking login ID is **nuucp**, used by remote computers to access your computer. Calls from **nuucp** are answered by **uucico**.)

uulog
Displays the contents of a specified computer's log files. Log files are created for each remote computer to which your computer communicates. The log files contain records of each use of **uucp**, **uuto**, and **uux**.

uucleanup
Cleans up the spool directory. It is normally executed from a shell script called **uudemon.cleanup**, which is started by **cron**.

Uutry
Tests call processing capabilities and does a moderate amount of debugging. It invokes the **uucico** daemon to establish a communication link between your computer and the remote computer you specify.

uucheck
Checks for the presence of basic networking directories, programs, and support files. It can also check certain parts of the **Permissions** file for obvious syntactic errors.

Daemons

There are three daemons in the BNU. A daemon is a routine that runs as a background process and performs a system-wide public function. These daemons handle file transfers and command executions. They can also be run manually from the shell.

uucico
Selects the device used for the link, establishes the link to the remote computer, performs the required login sequence and permission checks, transfers data and execute files, logs results, and notifies the user by **mail** of transfer completions. When the local **uucico** daemon calls a remote computer, it "talks" to the **uucico** daemon on the remote computer during the session.

The **uucico** daemon is executed by **uucp**, **uuto**, and **uux** programs, after all the required files have been created, to contact the remote computer. It is also executed by the **uusched** and **Uutry** programs.

uuxqt
Executes remote execution requests. It searches the spool directory for execute files (always named X.*file*) that have been sent from a remote computer. When an X.*file* file is found, **uuxqt** opens it to get the list of data files that are 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** checks the **Permissions** file to verify that it has permission to execute the requested command. The **uuxqt** daemon is executed by the **uudemon.hour** shell script, which is started by **cron**.

uusched
Schedules the queued work in the spool directory. Before starting the **uucico** daemon, **uusched** randomizes the order in which remote computers will be called. **uusched** is executed by a shell script called **uudemon.hour**, which is started by **cron**.

Internal Programs

uugetty
This program is similar to the **getty** program except it permits a line (port) to be used in both directions. A **uugetty** will be assigned to a port in the **/etc/inittab** file if bidirectional is chosen when you modify a port using the **sysadm**(1) **portmgmt** command. **uugetty** is executed as a function of the **init** program and is described in the *System Administrator's Reference Manual*.

Supporting Data Base

The BNU support files are in the **/usr/lib/uucp** directory. Most changes to these files can be made using the System Administration Menu commands described in the Procedure 9, *Basic Network Procedures*. The descriptions below, however, provide details on the structure of these files so you can edit them manually:

Devices
Contains information about the location and line speed of the automatic call unit, direct links, and network devices.

Dialers
Contains character strings required to negotiate with network devices (automatic calling devices) in the establishment of connections to remote computers.

Systems
Contains information needed by the **uucico** daemon and the **cu** program to establish a link to a remote computer. It contains information such as the name of the remote computer, the name of the connecting device associated with the remote computer, when the computer can be reached, telephone number, login ID, and password.

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 that is granted to computers when they attempt to transfer files or remotely execute commands on your computer.

Poll
This file defines computers that are to be polled by your system and when they are polled.

Devconfig
This file is used to configure utilities for the BNU on a transport provider that conforms to the Transport Interface.

Sysfiles
This file is used to assign different or multiple files to be used by **uucico** and **cu** as **Systems**, **Devices**, and **Dialers** files.

There are several other files that may be considered part of the supporting data base, but are not directly related to the process of establishing a link and transferring files. These files —**Maxuuxqts**, **Maxuuscheds**, and **remote.unknown** —are described briefly in Procedure 9, *Basic Network Procedures.*

Devices File

The **Devices** file (**/usr/lib/uucp/Devices**) contains information for all the devices that may be used to establish a link to a remote computer, devices such as automatic call units, direct links, and network connections. Although provisions are made for several types of devices, only Direct Links and the following devices are supported by Motorola:

Codex 5212 ACU Modem
UDS 212 A/D Modem
Codex 2280 Modem
Codex Intelligent Matrix Switch (IMS)

NOTE

This file works closely with the **Dialers, Systems,** and **Dialcodes** files. Before you make changes in any of these files, you should be familiar with them all. A change to an entry in one file may require a change to a related entry in another file.

Each entry in the **Devices** file has the following format:

Type Line Line2 Class Dialer-Token-Pairs

The following defines these fields:

Type
This field may contain one of two keywords (**Direct** or **ACU**), the name of a LAN switch, or a system name.

Direct
This keyword indicates a Direct Link to another computer or a switch (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 your computer or indirectly through a LAN switch.

LAN_Switch
This value can be replaced by the name of a LAN switch. **micom** and **develcon** are the only ones for which there are caller scripts in the **Dialers** file. You can add your own LAN switch entries to the **Dialers** file. If you are adding a Transport Interface-compatible network, such as TCP/IP, you would use the special dialer types **TLI** or **TLIS**.

Sys-Name
This value indicates a direct link to a particular computer. (*Sys-Name* is replaced by the name of the computer.) This naming scheme is used to convey the fact that the line associated with this **Devices** entry is for a particular computer in the **Systems** file.

The keyword used in the *Type* field is matched against the third field of **Systems** file entries:

Devices: `ACU tty11 - 1200 penril`

Systems: `eagle Any ACU 1200 3251 ogin: nuucp \`
`ssword: Oakgrass`

You can designate a protocol to use for a device within this field. See the *Protocols* section at the end of the description of this file.

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/tty11** line, the name entered in this field would be **tty11**.

Line2

If the keyword **ACU** was used in the *Type* field and the ACU is an AT&T 801 type dialer, *Line2* would contain the device name of the 801 dialer. (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 non-801 dialers do not normally use this configuration, the *Line2* field is ignored by them, but it must still contain a hyphen (-) as a placeholder.

Class

If the keyword **ACU** or **Direct** is used in the *Type* field, *Class* may be just the speed of the device. However, it may contain a letter and a speed (e.g., C1200, D1200) 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 another handles the external communications. Here, it becomes necessary to distinguish which line(s) should be used for internal communications and which should be used for external communications.

The keyword used in the *Class* field of the **Devices** file is matched against the fourth field of **Systems** file entries:

Devices: `ACU tty11 - D1200 penril`

Systems: `eagle Any ACU D1200 3251 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** file entry. If this field is **Any** and the **Systems** file *Class* field is **Any**, the speed defaults to 1200 bps.

Dialer-Token-Pairs:
This field contains pairs of dialers and tokens. The *dialer* portion may be the name of an automatic dial modem, a LAN switch, or it may be **direct** for a Direct Link device. You can have any number of Dialer-Token-Pairs. The *token* portion may be supplied immediately following the *dialer* portion or if not present, it will be taken from a related entry in the **Systems** file.

This field has the 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 contains only a *dialer* portion and the *token* portion is retrieved from the *Phone* field of the **Systems** file entry.

A valid entry in the *dialer* portion may be defined in the **Dialers** file or may be one of several special dialer types. These special dialer types are compiled into the software and are therefore available without having entries in the **Dialers** file:

801 - AT&T 801 auto dialer
TLI - Transport Level Interface Network (without STREAMS)
TLIS - Transport Level Interface Network (with STREAMS)

The *Dialer-Token-Pairs* (*DTP*) field may be structured four different ways, depending on the device associated with the entry:

1. If an automatic dialing modem is connected directly to a port on your computer, the *DTP* field of the associated **Devices** file entry only has one pair. This pair would normally be the name of the modem. This name is used to match the particular **Devices** file entry with an entry in the **Dialers** file. Therefore, the *dialer* field must match the first field of a **Dialers** file entry:

   Devices: `ACU tty11 - 1200` **ventel**

   Dialers: ventel `=&-% "" \r\p\r\c $ <K\T%%\r>\c ONLINE!`

   Notice that only the *dialer* portion (**ventel**) is present in the *DTP* field of the **Devices** file entry. This means that the *token* to be passed on to the dialer (in this case the phone number) is taken from the *Phone* field of a **Systems** file entry. (\T is implied, see below.) Backslash sequences are described below.

2. If a direct link is established to a particular computer, the *DTP* field of the associated entry would 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).

3. If a computer with which you wish to communicate is on the same local network switch as your computer, your computer must first access the switch and the switch can make the connection to the other computer. In this type of entry, there is only one pair. The *dialer* portion is used to match a **Dialers** file entry:

   Devices: `develcon tty13 - 1200` **develcon** `\D`

   Dialers: develcon `"" "" \pr\ps\c est:\007 \E\D\e \007`

   As shown, the *token* portion is left blank, which indicates that it is retrieved from the **Systems** file. The **Systems** file entry for this particular computer contains the token in the *Phone* field, which is normally reserved for the phone number of the computer (refer to **Systems** file, *Phone* field). This type of *DTP* contains an escape character (**\D**), which ensures that the contents of the *Phone* field is not interpreted as a valid entry in the **Dialcodes** file.

4. If an automatic dialing modem is connected to a switch, your computer must first access the switch and the switch makes the connection to the automatic dialing modem. This type of entry requires two *dialer-token-pairs*. The *dialer* portion of each pair (fifth and seventh fields of entry) are used to match entries in the **Dialers** file:

   Devices: `ACU tty14 - 1200` **develcon** `vent` **ventel**

   Dialers: develcon `"" "" \pr\ps\c est:\007 \E\D\e \007`
   Dialers: ventel `=&-% "" \r\p\r\c $ <K\T%%\r>\c ONLINE!`

   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 your 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.

There are two escape characters that may appear in 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 does 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).

Protocols

You can define the protocol to use with each device. In most cases it is not needed since you can use the default or define the protocol with the particular System you are calling (see **Systems** file, type field). If you do specify the protocol, you must do in the form *Type,Protocol*. Available protocols are:

g

This protocol is slower and more reliable than **e**. It is good for transmission over noisy telephone lines.

e

This protocol is faster than **g**, but it assumes error-free transmission.

For reliable local area networks, you should use the **e** protocol. Here is an example of adding a protocol designation to a device entry:

```
network,e tcpip - - TLIS \D
```

This says, for device **TCP/IP**, use **e** protocol.

Dialers File

The **Dialers** file (**/usr/lib/uucp/Dialers**) specifies the initial conversation that must take place on a line before it can be made available for transferring data. This conversation is usually a sequence of ASCII strings that is transmitted and expected, and it is often used to dial a phone number using an ASCII dialer (e.g., the Automatic Dial Modem).

As shown in the previous examples, the fifth field in a **Devices** file entry is an index into the **Dialers** file or a special dialer type (801, TLI, or TLIS). Here an attempt is made to match the fifth field in the **Devices** file with the first field of each **Dialers** file entry. In addition, each odd numbered **Devices** field starting with the seventh position is used as an index into the **Dialers** file. If the match succeeds, the **Dialers** entry is interpreted to perform the dialer negotiations. Each entry in the **Dialers** file has the following format:

dialer substitutions expect-send ...

The *dialer* field matches the fifth and additional odd numbered fields in the **Devices** file. The *substitutions* field is 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 dialtone" and "pause."

The remaining *expect-send* fields are character strings. The following are some character strings distributed with the BNU in the **Dialers** file:

```
penril -W-P "" \d > s\p9\c )-W\p\r\ds\p9\c-) y\c : \E\TP > 9\c OK
ventel -&-% "" \r\p\r\c $ <K\T%%\r>\c ONLINE!
hayes  -,-, "" \dAT\r\c OK\r \EATDT\T\r\c CONNECT
rixon  -&-% "" \d\r\r\c $ s9\c )-W\r\ds9\c-) s\c : \T\r\c $ 9\c LINE
vadiac -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
att2212c        -+-,    "" \r\c :--: ato12-y,T\T\r\c rod
att4000 -,-,    "" \033\r\r\c DEM: \033s0401\c \006 \033s0901\c \
         \006 \033s1001\c \006 \033s1102\c \006 \033dT\T\r\c \006
att2224 -+-,    "" \r\c :--: T\T\r\c rod
nls     ""      "" NLPS:000:001:1\N\c
```

The meaning of some of the escape characters (those beginning with "\") used in the **Dialers** file are:

`\p`	pause (approximately ¼ to ½ second)
`\d`	delay (approximately 2 seconds)
`\D`	phone number or token without **Dialcodes** translation
`\T`	phone number or token with **Dialcodes** translation
`\K`	insert a **BREAK**
`\E`	enable echo checking (for slow devices)
`\e`	disable echo checking
`\r`	carriage return
`\c`	no new-line or carriage return
`\n`	send new-line
`\nnn`	send 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 phone 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:

`""`
Wait for nothing (i.e., proceed to the next thing.)

`\d`
Delay for 2 seconds.

`>`
Wait for a >.

`s\p9\c`
Send an **s**, pause for ½ second, send a **9**, send no terminating new-line

`)-W\p\r\ds\p9\c-)`
Wait for a **)**. If it is not received, process the string between the **-** characters as follows. Send a **W**, pause, send a carriage-return, delay, send an **s**, pause, send a **9**, without a new-line, and then wait for the **)**.

`y\c`
Send a **y**.

`:`
Wait for a :.

`\E\TP`
Enable 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, send the phone number. The **\T** means take the phone number passed as an argument and apply the **Dialcodes** translation and the modem function translation specified by field 2 of this entry. Then send a **P**.

`>`
Wait for a >.

`9\c`
Send a **9** without a new-line.

`OK`
Waiting 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 computer. Each entry in the file represents a computer that can be called by your computer. In addition, the basic networking software can be configured to prevent any computer that does not appear in this file from logging in on your computer (refer to the section *Other Networking Files* in this chapter for a description of the **remote.unknown** file). More than one entry may be present for a particular computer. The additional entries represent alternative communication paths that will be tried in sequential order. The management of this file is supported by the System Administration Menu subcommand **systemmgmt**.

Using the **Sysfiles**, you can define several files to be used as "Systems" files. See the description of the **Sysfiles** file for details. Each entry in the **Systems** file has the following format:

System-Name Time Type Class Phone Login

Each of these fields is defined in the following section:

System-name
This field contains the node name of the remote computer.

Time
This field is a string that indicates the day-of-week and time-of-day when the remote computer can be called. The format of the *Time* field is:

daytime[;retry]

The day portion may be a list containing some of the following:

Su Mo Tu We Th Fr Sa
for individual days

Wk
for any week-day (Mo Tu We Th Fr)

Any
for any day

Never
for a passive arrangement with the remote computer. If the *Time* field is **Never**, your computer will never initiate a call to the remote computer. The call must be initiated by the remote computer. In other words, your computer is in a passive mode in respect to the remote computer (see discussion of **Permissions** file).

For example:

```
Wk 1700-0800, Sa, Su
```

This example allows calls from 5:00 p.m. to 8:00 am, Monday through Thursday, and calls any time Saturday and Sunday. The example would be an effective way to call only when phone rates are low, if immediate transfer is not critical.

The *time* portion should be a range of times, e.g., 0800-1230. If no *time* portion is specified, any time of day is assumed to be allowed for the call. 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, *retry*, is available to specify the minimum time (in minutes) before a retry, following a failed attempt. The default wait is 60 minutes. The subfield separator is a semicolon (;). For example, **Any;9** is interpreted as call any time, but wait at least 9 minutes before retrying after a failure occurs.

Type
This field contains the device type that should be used to establish the communication link to the remote computer. The keyword used in this field is matched against the first field of **Devices** file entries:

```
Systems:  eagle Any ACU,g D1200 3251 ogin: nuucp \
            ssword: Oakgrass

Devices:  ACU tty11 - D1200 penril
```

You can define the protocol used to contact the system by adding it on to the *Type* field. The example above shows how to attach the protocol **g** to the device type **ACU**. See the information under the *Protocols* section in the description of the **Devices** file for details.

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) 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** file entry:

```
Systems:  eagle Any ACU D1200 NY3251 ogin: nuucp \
            ssword: Oakgrass

Devices:  ACU tty11 - D1200 penril
```

If information is not required for this field, use a **-** as a place holder for the field.

Phone

This field is used to provide the phone number (token) of the remote computer for automatic dialers (LAN switches). The phone number is made up of an optional alphabetic abbreviation and a numeric part. If an abbreviation is used, it must be one that is listed in the **Dialcodes** file. For example:

```
Systems:  eagle Any ACU D1200 NY3251 ogin: nuucp \
            ssword: Oakgrass

Dialcodes:  NY 9=1212555
```

In this string, an equal 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 four seconds before dialing the next digit.

If your computer is connected to a LAN switch, you may access other computers that are connected to that switch. The **Systems** file entries for these computers will not have a phone 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 computer your computer wishes to communicate with. (This is usually just the system name.) The associated **Devices** file entry should have a **\D** at the end of the entry to ensure that this field is not translated using the **Dialcodes** file.

Login

This field contains login information given as a series of fields and subfields of the 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 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 computer, 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**.

The following is an example of a **Systems** file entry that uses an expect-send string:

```
owl Any ACU 1200 Chicago6013 "" \r ogin:-BREAK-ogin: \
uucpx word: xyzzy
```

This example says send a carriage return and wait for `ogin:` (for **Login:**). If you don't get `ogin`, send a `BREAK`. When you do get **ogin:** send the login name **uucpx**, then when you get **word:** (for Password:), send the password **xyzzy**.

There are several escape characters that cause specific actions when they are a part of a string sent during the login sequence. The following escape characters are useful in UUCP communications:

\N	Send or expect a `NULL` character (ASCII NUL).
\b	Send or expect a backspace character.
\c	If at the end of a string, suppress the newline that is normally sent; otherwise ignored.
\d	Delay two seconds before sending or reading more characters.

`\p`	Pause for approximately ¼ to ½ second.
`\E`	Start echo checking. (From this point on, whenever a character is transmitted, it will wait for the character to be received before doing anything else.)
`\e`	Echo check off.
`\n`	Send a newline character.
`\r`	Send or expect a carriage-return.
`\s`	Send or expect a space character.
`\t`	Send or expect a tab character.
`\\`	Send or expect a \ character.
`EOT`	Send or expect EOT newline twice.
`BREAK`	Send or expect a break character.
`\K`	Same as `BREAK`.
`\ddd`	Collapse the octal digits (ddd) into a single character.

Dialcodes File

The **Dialcodes** file (**/usr/lib/uucp/Dialcodes**) contains the dial-code abbreviations that can be used in the *Phone* field of the **Systems** file. Each entry has the 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** file entry is accessed.

The following entry would be set up to work with a *Phone* field in the **Systems** file such as **jt7867**:

```
jt 9=847-
```

When the entry containing **jt7867** is encountered, the sequence 9=847-7867 would be sent to the dialer if the token in the dialer-token-pair is `\T`.

Permissions File

The **Permissions** file (**/usr/lib/uucp/Permissions**) specifies the permissions that remote computers have with respect to login, file access, and command execution. There are options that restrict the remote computer's ability to request files and its ability to receive files queued by the local site. Another option is available that specifies the commands that a remote site can execute on the local computer.

How Entries are Structured

Each entry is a logical line with physical lines terminated by a \ to indicate continuation. Entries are made up of options delimited by white space. Each option is a name/value pair in the following format:

name=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 newline character. Blank lines are ignored (even within multi-line entries).

There are two types of **Permissions** file entries:

LOGNAME
Specifies the permissions that take effect when a remote computer logs in on (calls) your computer.

MACHINE
Specifies permissions that take effect when your computer logs in on (calls) a remote computer.

LOGNAME entries contain a LOGNAME option; MACHINE entries 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 computers:

- All login IDs used by remote computers to login for UUCP communications must appear in one and only one LOGNAME entry.

- Any site that is called whose name does not appear in a MACHINE entry, will have the following default permissions/restrictions:

  1. Local send and receive requests will be executed.
  2. The remote computer can send files to your computer's **/usr/spool/uucppublic** directory.
  3. The commands sent by the remote computer for execution on your computer must be one of the default commands; usually **rmail**.

Options

This section describes each option, specifies how they are used, and lists their default values.

REQUEST

When a remote computer calls your computer and requests to receive a file, this request can be granted or denied. The REQUEST option specifies whether the remote computer can request to set up file transfers from your computer. The following string specifies that the remote computer can request to transfer files from your computer:

```
REQUEST=yes
```

The following string specifies that the remote computer cannot request to receive files from your computer:

```
REQUEST=no
```

This is the default value. It is 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. A note on security: when a remote machine calls you, unless you have a unique login and password for that machine you do not know if the machine is who it says it is.

SENDFILES

When a remote computer calls your computer and completes its work, it may attempt to take work your computer has queued for it. The SENDFILES option specifies whether your computer can send the work queued for the remote computer.

The following string specifies that your computer may send the work that is queued for the remote computer as long as it logged in as one of the names in the LOGNAME option:

```
SENDFILES=yes
```

This string is mandatory if your computer is in a "passive mode" with respect to the remote computer.

The following string specifies that files queued in your computer are sent only when your computer calls the remote computer:

```
SENDFILES=call
```

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 computers. If the option is used with a MACHINE entry, it is 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 following strings specify permission to access any file that can be accessed by a local user with "other" permissions:

```
READ=/ WRITE=/
```

The value of these entries is a colon separated list of path names. The READ option is for requesting files; the WRITE option for depositing files. One of the values must be the prefix of any full path name 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 would be used with the WRITE option:

```
WRITE=/usr/spool/uucppublic:/usr/news
```

If the READ and WRITE options are used, all path names must be specified because the path names are not added to the default list. For example, if the **/usr/news** path name was the only one specified in a WRITE option, permission to deposit files in the public directory would be denied.

You should be careful what directories you make accessible for reading and writing by remote systems. For example, you probably would not want remote computers to be able to write over your **/etc/passwd** file so **/etc** should not be open to writes.

NOREAD and NOWRITE

The NOREAD and NOWRITE options specify exceptions to the READ and WRITE options or defaults. The following strings 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:

```
READ=/ NOREAD=/etc WRITE=/usr/spool/uucppublic
```

NOWRITE works in the same manner as the NOREAD option. The NOREAD and NOWRITE can be used in 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. There are two examples of when you would use CALLBACK. From a security standpoint, if you call back a machine you can be sure it is the machine it says it is. If you are doing long data transmissions, you can choose the machine that will be billed for the longer call.

The following string specifies that your computer must call the remote computer back before any file transfers will take place:

```
CALLBACK=yes
```

The default for the COMMAND option is:

```
CALLBACK=no
```

The CALLBACK option is very rarely used. Note that if two sites have this option set 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 computer. Files and a command are sent to the target computer for remote execution. The COMMANDS option can be used in MACHINE entries to specify the commands that a remote computer can execute on your computer. Note that COMMANDS is not used in a LOGNAME entry; COMMANDS in MACHINE entries define command permissions whether we call the remote system or it calls us.

The following string indicates the default commands that a remote computer can execute on your computer:

```
COMMANDS=rmail
```

If a command string is used in a MACHINE entry, the default commands are overridden. For instance, the following entry overrides the COMMAND default so that the computers owl, raven, hawk, and dove can now execute **rmail**, **rnews**, and **lp** on your computer:

```
MACHINE=owl:raven:hawk:dove \
COMMANDS=rmail:rnews:lp
```

In addition to the names as specified above, there can be full path names of commands. For example, the following specifies that command **rmail** uses the default path:

```
COMMANDS=rmail:/usr/lbin/rnews:/usr/local/lp
```

The default paths for your computer are **/bin**, **/usr/bin**, and **/usr/lbin**. When the remote computer specifies **rnews** or **/usr/lbin/rnews** for the command to be executed, **/usr/lbin/rnews** is 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 computer(s) specified in the entry is executed. If you use this value, you give the remote computer full access to your computer. Be careful; this allows far more access than normal users have.

The following string illustrates two points: the ALL value can appear anywhere in the string, and the path names specified for **rnews** and **lp** are used (instead of the default) if the requested command does not contain the full path names for **rnews** or **lp**.

```
COMMANDS=/usr/lbin/rnews:ALL:/usr/local/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 in conjunction with the COMMANDS option when specifying commands that are potentially dangerous to your computer's security. It is used to provide a certain degree of verification of the caller's identity. The use of the VALIDATE option requires that privileged computers 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. (VALIDATE is merely an added level of security on top of the COMMANDS option, though it is a more secure way to open command access than ALL.)

Careful consideration should be given to providing a remote computer with a privileged login and password for UUCP transactions. Giving a remote computer a special login and password with file access and remote execution capability is like giving anyone on that computer a normal login and password on your computer. Therefore, if you cannot trust someone on the remote computer, do not provide that computer with a privileged login and password.

The following LOGNAME entry specifies that if one of the remote computers that claims to be eagle, owl, or hawk logs in on your computer, it must have used the login **uucpfriend**:

```
LOGNAME=uucpfriend VALIDATE=eagle:owl:hawk
```

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, which 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 computer is logged in. In fact, it is an asynchronous process with no knowledge of what computer sent the execution request. Therefore, the real question is how does your computer know where the execution files came from?

Each remote computer 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 computer 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 computer name does not appear in the **Permissions** file, the default list is used:

The following example shows the relationship between the MACHINE and LOGNAME entries:

```
MACHINE=eagle:owl:hawk REQUEST=yes \
COMMANDS=rmail:/usr/lbin/rnews \
READ=/  WRITE=/

LOGNAME=uucpz VALIDATE=eagle:owl:hawk \
REQUEST=yes SENDFILES=yes \
READ=/  WRITE=/
```

The value in the COMMANDS option means that remote mail and **/usr/lbin/rnews** can be executed by remote users.

In the first entry, you must make the assumption that when you want to call one of the computers listed, you are really calling either **eagle**, **owl**, or **hawk**. Therefore, any files put into one of the **eagle**, **owl**, or **hawk** spool directories is put there by one of those computers. If a remote computer logs in and says that it is one of these three computers, its execution files will also be put in the privileged spool directory. You therefore have to validate that the computer has the privileged login **uucpz**.

You may want to specify different option values for the computers your computer calls that are not mentioned in specific MACHINE entries. This may occur when there are many computers calling in, and the command set changes from time to time. The name "OTHER" for the computer name is used for this entry:

```
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 computers 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 following two entries share the same REQUEST, READ, and WRITE options:

```
MACHINE=eagle:owl:hawk REQUEST=yes \
  READ=/  WRITE=/

LOGNAME=uucpz REQUEST=yes SENDFILES=yes \
  READ=/  WRITE=/
```

These two entries can be merged:

```
MACHINE=eagle:owl:hawk REQUEST=yes \
LOGNAME=uucpz SENDFILES=yes \
  READ=/  WRITE=/
```

Poll File

The **Poll** file (**/usr/lib/uucp/Poll**) contains information for polling remote computers. Each entry in the **Poll** file contains the name of a remote computer to call, followed by a <TAB> character (a space won't work), and finally the hours the computer should be called. The format of entries in the **Poll** file are:

sys-name hour ...

For example, the following entry provides polling of computer **eagle** every four hours:

```
eagle 0 4 8 12 16 20
```

The **uudemon.poll** script does not actually perform the poll. It sets up a polling work file (always named C.*file*), in the spool directory that will be seen by the scheduler, which is started by **uudemon.hour**.

Devconfig File

The **/usr/lib/uucp/Devconfig** file is used when your computer communicates over a Streams-based transport provider that conforms to the Transport Interface (TI).

Devconfig entries define the STREAMS modules that are used for a particular TI device. Entries in the **Devconfig** file have the format:

service=*x* **device**=*y* **push**=*z*[:*z* ...]

where *x* can be **cu**, **uucico**, or both, separated by a colon; *y* is the name of a TI network and must match an entry in the **Devices** file; and *z* is replaced by the names of STREAMS modules in the order they are to be pushed onto the Stream. Different modules and devices can be defined for **cu** and **uucp** services.

The following entries should most commonly be used in the file:

```
service=cu      device=tcpip  push=tirdwr
service=uucico  device=tcpip  push=tirdwr
```

This example pushes **tirdwr**. The **Devconfig** file cannot be modified with the System Administration Menus command **sysadm**. If you want to change the contents of this file, you must use one of operating system text editors.

Sysfiles File

The **/usr/lib/uucp/Sysfiles** file lets you assign different files to be used by **uucp** and **cu** as **Systems**, **Devices**, and **Dialers** files. Here are some cases where this optional file may be useful:

- You may want different **Systems** files so requests for login services can be made to different addresses than **uucp** services.
- You may want different **Dialers** files to use different handshaking for **cu** and **uucp**.
- You may want to have multiple **Systems**, **Dialers**, and **Devices** files. The **Systems** file in particular may become large, making it more convenient to split it into several smaller files.

The format of the **Sysfiles** file is

service=*w* **systems**=*x*:*x* **dialers**=*y*:*y* **devices**=*z*:*z*

where *w* is replaced by **uucico**, **cu**, or both separated by a colon; *x* is one or more files to be used as the **Systems** file, with each file name separated by a colon and read in the order presented; *y* is one or more files to be used as the **Dialers** file; and *z* is one or more files to be used as the **Devices** file. Each file is assumed to be relative to the **/usr/lib/uucp** directory, unless a full path is given. A backslash-carriage return (\<**CR**>) can be used to continue an entry on to the next line.

The following is an example of using a local **Systems** file in addition to the usual **Systems** file:

```
service=uucico:cu        systems=Systems:Local_Systems
```

If this is in **/usr/lib/uucp/Sysfiles**, then both **uucico** and **cu** first look in **/usr/lib/uucp/Systems**. If the system they are trying to call does not have an entry in that file, or if the entries in the file fail, then they will look in **/usr/lib/uucp/Local_Systems**.

When different **Systems** files are defined for **uucico** and **cu** services, your machine will store two different lists of Systems. You can print the **uucico** list using the **uuname** command or the **cu** list using the **uuname –c** command.

Other Networking Files

There are three other files that impact the use of basic networking facilities. In most cases, the default values are fine and no changes are needed. If you want to change them, however, use any standard operating system text editor (**ed** or **vi**):

Maxuuxqts
This file defines the maximum number of **uuxqt** programs that can run at once.

Maxuuscheds
This file defines the maximum number of **uusched** programs that can run at once.

remote.unknown
This file is a shell script that executes when a machine that is not in any of the **Systems** starts a conversation. It will log the conversation attempt and fail to make a connection. If you change the permissions of this file so it cannot execute (**chmod 000 remote.unknown**), your system will accept any conversation requests.

Administrative Files

The basic networking administrative files are described below. These files are created in spool directories to lock devices, hold temporary data, or keep information about remote transfers or executions.

TM (temporary data file)

These data files are created by Basic Networking processes under the spool directory (i.e., **/usr/spool/uucp/***X*) when a file is received from another computer. The directory *X* has the same name as the remote computer that is sending the file. The names of the temporary data files have the format:

TM.*pid.ddd*

where *pid* is a process-ID and *ddd* is a sequential three digit number starting at 0.

When the entire file is received, the **TM.***pid.ddd* file is moved to the path name specified in the **C.***sysnxxxx* file (discussed below) that caused the transmission. If processing is abnormally terminated, the **TM.***pid.ddd* file may remain in the *X* directory. These files should be automatically removed by **uucleanup**.

LCK (lock file)

Lock files are created in the **/usr/spool/locks** directory for each device in use. Lock files prevent duplicate conversations and multiple attempts to use the same calling device. The names of lock files have the format:

LCK..*str*

where *str* is either a device or computer name. These files may remain in the spool directory if the communications link is unexpectedly dropped (usually on computer crashes). The lock files will be ignored (removed) after the parent process is no longer active. The lock file contains the process ID of the process that created the lock.

C. (work file)

Work files are created in a spool directory when work (file transfers or remote command executions) has been queued for a remote computer. The names of work files have the format:

C.*sysnxxxx*

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 four digit job sequence number assigned by UUCP. Work files contain the following information:

- Full pathname of the file to be sent or requested
- Full pathname of the destination or user/file name
- User login name
- List of options
- Name of associated data file in the spool directory. If the **uucp –c** or **uuto –p** option was specified, a dummy name (**D.0**) is used
- Mode bits of the source file
- Remote users login name to be notified upon completion of the transfer

D. (Data file)

Data files are created when it is specified in the command line to copy the source file to the spool directory. The names of data files have the following format:

"**D.***systmxxxxyyy*"

where *systm* is the first five characters in the name of the remote computer, *xxxx* is a four-digit job sequence number assigned by **uucp**. The four digit job sequence number may be followed by a sub-sequence number, *yyy* that is used when there are several **D.** files created for a work (**C.**) file.

X. (Execute file)

Execute files are created in the spool directory prior to remote command executions. The names of execute files have 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 a four digit sequence number assigned by UUCP. Execute files contain 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 file name to receive standard output from the command execution.

- Command string.
- Option lines for return status requests.

Direct Links

General

This section discusses how direct links are created. Direct links are beneficial only when:

- It is not possible to link the computers together through a LAN.
- Two computers transfer large amounts of data on a regular basis.
- Two computers are located no more than several hundred cable feet apart.

The distance between two directly linked computers is dependent on the environment in which the cable is run. The standard for RS-232 connections is 50 feet or less with transmission rates as high as 19200 bps. As the cable length is increased, noise on the lines may become a problem, which means that the transmission rate must be decreased or limited distance modems be placed on each end of the line.

Do not use more than 1000 cable feet to connect the two computers or communications will be unreliable. This link should operate comfortably at 9600 bps in a clean (noise free) environment.

BNU Software and Direct Links

Ideally, systems that have a direct link should be running the same release of the operating system to have the full set of capabilities available. (Bidirectional ports, which are supported by the **uugetty** program, were introduced with SYSTEM V/68 Release 3.) However, lack of a common version of the operating system will not prevent you from using the BNU.

This section describes the software files that must be modified on your computer to accommodate a direct link connection. You may want to consult the documentation provided with various computers concerning linking directly to a remote 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.**

All necessary additions/modifications to these three files can be accomplished using the System Administration Menus subcommand **uucpmgmt.**

Making Devices File Entries

The **Devices** file contains the information concerning the location (line) and transmission rate of the link. Entries can be added to the **Devices** file using the System Administration Menus subcommand **uucpmgmt devicemgmt** and selecting the **add** operation. You will be prompted for the following information:

- port name (for /dev/tty21 use **tty21**)
- device type to call on (**direct**)
- speed at which you want to call (**9600** or **19200**).

To access the System Administration Menus subcommand **devicemgmt**, enter:

 # sysadm devicemgmt<CR>

You should see the following output on your screen:

```
Running subcommand 'devicemgmt' from menu 'uucpmgmt',
BASIC NETWORKING UTILITIES MANAGEMENT
```

NOTE

After a brief introduction to the **devicemgmt** subcommand, the procedure interactively prompts you for the information listed above.

Making Changes to the /etc/inittab File

There are two versions of the BNU. The differences between them are reflected in the **/etc/inittab** file. The newest version allows for bidirectional login capability by respawning **uugetty** instead of **getty**. This means that if two computers (both using **uugetty**) were connected via a direct link, either of these computers could request communication with the other. This would not be true if only one computer was capable of respawning **uugetty**.

If the direct link is connecting your computer with a computer that has the new version of basic networking, the **/etc/inittab** files on both computers should be set up to allow "bidirectional" traffic on the associated lines. This means that the lines used must respawn **uugetty** on each end of the link. This would allow either computer to request communication with (call) the other.

If the direct link is connecting your computer with a computer that does not have the new version of basic networking, the **/etc/inittab** file would be set up differently on each system. The **/etc/inittab** file on each computer would be set up to allow either "incoming" or "outgoing" traffic on its line. If one computer allows incoming traffic, the other must allow only outgoing traffic. A **uugetty** could not be used on either computer in this case.

A computer's **/etc/inittab** entry would be respawning **getty** for incoming traffic, or have respawn turned off for outgoing traffic. In order for this type of link to work, one of the computers must be set up to poll the other. If the remote computer is allowing only incoming traffic, you must set up your computer to poll the remote computer. (You can use the System Administration Menus subcommand **uucpmgmt pollmgmt** to do this). If the remote computer is allowing only outgoing traffic on the link, it must be set up to poll your computer.

Entries in the **/etc/inittab** file can be changed using the System Administration Menus subcommand **uucpmgmt portmgmt**, and selecting the **modify** operation. The **modify** operation will prompt you for the following information:

- port name you want to modify (for /dev/tty21 use **tty21**)
- direction of traffic on port (**bidirectional**, **incoming**, or **outgoing**)
- transmission speed of the link (**9600** or **19200**).

The modify procedure displays the ports that are currently dedicated for use by UUCP (listed in **Devices** file). The port name to be modified must be one that is listed.

To access the **portmgmt** subcommand and modify entries in the **/etc/inittab** file, enter:

 # sysadm portmgmt<CR>

You should see the following output on your terminal screen:

```
Running subcommand 'portmgmt' from menu 'uucpmgmt',
BASIC NETWORKING UTILITIES MANAGEMENT
```

NOTE

After a brief introduction to the **portmgmt** subcommand, the procedure interactively prompts you for the information listed above.

Making Systems File Entries

An entry must be made into the **Systems** file for the computer associated with the direct link. This can be done using the System Administration Menus subcommand **systemmgmt** and selecting the add operation. The add operation will prompt you for the following information:

- node name of system
- type of device to call on (**direct**)
- transmission speed of link (**9600** or **19200**)
- device port used with link (for /dev/tty21 use **tty21**)
- login ID used to login on system (**nuucp** or some other login you set up)
- password used by above login

To prevent possible problems logging on when operating at high speeds, you can insert pauses (**\p**) between the characters being sent out for the login ID and the password. For instance, instead of **nuucp**, you should enter **n\pu\pu\pc\p\pp** when prompted for the login ID. Do *not* select the default by pressing **RETURN**. The same applies for the password assigned.

To access the **systemmgmt** subcommand and add entries in the **Systems** file, enter:

 # sysadm systemmgmt<CR>

You should see the following output on your terminal screen:

```
Running subcommand 'systemmgmt' from menu 'uucpmgmt',
BASIC NETWORKING UTILITIES MANAGEMENT
```

NOTE

After a brief introduction to the **systemmgmt** subcommand, the procedure interactively prompts you for the information listed above.

Upon completion of the **add** operation, a new entry is added to the **Systems** file for the remote computer. You must also make an additional entry in the **Devices** file. When you use **devicemgmt** to create an entry for the link, it creates an entry similar to:

```
Direct tty21 - 9600 direct
```

10 Remote File Sharing

Overview

Remote File Sharing (RFS) allows computers running SYSTEM V/88 to selectively share resources (directories containing files, subdirectories, devices, and/or named pipes) across a network. As an administrator of a 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.

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 looks to your users as though it is part of the local system. You can use most standard operating 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

Because you can mount a remote resource on any directory on your system, you can set up your hosts view of the world. You do not have to open up all your files to every host 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 host 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 a 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.)

Resource Sharing

Sharing a resource on an RFS system begins with a pathname to an operating system directory. If there is a directory you want to share, you assign it a resource identifier and "advertise" it to other hosts, using the **adv**(1M) command. The resource identifier is how other machines reference that directory. Hosts 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 10-1 shows how two hosts can share resources. In this example, the administrator of a host 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.

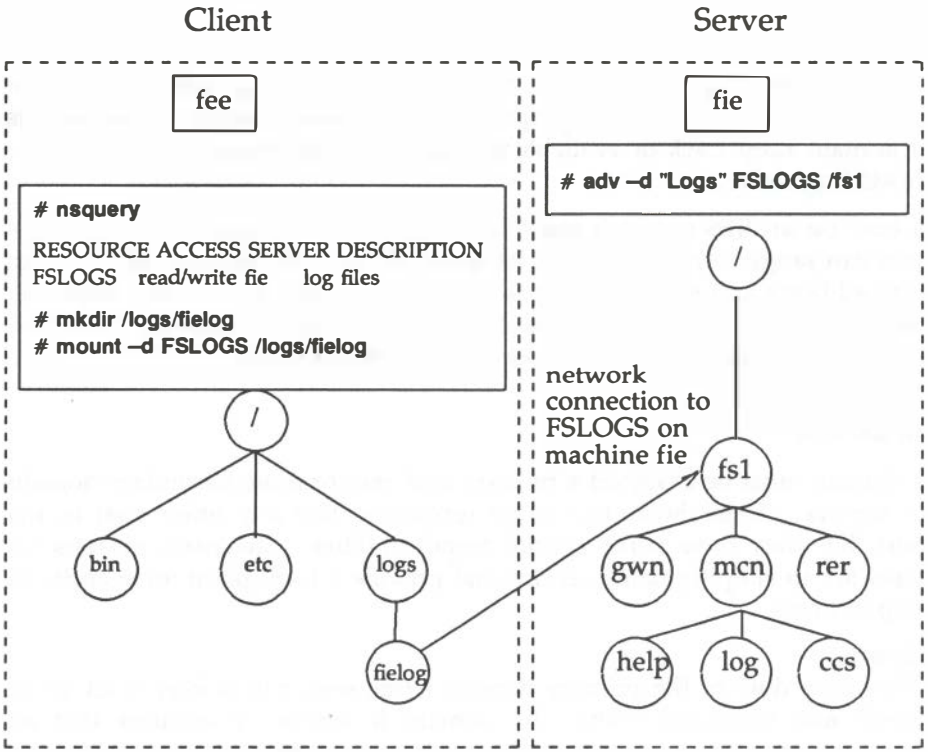


Figure 10-1. Example — Sharing Resources

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 **cd** 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 host.

Domains

A "domain" in RFS provides a means of logically grouping hosts for administrative and addressing purposes. Designated domain name servers for a domain keep track of available resources and password information for the hosts they serve.

Each host on an RFS network must be assigned to a domain. The size of a domain can range from one host to as many that fit on the network. You can address all hosts and resources in your domain directly. For outside domains, you 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 servers. These hosts can share resources, like any other host in the domain, but they have some special responsibilities. The main reasons for domains are to simplify name service and provide a focal point for security of a group of hosts.

Primary

The main duty of the primary domain name server is to keep track of all hosts 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 host to the Domain Member List and assign its RFS password.

A list of advertised resources are automatically stored on the primary, so any host 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. When a host tries to mount another host's resource, 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 host's users (**/etc/passwd**) and groups (**/etc/group**). Each host in the domain can then use these lists to specifically define the permissions each host 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, hosts in its domain have the potential to access resources from any host 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 has 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.

Administration

RFS administration should be viewed on two levels —host and domain. As a host administrator you need to do the following:

1. Start up RFS. This connects you to the RFS service on the network.
2. Choose what resources you want to share and who you want to share them with. When you advertise resources, they are listed with your domain name server.
3. Choose the resources you want to use from other hosts and mount them on your local file system.
4. Optional: Define how group IDs and user IDs from remote hosts are mapped into your host. You can also provide lists of your users and groups so they can be mapped into other hosts.

All domain administration is done from the primary domain name server. These activities include:

- defining the names and network addresses of the primary and secondary domain name servers, including those of name servers for any other accessible domains
- maintaining a list of all hosts in the domain

- (optional) maintaining password and user mapping information for all accessible hosts within the domain

If the primary goes down, a secondary takes over as domain name server. However, an administrator on a secondary cannot add or remove hosts or change host passwords. You can only pass name server responsibility back to the primary when it comes up.

A primary name server can serve several domains. However, the secondaries must be the same for all domains the primary serves. This simplifies the process of passing name server responsibilities back to the primary after it has gone down and come back up.

The tools you use for RFS extend, instead of change, the standard operating system interface. You can use the same commands and system calls on remote resources as you did on local resources. However, some new options and commands have been added to address special RFS needs.

For example, the **sar –u** command still produces system activity reports for your computer, but **sar –Du** adds RFS records to the report. Likewise, your operating system file structure looks the same, but it is extended by mounting remote resources (**mount –d** command).

Transport Provider

The transport provider provides the pathway used by RFS to communicate with other hosts. The term transport provider is used to refer to the physical network that connects the hosts and the software needed to send messages across the network.

RFS can communicate using any transport provider that is compatible with the criteria in Appendix B of the *NSE Programmer's Guide, Part 1*.

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: the *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 requests a particular service code. The service code tells 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 *SYSTEM V/88 SRG*, the RFS service code should be automatically configured for the listener of every transport provider you installed. Otherwise, you need to use the **nlsadmin**(1M) 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 host, you must tell RFS which transport provider will handle RFS on your host. The network specification is the name you use when you initially configure RFS to indicate its transport provider.

Network Addresses

When RFS is started on a host, the host tries to contact its domain's primary name server. To do that, the host must know the primary's network address. The form of the network address varies according to the transport provider used.

Security

As a host administrator you can maintain strict control of your resources. No files, directories, or devices in an unshared file system can be accessed by other hosts. Standard operating 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 hosts to access your resources.

The major mechanisms in RFS for protecting your resources are described in the sections *Verify Computers*, *Restrict Resources*, and *Map IDs*.

Verify Computers

When a remote host tries to mount a resource from your host, 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 host 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 host is the one it claims to be by checking its RFS password. The following 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 hosts can connect

  If you use the RFS verification feature, you can make sure that only specific hosts can use any of your resources. Those hosts 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 host has established a connection to your host, the resources it can mount from your host depend on how you advertised each resource. Your choices are:

- any host can mount

You may have advertised the resource so that any host that can connect to your host can mount it.

- some host can mount

  You restricted access to the resource to certain hosts. The remote host trying to mount it must be one of those hosts.

You also may have advertised the resource as read-only. In that case, the remote host 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 host'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-host basis. The global rules set user and group permissions for all remote hosts that do not have explicit mapping rules.

Here are the ways you can map remote host users into your machine; these rules apply to both global and host mapping:

- no mapping

  If you do not set any special mapping for any remote computer, all users are mapped into your host as a "special guest" user ID/group ID. This is the easiest approach because you do not need to keep any records for the remote machine, create rules files or run the **idload**(1M) 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 (e.g., **exclude 0**).

- specific mapping

  You can map any user or group from any remote host into a specific user or group on your machine. This can be done 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.)

Setting Up RFS

In most cases, you do not need the set of tasks described in this section because the basic RFS configuration and reconfiguration can be handled using the **sysadm setuprfs** command, as described in Procedure 10.1. 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: SYSTEM V/88 software, RFS Utilities, Networking Support Utilities, and transport provider software. (See the *SYSTEM V/88 Software Release Guide* and the transport provider manuals that accompany the product for installation instructions.) You must also log in as **root**.

Set Node Name

> **CAUTION**
>
> **Changing the node name of your computer requires careful coordination with all machines that communicate with yours using RFS or other communications packages that rely on node name.**

Check to see if your computers node name is set to the name you want (**uname –n**). If it's not, set it by typing:

sysadm nodename

You will be asked to type in your computers node name. A node name that is valid for RFS can consist of up to eight characters of letters (upper- and lowercase), digits, hyphens (-), and underscores (_). Some networks 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 and RFS in the order described in the *SYSTEM V/88 Software Release Guide*, 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 listener is improperly set up, this task shows how to manually set up the listener. To set up the listener for other networks compatible with the Transport Interface, you would replace **network** with the name of the network (network specification) you are installing. (For more details, see the **nlsadmin**(1M) manual page.)

To check if the listener is properly installed and set up for use by RFS, type the following:

nlsadmin –v network

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 does not harm your listener configuration. Type the following to initialize the files needed for the listener process for the network specified, in this case **network**:

nlsadmin –i network

Next type the following to add the RFS service (**rfsetup**) to the list of services available to the **network** listener:

nlsadmin –a 105 –c /usr/net/servers/rfs/rfsetup –y "rfsetup" -p timod network

Use the following command line to report the status of the **network** listener process installed on this machine (ACTIVE or INACTIVE):

nlsadmin –x

Next type the following to register the network addresses of your machine:

nlsadmin –l "net_address" –t "*nodename*" **network**

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. (Other networks will use other types of network addresses. See the description of **rfmaster**(4) in the *System Administrator's Reference Manual* for the syntax of different address types.)

To start the listener, type:

nlsadmin –s network

Normally, it is 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 your host will be a member of. The domain name must:

- contain no more than 14 characters
- consist of any combination of letters (upper- 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 the network (transport provider) it should use:

dname –N network

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 is automatically 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 defines:

- 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:

- 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 host **peanuts.charlie** the network address is **abc**.
- For host **peanuts.linus** the network address is **def**.
- For host **peanuts.lucy** the network address is **ghi**.

```
peanuts          p          peanuts.charlie
peanuts          s          peanuts.linus
peanuts          s          peanuts.lucy
peanuts.charlie  a          abc
peanuts.linus    a          def
peanuts.lucy     a          ghi
```

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. The following 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.

Some special considerations when creating the file:

- Fields in each entry must be separated by one blank or one tab.
- The address must be in ASCII text or hexadecimal notation. For hexadecimal, the field must begin with **\x** and contain an even number of digits. If the address contains tabs or spaces, the field must be surrounded by double quotes (").
- An entry can extend beyond one line if you enter a back slash, 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 is 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 are prompted for an initial password, which is 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**(1) 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 (Optional)

NOTE

This procedure assumes you are starting RFS from the shell using **rfstart** with the **v** option or **init 3** either manually or automatically at boot time to start RFS (**sysadm setauto**).

When you start RFS, you can indicate that all remote machine passwords be verified when they try to use your computer's resources. **rfstart**(1M) 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 does not match, or if no *domain*/**passwd** file exists, the remote mount fails. (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 occurs. If a *domain*/**passwd** exists for the remote computer's domain on your host and the remote computer is listed, but the password does not match, a mount request fails. If the computer is not listed in the file or if the *domain*/**passwd** file does not exist, the computer is allowed to mount your resources without validation. (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 contains a file called **passwd**. (*domain* is replaced by the domain name.) This file has 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 host. 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 72 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 from 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 (Optional)

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 *System Administrator's Reference Manual* for 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           abc
docs.little     a           def
```

Step 3: Stop RFS on the primary (**rfstop**, **init 2**, or **sysadm stoprfs**).

Step 4: Restart RFS on the primary (**rfstart**, **init 3**, or **sysadm startrfs**). 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 prepend the domain name to the resource identifier. For example, the following command could be used to mount a resource called INFO that is advertised in domain **docs** with read/write permissions:

mount –d docs.INFO /usr/info

Multiple Domain Name Service (Optional)

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           abc
docs.little     a           def

peanuts         p           docs.big
peanuts         s           docs.little
```

Step 2: Stop and restart RFS. You must stop all machines served by the primary (**rfstop, init 2,** or **sysadm stoprfs**). You must then restart the primary (**rfstart, init 3,** or **sysadm startrfs**). 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 ensures that the new information is picked up by each machine.

Complex User ID/Group ID Mapping (Optional)

ID mapping lets you control the access remote users 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" never overlaps 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 using this or the **sysadm** procedures, all users are 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 are owned by 60001 or 60002. 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, remote users can only access the file if the "other" permissions are set (the last three 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. That way you would be able to modify any remote resources mounted on any machine from which you are working.

Transparent mapping
If you set up a group of computers to have the exact 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 to 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.

NOTE

The default mapping and transparent mapping can be set up using the **sysadm** interface described in Chapter P10, *Remote File Sharing Procedures*. For other mapping techniques, use the following procedure.

Mapping Tools and Files

The result of this procedure is "mapping translation tables." These tables are 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 10-2 illustrates the components described in the previous paragraphs.

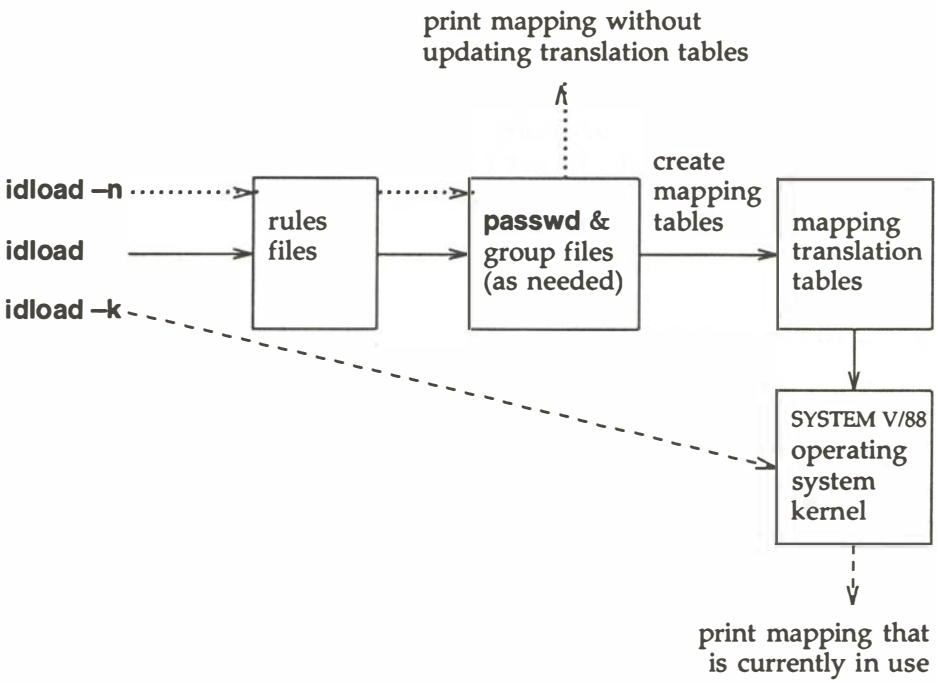


Figure 10-2. ID Mapping Components

The files that are involved in setting up ID mapping are illustrated in Figure 10-3.

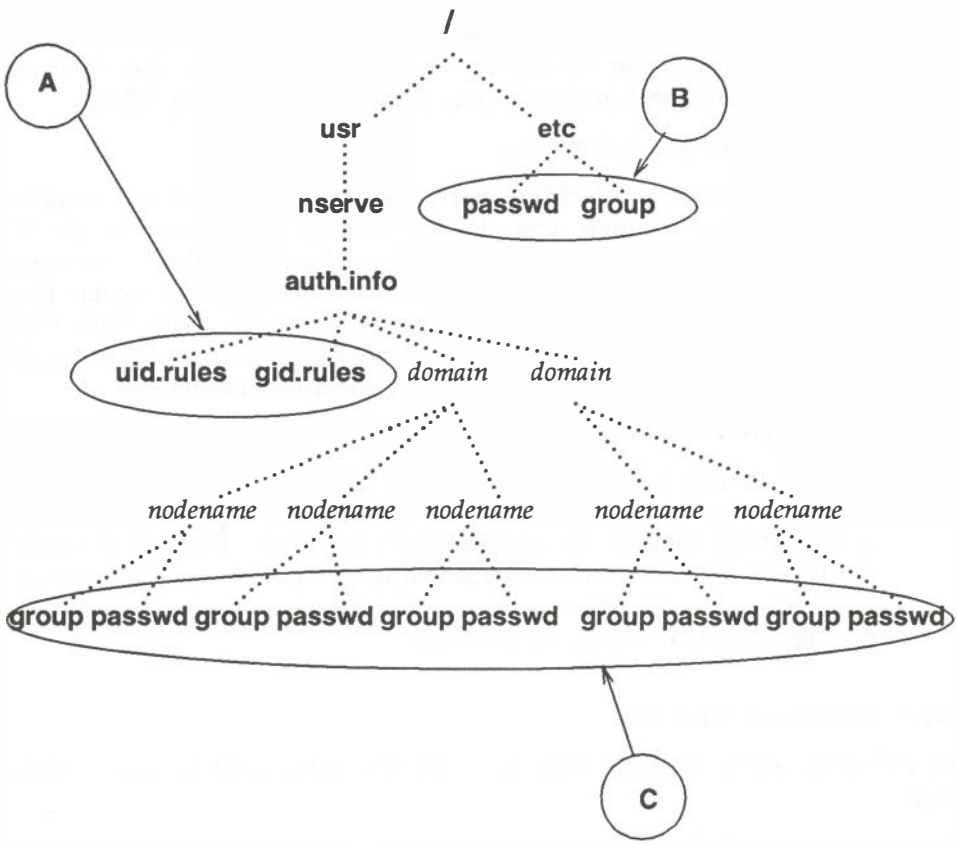


Figure 10-3. ID Mapping Files

The files used for ID mapping are divided into the following three groups, as shown in Figure 10-3.

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 do not 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 contain 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. (Only add this line 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. (Only add this line 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 follow:

default transparent
or
default *local*

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.)

Step 1C: Add **exclude** line(s). (Only add this line(s) 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 ensure that the **root** user does not have permission to modify the restricted files owned by **root** in your resources. The two types of exclude lines follow:

exclude *remoteid*
or
exclude *remoteid–remoteid*

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.

Step 1D: Add **map** line(s). (Only add this line 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 follow:

map *remoteid*:*local*
or
map *remoteid*

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.

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. (Only add this line 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. (Only add this line 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 follow:

default transparent
or
default *local*

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.

Step 1G: Add **exclude** line(s). (Only add this line(s) 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 follow:

exclude *remote*
or
exclude *remoteid–remoteid*

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.

Step 1H: Add **map** line(s). (Only add this line(s) 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 follow:

map all
or
map *remote*:*local*
or
map *remote*

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.

Repeat steps 1E-1H for each specific computer whose users you want to map. The **uid.rules** file is now complete.

Figure 10-4 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 10-4. 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 **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. Though **idload** will still work without the **gid.rules** file (**idload** uses 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 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 prints a listing of the mapping rules you set up, without creating translation tables. Figure 10-5 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 10-5. 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 take effect as soon as that computer mounts one of your resources.

Step 4C: Run **idload –k**. This prints 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.) ID mapping is now complete.

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 does not 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 either **sysadm isrfson** or **rfadmin –q**. These tell you that RFS is or is not 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, if you are not using the **sysadm** interface, you should use the following command (a primary can start RFS initially by using **init 3**):

 # rfstart –p *primary_ns_address*
rfstart: Please enter machine password:

The *primary_ns_address* is replaced by the network address of the primary name server for your domain.

RFS Password

You are 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 saves this password automatically so you do not have to enter it again.

Likewise, your machine saves the network address of the primary name server. Therefore, the next time you start up RFS, you are 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 receive a warning, but **rfstart** does not fail.

Though RFS is 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 fail if the passwords do not 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 are not 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 the 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 reentering it with the **rfpasswd** command. If that does not work, do the following sequence shown, 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:

```
# sysadm stoprfs
# rm /usr/nserve/loc.passwd
# sysadm startrfs
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 changes 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 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.

> CAUTION
>
> **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 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 are 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.

See the *Operating Levels* section of Chapter 3, *Processor Operations*, for details on the run levels. This section describes 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. See Procedure 10.1 for information on setting up 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**s, **cron**, etc.) are 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 RFS mode processes. (**init 3** does not spawn another process for a level 2 script that is already running.)

3. At boot time

   By default, your system enters 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:

   ```
   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 is tried continuously until it does via **/etc/rmount**(1M). (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 executes **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**(1M) and **fumount**(1M) for each advertised resource, **umount** for each mounted remote resource, and **rfstop**.

Changing init 3 Processing

Going into **init** state **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** keeps 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.

You should name the files using the following conventions:

S*00name*
or
K*00name*

The filenames 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 are started (S00, S01, S02, etc.) or stopped (K00, K01, K02, etc.).

name
The rest of the filename is the **/etc/init.d** filename this file is linked to.

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 S68netdaemon start**. When you enter **init 0**, this file is executed with the **stop** option: **sh K67netdaemon 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 simply 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 happen automatically when you leave **init** state 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 can start RFS. Computers that are already running RFS continue to run RFS; however, they are not to mount or advertise new resources.

Sharing Resources

This section describes how to share your local resources with other computers (advertising) on a 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 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**(1M) command). A remote computer does not know the exact location of the resource on your machine. All the remote computer knows 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 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 either:

adv –m *resource* **–d** "*description*" [*clients ...*]

or

adv –m *resource* [**–d** "*description*"] *clients ...*

The options are:

–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 zero to 32 characters and should be in quotes.

resource

This is the resource name you assign. The name is limited to 14 printable ASCII characters; slash (/), period (.), and spaces and tabs may not be used. (If you enter more than 14 characters, the name is 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 is 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* below.)

–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 3re 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** are 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 back slash, then a carriage return to continue to the next line.

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 only selected remote computers can mount it (see the *client* option of the **adv** command).

Map IDs

The permissions remote users 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 apply to any of your resources that computer mounts.

SYSTEM V/88 security

Normal SYSTEM V/88 access security, governing read, write, and execute permissions, 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 is 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   "SYSTEM V/88 calendar" peanuts. compgrp
```

The information matches 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
To list the resources advertised by a particular computer in your domain.

domain.
To list 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
To list all resources advertised by a particular computer in a domain outside your own domain.

If the *name* option is not used, **nsquery** prints 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** lists 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**(1M) 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:

rmntstat [**–h**] [*resource*]

where **–h** prints 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**(1M) command. It removes 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 do 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 *Automatic Remote Mounts* section of this chapter for information on how to modify the **/etc/rstab** file.)

The syntax of the 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 and a period (.).

Forced Unmount

You cannot unmount a local file system using the **umount**(1M) 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**(1M) 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 writes 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** tries to remount the resource every 60 seconds until it succeeds.

See **rfuadmin**(1M) and **rmount**(1M) for further information on the processing of these commands.

Remote Resource Mounting

You can attach another computer's advertised resource to your system using the standard **mount**(1M) command. You simply choose an existing directory, preferably empty, 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 is 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:

mount [**–r**] [**–c**] **–d** *resource directory*

The options are:

–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.)

–d
This option is used to indicate that you are mounting a remote resource.

resource
This must be replaced by the resource identifier assigned by the computer that advertised the resource.

directory
This must be replaced by the full path to the local directory on which you want to mount the resource.

–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 generally want the default (buffer caching on), since caching cuts 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.)

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:

resource directory **–d**[**r**]

resource is replaced by the resource name, *directory* is replaced by the directory where to mount the resource, 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

The following 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 becomes 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 does not become part of the resource, however. Only your local users can access it. (Remote users can see this mount point directory, but 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 results in essential local files being inaccessible to your system.

For example, you should not mount a remote **/dev** on your machine's **/dev** directory or you 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. In this case, the problem is spool files and workspace directories. Applications such as **uucp**(1C) and **lp**(1) 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 gobble up 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 (**fumount**(1M)), 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. (See Chapter 6 for a discussion of local sticky bit use.)

Local Mount Table

You can list the remote resources that are mounted on your computer as you would list local file systems: the **mount** command with no options. Remote resource output from this command appears in the 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/c1d0s0 read/write on Thu Jan 16 09:07:19 1986
/usr2 on /dev/dsk/c1d0s8 read/write on Thu Jan 16 09:07:32 1986
/usr on /dev/dsk/c1d1s2 read/write on Thu Jan 16 09:07:33 1986
/s/codes on LCODE  read/write/remote on Thu Jan 16 09:10:13 1986
/s/timing on TEMPO read only/remote on Thu Jan 16 09:10:27 1986
$
```

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 that server are 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, **rfudaemon** 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** sends a 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** sends a **fuwarn** message, the resource name, and the number of seconds before the resource is 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**(1) command:

resource **has been disconnected from the system.**

Then it executes **fuser**(1M) to kill all processes using the resource, unmounts the resource (**umount**(1M)) 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**(1M) command. The syntax for using **umount** to unmount a remote resource from your computer is:

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 **–u** lists user names in the report of processes that have files open in any directory or subdirectory relating to the resource, and **–k** kills all processes that have files open in any directory or subdirectory relating to the resource.

Sharing Printers

One of the common uses of RFS is for sharing printers. The concepts presented here may also apply to sharing other peripherals that use spool files.

On the server machine:

1. Set up **lp** on the server the way you normally would on any machine. (The *server* machine is the one that does all the spooling). Make sure that the printer works and you are able to print text on this machine.
2. Advertise **/usr/spool/lp** to all the *client(s)* that will be using this printer.
3. In **/usr/nserve/uid.rules**, map the user id of **lp** to itself. 71 is usually the uid of **lp**, so the entry in **uid.rules** would be: **map 71:71**.

On the client machines:

1. Do not run the scheduler on the client machines. All you need on the client machines are the **lp** and the **lpstat** commands.
2. Mount the resource that was advertised by the server on the client's **/usr/spool/lp**.

The **–c** option of **lp** should be used for any user file that is not a shared resource. The **lp** commands on the client machines generate the file names for the spooler utilizing the PID. For RFS versions prior to Release 1.2, identical spool file names might be generated by multiple clients. If this happens simultaneously, there might be a collision in the spooling directory. The consequence of this is that one of the files will get lost, but the chances of this happening are very small.

Mapping Remote Users

NOTE

The *Complex User ID/Group ID Mapping* procedure in this chapter 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 has 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 does not map into permissions of any existing users and groups on your system.

If you don't 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 are ever 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 translates the request from **abc**'s **101** into a request from **115**. If local ID **115** has permissions to create a file in that directory, the file is 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. However, it could potentially cause confusion. 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: **101** user ID.

NOTE

This same result occurs 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**(1M) 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 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 10-6 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 10-6. Format of **uid.rules** and **gid.rules** Files

The following notation is used in the previous figure:

local_name
is a local user name

local_id
is a local user ID number

remote_id
is a remote user ID number

remote_name
is a remote user name

local
is a local name or ID number

remote
is 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 is 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 is 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** reads 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 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 has the permissions of local uid **101**.

   local is replaced by a local user name or ID number. By default, all remote users are 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 are 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 still has access to your resources but only has 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*; separate 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, the following gives the remote **mcn** the same permissions of the local user **mcn**:

   map 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** usually produces warning messages, since multiple administrative logins have uid 0, and **idload** tries to map each one 0 to 0. There is no harm in this.

idload Command

Once the rules files are created, use the **idload**(1M) 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 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:

–n
is the "no update mode" option. When it is used, **idload –n** prints the mapping that would result from the rules files without putting them into effect.

–k
shows the mapping that is currently active on your machine. (Note that there is 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*
lets you use a group rules file other than **/usr/nserve/auth.info/gid.rules** as input for group mapping rules.

–u *u_rules*
lets you use a user rules file other than **/usr/nserve/auth.info/uid.rules** as input for user mapping rules.

directory
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** is assumed.

Each time you set up or change your rules files, first run **idload** with the **–n** option. The results show you the mapping that occurs 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 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 checks the directory for that computer for **passwd** and **group** files. The pathname to the remote computer's directory is **/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 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 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 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 10-7, all remote user IDs are mapped into the same user ID permissions on your computer, except for **root** (ID number **0**), which would only have special guest permissions. This applies to all remote computers.

> **CAUTION**
>
> **The `exclude 0` line is strongly recommended to prevent possible security breaches from root users on other systems.**

```
global
default transparent
exclude 0
```

Figure 10-7. uid.rules File: Setting Global Defaults

In Figure 10-8, users have the same permissions as in the previous example, except remote user IDs **0** through **100** have MAXUID +1 permissions, and any user ID **732** have the same permission as local user ID **106**.

```
global
default transparent
exclude 0-100
map 732:106
```

Figure 10-8. uid.rules File: Global Mapping by Remote ID

In Figure 10-9, the users from computer **lucy** in domain **peanuts** are not mapped by the global rules. Instead, all users have the permissions of local user **mpg** except that user IDs 0 through 50 have MAXUID +1 permissions.

```
global
default transparent
exclude 0-100
map 732:106

host peanuts.lucy
default mpg
exclude 0-50
```

Figure 10-9. 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 need to have access to **passwd** and **group** files from those computers on your system. The following 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 10-10.

```
global
default transparent
exclude 0

host peanuts.lucy
exclude mary 0 uucp
map all
```

Figure 10-10. uid.rules File: Mapping by Name with **map all**

In the above example, each user name from computer **lucy** in domain **peanuts** have the same permissions as the same user name on your computer. The only exceptions are users **mary**, **root**, and **uucp**, who have MAXUID +1 permissions.

map name:name

You can also map particular remote user names into local user names or user IDs on your computer (Figure 10-11).

```
global
default transparent
exclude 0

host peanuts.lucy
default transparent
exclude 0
map mcn:jcb ral gwn:103
```

Figure 10-11. uid.rules File: Mapping Specific Users by Name

Here, all users from the computers are mapped into their same user ID with the following exceptions. Remote user **mcn** have the permission of local user **jcb**, remote user **ral** have permissions of local user **ral**, and remote user **gwn** 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 10-12 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 rules 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 10-12. 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 10-13 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 10-13. Output from **idload –k**

All mapping to **peanuts.lucy** would be 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 does 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* heading 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 want to know which machines are assigned as name servers and which machine is the current name server. To find out the current name server, type:

rfadmin

To find out 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 describe primary name server responsibilities as you set up your machine and domain. You may want to refer to the following paragraphs in that section if you want 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 picks 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 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

The **rfadmin –p** command passes the domain name server information to the primary, or one of the other computers listed in the domain's **rfmaster** file if it cannot contact the primary. (Note that name service is automatically 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, these functions cannot be done:

- maintaining domain member lists

  Computers cannot be added or deleted from domain member lists while a secondary is acting domain name server.

- changing RFS passwords

  Neither the secondary nor another computer can change RFS authentication passwords while a secondary is acting domain name server.

The secondary maintains 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 should not 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 **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 are lost. Active mounts and links, however, are not disturbed. The problem is that when the primary comes back up, each computer still thinks its resources are advertised but the primary has 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 **–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 RFS

  You can bring down RFS, bring it back up, and then readvertise your resources. This can be done 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 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 Guidelines* section in 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).

NOTE

The **sar** command and other performance analysis tools are in the System Performance Analysis Utilities. You must install these utilities before you can use the performance analysis tools.

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 10-14).

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**s per second.

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), or strictly local system calls.

```
$ sar-Dc
lucy lucy 3.0 1.0C m88000    02/14/88

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 10-14. Output from **sar –Dc**

NOTE

Some statistics do 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**(1M) 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 10-15).

```
$ sar -Du
lucy lucy 3.0 1.0C m88000  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 10-15. 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**(1M) 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 are retrieved across the network. With client caching, the first read of data brings the data into local buffers. Once data is in the local buffer, they remain there so subsequent reads can get the data locally.

Client caching is assigned by default on a system-wide 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 10-16).

```
$ sar -Db

charlie charlie 3.1 1.0C m88000    09/03/88

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 10-16. Output from **sar –Db**

The fields on this report are:

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 were present in the cache. (The information in this field has no performance implications when comparing using caching versus not using caching.)

lwrit/s
The total 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 10-17).

```
$ sar-C

charlie charlie 3.1 1.0C m88000    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 10-17. 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 the 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 that 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 re-opening 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 10-18).

```
$ sar-S

lucy lucy 3.0 1.0C m88000    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 10-18. 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 shown in the following sections.

Too Few Servers

If the receive queue is almost always busy (request %busy), you may want to raise the number of servers:

- 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:

- 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. **fusage** with no options produces a full report of data usage for all disks and advertised directories on your system, as shown in Figure 10-19.

```
# fusage
FILE USAGE REPORT FOR charlie
        /dev/dsk/c1d0s0       /
                              /
                                  charlie       649 KB
                                Clients           0 KB
                                  TOTAL         649 KB
        /dev/dsk/c1d0s8       /usr2
                              /usr2
                                  charlie       563 KB
                                Clients           0 KB
                                  TOTAL         563 KB
```

Figure 10-19. 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**(1M) 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. (**df** with no options produces information for all mounted remote resources, plus all locally mounted devices.) Figure 10-20 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 10-20. Output from **df**

NOTE

When multiple remote resources are reported that reside on the same disk, all listings of space on that disk, after the first, are 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 tells you the potential disk space available for you to write in. (Note that the space reported only 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. 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.

All parameters have set default values that should work well for an average system (see the table at the end of this section). The following paragraphs describe these parameters and cases where you may want to change them.

RFS Parameters

RFS parameters define the extent remote computers can use your resources, but they also control your own remote access to remote resources. 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.

The parameters described below are in **/etc/master.d/du**, except for NSRMOUNT, which is in **/etc/master.d/kernel**. Read these files for default values. If you change any of these values, see Chapter 6 for information on how to make the updated parameters take effect.

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:

- UID/GID tables (size and number).

  There is always two global tables, one UID and one GID. Also, any machine with a **host** entry in **uid.rules** or **gid.rules** files 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 is one four-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 uses 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 prints 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: 1) to turn off caching for your entire machine; 2) 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.

NHBUF
While this parameter is not exclusively an RFS parameter, it has implication 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 Chapter 6 for another discussion of NHBUF.)

The following table lists the RFS parameters and recommended values for different uses of RFS. "Client Only" means that your machine is only using remote resources, not sharing any from your own machine. "Server Only" means you only offer your resources to other machines without mounting any remote resources. "Client+Server" means you offer local resources and use remote resources.

RFS Tunable Parameter Settings											
File /etc/master.d/du											
	Client Only			Server Only			Client+Server			Default	Size
Parameter	2M	3M	4M	2M	3M	4M	2M	3M	4M	Value	per Entry in Bytes
-											
NSRMOUNT*	0	0	0	50	50	50	50	50	50	50	24
MAXGDP	10	15	20	24	32	32	24	32	32	24	104
NADVERTISE	0	0	0	25	25	25	25	25	25	25	32
NRCVD	40	60	80	300	400	500	150	250	350	150	48
NRDUSER	0	0	0	450	600	700	225	375	525	225	24
NSNDD	150	250	350	30	30	30	150	250	300	150	44
MINSERVE	0	0	0	3	3	3	3	3	3	3	9K
MAXSERVE	0	0	0	6	6	6	6	6	6	6	-
RFHEAP	2048	2048	2048	3072	3072	3072	3072	3072	3072	3072	1
NREMOTE	0	0	0	0	0	0	0	0	0	0	-
NLOCAL	0	0	0	0	0	0	0	0	0	0	-
RCACHETIME	10	10	10	10	10	10	10	10	10	10	-

* This tunable is found in **/etc/master.d/kernel**.

A Device Names and Specifications

Device Naming Conventions

The *System Administrator's Reference Manual* outlines the conventions for the names of peripheral device entries in **/dev** (see **intro**(7)). In addition, SYSTEM V/88 provides some shorthand naming notations for day-to-day references to the devices.

Note that the following shorthand notations do not apply to MVME327 devices:

/dev/[**r**]*xy*[**sz**]
for disk devices

/dev/[**r**]*xy*[**t**][**n**]
for cartridge tape devices (truncate)

/dev/[**r**]*xy*a[**n**]
for cartridge tape devices (append)

/dev/[**r**]*xy*[**n**]
for 9-track tape devices

/dev/tty*mp*
for TTY devices

where:

x
is the controller logical unit number (currently 0 through 9).

y
is the drive logical unit number on that controller.

z
is the slice number for disk devices.

m
is the back panel distribution module slot (1 through 9 corresponding to a serial I/O board, MVME332, or MVME335) for a TTY.

p
is the port on that module (1 through 4 for an MVME335; 1 through 6 or 8 for an MVME332XT). These nodes are created with the **portconfig**(1M) command and are based on a back panel board placement that follows the rules outlined in the system manual for your system. These rules are further presented in **portconfig**(1M) in the *System Administrator's Reference Manual*.

The controller numbering scheme partitions the set of controller logical unit numbers into groups of two, reserving two logical unit numbers for each controller type, with drive logical unit numbers assigned per the conventions of each controller, as shown in Table A-1.

NOTE

The following table does not apply to MVME327 devices.

Table A-1. Controller Numbering Scheme

LUN		Device	
Cntlr	**Drives**		
4	0	First MVME350	Only Streaming Tape Drive
6	0,1	First MVME355	First & Second 9-Track Drive
8	0-3	First MVME323	First - Fourth ESDI Drives

For example:

```
fsck /dev/r00s2          # Check the /usr file system on Winchester-based system
fsck /dev/r20s1          # Check the file system on slice 1 of the first SMD
tar cvbf 20 /dev/r40 .   # Tar current directory to streaming tape
```

For disk devices, the name **/dev/**[r]*xy*, without a slice number, is linked to **/dev/**[r]*xys7* which, by convention refers to the entire disk. See the following section for a discussion of disk slicing conventions.

For tape devices, the name **/dev/**[r]*xy*, without any qualifiers (e.g., *t*) is linked to the device providing the most general interface (open for truncate, rewind on close, 1600bpi [magnetic tape]).

All the above names are linked to the corresponding device-specific entries in **/dev/[r]dsk** and **/dev/[r]mt**. The names in these subdirectories are shown in Table A-2.

Table A-2. Disk and Tape Device Names in **/dev**

Device Name	Description
[r]dsk/m323_*y*sz	ESDI Drive *y* on First MVME323
[r]dsk/{m327}_*xy*sz	Winchester Drive *y* at SCSI address *x* on MVME327
[r]mt/{m327}_*xy*t[n]	Cartridge Tape *y* at SCSI address *x* on MVME327 (truncate)
[r]mt/{m327}_*xy*a[n]	Cartridge Tape *y* at SCSI address *x* on MVME327 (append)
[r]dsk/{m327}_d*xy*sz	Double Density Floppy Drive *y* at SCSI address *x* on MVME327
[r]dsk/{m327}_s*xy*sz	High Speed Floppy Drive *y* at SCSI address *x* on MVME327
[r]mt/m350_{0|1}t[n]	Cartridge Tape on First/Second MVME350 (truncate)
[r]mt/m350_{0|1}a[n]	Cartridge Tape on First/Second MVME350 (append)
rmt/m355_*y*[hsbn]	9-Track Tape Drive *y* on MVME355

In addition, the (linked) names in Table A-3 also exist in **/dev**.

Table A-3. Disk and Tape Device Linked Names in **/dev**

Name	Linked to	Purpose
/dev/TAPE.CART	**/dev/r40**	Simple archive to cartridge streamer
/dev/TAPE.9TRK	**/dev/r60**	Simple archive to 9-track tape (1600bpi)
/dev/rmt0[h][n]	**/dev/rmt/m355_0[h][n]**	Historical magnetic tape interface (1600 [6250] bpi)
/dev/[r]st{0|1}[a|t][n]	**/dev/[r]{4|5}0[a|t][n]**	General streaming tape interface
/dev/[r]st{2|3}[a|t][n]	**/dev/[r]{4|5}1[a|t][n]**	General streaming tape interface (no auto-retension)

For example:

```
tar cvbf 20 /dev/rst0 .     # Tar current directory to streaming tape (truncate)
tar cvbf 20 /dev/rst3a .    # Tar current directory to second streaming tape
                              (append; no retension)
```

Disk Block Size

The default file system logical block size is 1024 bytes (file systems may be created with 2K-, 4K-, or 8K-byte blocks via **mkfs2k**, **mkfs4k**, or **mkfs8k**, respectively). When **fsck** reports problems accessing a file system block, it is a **logical** block number. When **fsck** reports a block count for file system size, this is a **physical** block count.

The physical disk block size is 512 bytes. All utilities that report block counts report in 512-byte units. This includes **df**, **du**, **ls**, **fsck**, and **cpio**. The number–of–blocks argument to **mkfs** is a physical block count. The file system size in 1K blocks would be one-half this number (or 1/4, 1/8, or 1/16 of it for a 2K-, 4K-, or 8K-block file system, respectively).

Disk Partitioning

As shipped, SYSTEM V/88 uses one partitioning (slicing) scheme for all supported drives on the system, and separate schemes for the 337Mb Fujitsu SMD and 182Mb CDC ESDI drives. There are 8 slices possible per drive (0 through 7). Each slice begins at some offset and extends for a certain number of blocks (see **sledit** (1M)). By convention, slice 7 refers to the entire disk, beginning at physical block 0 (a physical block is 512 bytes). Slice 7 should not be used for any purpose other than formatting, initializing track redirection, or installing the disk-based bootloader (see **dinit**(1M)).

Floppy Disk Specifications

Two types of floppy disks are supported from the MVME327 disk driver:

- Low speed with single density track zero (normal floppy)
- High speed compatible with IBM PC/AT high speed

Table A-4. Floppy Disk Specifications

	Normal 5¼-inch	High Speed
Heads	2	2
Cylinders	80	80
Tracks	160	160
Tracks/Cylinder	2	2
Sectors/Track	16	15
Sector Size	256 bytes	512 bytes
Track 0 Sector Size	128	512
Total Capacity	653 Kbytes or 1276 blocks	1228 Kbytes or 2400 blocks
Gap	1	1
Blocks/Cyl	16	30

Definitions

speed
The IBM PC/AT 5¼-inch floppy disk drives are capable of *high* and *low* speed operation. *Low* speed operation makes the floppy appear like a normal 5¼-inch floppy disk. *High* speed operation makes the floppy appear like an 8-inch floppy disk.

slice
In SYSTEM V/88, all disks or floppies are subdivided into one or more logical disks called disk slices. In old style drivers, the slices are defined by hard coded disk slice tables. In new style drivers, slice tables for hard disks are kept on the disk media. For most floppy disks, slice tables continue to be hard coded.

whole disk slice

In SYSTEM V/88, one of the slices for each disk is defined to cover the whole disk (including volume identification and configuration information, diagnostic tracks, alternate tracks). If the disk driver permits *N* slices per disk (*N* is eight for most old drivers), slice *N*–1 contains the whole disk. In the case of MVME327 floppies, slice 7 contains the whole floppy.

usable disk slice

In SYSTEM V/88, MVME327 floppies, slice 0 contains only the usable portion of the floppy. This portion is the part of the disk (usually starting on track 2) following the volume identification and configuration information.

density

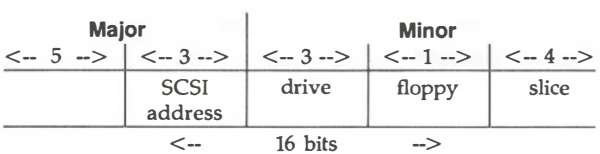
Two major recording methods are used for floppy disks: FM or *single* density and MFM or *double* density. All MVME327 floppy disk types are formatted and accessed using the MFM recording method (double density). Some floppy disk types record track zero using the FM method (single density).

blocks

In SYSTEM V/88, disks are managed in units of 512-byte disk blocks, regardless of physical sector size.

Major and Minor Numbers - SCSI Devices

In SYSTEM V/88, SCSI device major and minor numbers are used to describe a controller, drive, and slice and to indicate whether the device is a floppy, as shown in the following:



For the MVME327, the bit within the "floppy" field (if set to 1) designates that the device is a floppy disk. The floppy disk slicing is the same as that shown in Table A-5 substituting m327_ in the "Device Name" field.

Minor Numbers - MVME323

In SYSTEM V/88, MVME323 device minor numbers are used to describe a controller, drive, and slice:

<-- 1 -->	<-- 1 -->	<-- 2 -->	<-- 4 -->
controller	not used	drive	slice
<--	8 bits		-->

182Mb ESDI Specifications

Table A-5. CDC Wren III Model 94166 Specifications (no sector slip)

Heads	9
Cylinders	969 (968 usable)
Tracks	8712
Tracks/Cylinder	9
Sectors/Track	36
Sector Size	512 bytes
Alternate Tracks	144 (16 Cylinders, 5184 sectors)
Diagnostic Tracks	none
Gap	1
Blocks/Cyl	324

Again, the Gap and Blocks/Cyl values are those optimal for **mkfs**(1M) use.

You can see a complete disk description by running **ddefs**(1M). See Figure A-1 for a sample **ddefs** disk definition file.

```
                Disk definitions for `m327cdcIV'
                        Comment: CDC WRENIV 300mb SCSI Drive
                      Disk type: 4866
                 Format command: (none)
              Diagnostic tracks: no
              Bad spot strategy: PERFECT
   Maximum number of bad spots: 100
              Number of sectors: 597699
        Sector size (in bytes): 512
              Sectors per track: 47
                       Cylinders: 1413
                           Heads: 9
      Precompensation cylinder: 1413
              Sector interleave: 0
                  Spiral offset: 0
                      Step rate: 0
          Starting head number: 0
              ECC error length: 0
          Attributes mask (hex): 10
Extended attributes mask (hex): 0
          Attributes word (hex): 10
Extended attributes word (hex): 0
               Gap byte 1 (hex): 0
               Gap byte 2 (hex): 0
               Gap byte 3 (hex): 0
               Gap byte 4 (hex): 0
    Controller attribute (hex): 850
       Unformatted sector size: 0
              Sector slip count: 1
                    Slice count: 8
        Root filesystem offset: 324
          Root filesystem size: 60000
          /usr filesystem size: 200000
         /usr filesystem slice: 2
                     Swap size: 40000
                    Swap slice: 1
     End-of-disk reserved area: 0
                Alternates list: (1) 1
         Read bad list command: (none)
```

Figure A-1. Sample Disk Definition File

182Mb CDC ESDI Partitioning

Slice 0 of the disk begins 9 tracks (324 physical blocks) from the beginning of the disk. The area between the beginning of the disk and slice 0 is used by the firmware bootloader, **dinit**(1M), and the kernel. This area contains drive configuration information, alternate track redirection tables, and the disk-based bootloader.

Default 182Mb Partition Assignments

When the **root** disk is a 182Mb CDC ESDI drive, the SYSTEM V/88 default kernel configurations utilize the slices in the following way:

Area	Slice	Size
root	0	62208
swap	1	41472
usr	2	204120
free	3	0

You may divide the free space into additional partitions by using **sledit**(1M).

390Mb ESDI Specifications

Table A-6. CDC Wren V, Model 94186 Specifications (no sector slip)

Heads	15
Cylinders	1412 (1411 usable)
Tracks	21165
Tracks/Cylinder	15
Sectors/Track	36
Sector Size	512 bytes
Alternate Tracks	270 (18 Cylinders, 9720 sectors)
Diagnostic Tracks	none
Gap	1
Blocks/Cyl	540

Again, the Gap and Blocks/Cyl values are those optimal for **mkfs**(1M) use.

You can see a complete disk description by running **ddefs**(1M). See Figure A-1 for a sample **ddefs** disk definition file.

390Mb CDC ESDI Partitioning

Slice 0 of the disk begins 15 tracks (540 physical blocks) from the beginning of the disk. The area between the beginning of the disk and slice 0 is used by the firmware bootloader, **dinit**(1M), the kernel. This area contains drive configuration information, alternate track redirection tables, and the disk-based bootloader.

Default 390Mb Partition Assignments

When the **root** disk is a 390Mb CDC ESDI drive, the SYSTEM V/88 default kernel configurations utilize the slices in the following way:

Area	Slice	Size
root	0	60480
swap	1	43200
usr	2	203040
free	3	444420

You may divide the free space into additional partitions by using **sledit**(1M).

SCSI Devices

The following disks are currently supported by the MVME327 (where *XXX* is replaced with 327):

Table A-7. Supported Disk Devices

Size	Description	**ddefs**(1M) File
300Mb	CDC Model 94171 Wren IV	m*XXX*cdcIV
600Mb	CDC Model 94181 Wren V	m*XXX*cdcV
1.2Gb	CDC Model 94191 Wren VII	m*XXX*cdcVII

In addition, the following streaming tapes are currently supported by the MVME327.

Table A-8. Supported Tape Devices

Description	Format
Archive 2150S	QIC24, QIC120, QIC150
Kennedy 9-Track	Various

Table A-9. Default Configuration

SCSI Address	Device
0	Any SCSI disk
1	Any SCSI disk
2	Any SCSI disk
3	Any SCSI disk
4	Any SCSI tape
5	Any SCSI tape

SCSI Device Partitioning

The MVME327 driver supports dynamic slicing; the slice table resides on the disk. Therefore, the size and slicing of a disk does not have to be configured into the driver. Any size disk can be attached to these drivers without having to change any configuration information.

SCSI Device Specifications

You may see a complete disk description for the following disks by running **ddefs**(1M). See Figure A-1 for a sample **ddefs** disk definition file.

Table A-10. CDC 94171 Wren IV Specifications

Heads	9
Cylinders	1413
Tracks	12717
Tracks/Cylinder	9
Sectors/Track	47
Sector Size	512
Alternate Tracks/Cylinders	1
Diagnostics	none
Addressable Area	597699
Gap	0
Blocks/Cyl	423

Area	Slice	Size
root	0	60000
swap	1	40000
usr	2	200000
free	3	297051

Table A-11. CDC 94181 Wren V Specifications

Heads	15
Cylinders	1533
Tracks	22995
Tracks/Cylinder	15
Sectors/Track	52
Sector Size	512
Alternate Tracks/Cylinders	1
Diagnostics	none
Addressable Area	1195740
Gap	0
Blocks/Cyl	780

Area	Slice	Size
root	0	60000
swap	1	40000
usr	2	200000
free	3	895092

Table A-12. CDC 94191 Wren VII Specifications

Heads	15
Cylinders	1931
Tracks	28965
Tracks/Cylinder	15
Sectors/Track	69
Sector Size	512
Diagnostics	none
Number of Sectors	3997170
Gap	0
Blocks/Cyl	1035

Area	Slice	Size
root	0	100000
swap	1	50000
usr	2	250000
free	3	1598585

B Directories and Files

Introduction

Appendix B describes directories and files used by a System Administrator.

Directories

The directories of the **root** file system (/) are:

backups
Directory containing the control files and executables for the backup and restore facility (**sysadm br**) described in Chapter 5.

bin
Directory containing public commands.

dev
Directory containing special files that define all the devices on the system.

etc
Directory containing administrative programs and tables.

install
Directory used by System Administration to mount utilities packages for installation and removal (**/install** file system).

lib
Directory containing public libraries.

lost+found
Directory used by **fsck**(1M) to save disconnected files.

mnt
Directory used to temporarily mount file systems.

stand
Directory containing boot loader and bootable operating system.

tmp
Directory used for temporary files.

usr
Directory used to mount the **/usr** file system.

Files

The following files and directories are important in the administration of the computer:

- **/etc/checklist**
- **/etc/fstab**
- **/etc/gettydefs**
- **/etc/group**
- **/etc/init.d Directory**
- **/etc/inittab**
- **/etc/motd**
- **/etc/passwd**
- **/etc/profile**
- **/etc/rc0**
- **/etc/rc0.d Directory**
- **/etc/rc2**
- **/etc/rc2.d Directory**
- **/etc/init.d Directory**
- **/etc/rc3**
- **/etc/rc3.d Directory**
- **/etc/rstab**
- **/etc/save.d Directory**
- **/etc/shutdown**
- **/etc/shutdown.d Directory**
- **/etc/TIMEZONE**
- **/etc/utmp**
- **/etc/wtmp**

- **/usr/adm/sulog**
- **/usr/lib/cron/log**
- **/usr/lib/help/HELPLOG**
- **/usr/lib/spell/spellhist**
- **/usr/news Directory**
- **/usr/spool/cron/crontabs Directory**

Each of these files is briefly described in this appendix.

/etc/checklist

The **/etc/checklist** file is used to define a default list of file system devices to be checked for consistency by **/etc/fsck** and **/etc/ncheck**. The character (raw) device partition for the file system should be identified. The devices listed normally correspond to those mounted when the system is in the multi-user mode (run level 2). The **root** file system (**/dev/rdsk/m323_0s0**) should not be listed in this file. Remember that with the exception of **root**, a file system must be unmounted to be checked. Therefore, the **checklist** file is a convenience for use when in the single-user mode of operation with only the **root** file system mounted. When the system is delivered, this file is empty. For a single-disk system the list is brief. A typical **/etc/checklist** file is shown in Figure B-1. (See the **checklist**(4) manual page in the *Programmer's Reference Manual* for additional information.)

```
/dev/dsk/m323_1s2
/dev/dsk/m323_0s2
```

Figure B-1. Typical **/etc/checklist** File

/etc/fstab

The **/etc/fstab** file is used as an argument to the **/etc/mountall** command. The **fstab** file specifies the file system(s) to be mounted by **/etc/mountall** and remote file system(s) to be mounted by **/etc/rmountall**. A typical **/etc/fstab** file is shown in Figure B-2. The format of the file is the block device name followed by the mount point name. (See the **mountall**(1M) manual page in the *System Administrator's Reference Manual* for additional information.)

```
/dev/dsk/m323_0s2 /usr
/dev/dsk/m323_1s2 /usr2
RES1 /mnt -d
```

Figure B-2. Typical **/etc/fstab** File

/etc/gettydefs

The **/etc/gettydefs** file contains information that is used by **/etc/getty** to set the speed and terminal settings for a line. The **getty** command accesses the **gettydefs** file with a label. The general format of the **gettydefs** file is:

label# initial-flags # final-flags #login-prompt #next-label

Each line entry in the **gettydefs** file is followed by a blank line. (Refer to the **gettydefs**(4) manual page in the *Programmer's Reference Manual* for complete information.) Figure B-3 shows a typical **/etc/gettydefs** file.

```
19200# B19200 HUPCL # B19200 SANE IXANY TAB3 HUPCL #login: #9600
9600# B9600 HUPCL # B9600 SANE IXANY TAB3 HUPCL #login: #4800
4800# B4800 HUPCL # B4800 SANE IXANY TAB3 HUPCL #login: #2400
2400# B2400 HUPCL # B2400 SANE IXANY TAB3 HUPCL #login: #1200
1200# B1200 HUPCL # B1200 SANE IXANY TAB3 HUPCL #login: #300
300# B300 HUPCL # B300 SANE IXANY TAB3 HUPCL #login: #19200
console# B9600 HUPCL OPOST ONLCR # B9600 SANE IXANY TAB3
#Console Login: #console1
console1# B1200 HUPCL OPOST ONLCR # B1200 SANE IXANY TAB3
#Console Login: #console2
console2# B300 HUPCL OPOST ONLCR # B300 SANE IXANY TAB3
#Console Login: #console3
console3# B2400 HUPCL OPOST ONLCR # B2400 SANE IXANY TAB3
#Console Login: #console4
console4# B4800 HUPCL OPOST ONLCR # B4800 SANE IXANY TAB3
#Console Login: #console5
console5# B19200 HUPCL OPOST ONLCR # B19200 SANE IXANY TAB3
#Console Login: #console
contty# B9600 HUPCL OPOST ONLCR # B9600 SANE IXANY TAB3
#login: #contty1
contty1# B1200 HUPCL OPOST ONLCR # B1200 SANE IXANY TAB3
#login: #contty2
contty2# B300 HUPCL OPOST ONLCR # B300 SANE IXANY TAB3
#login: #contty3
contty3# B2400 HUPCL OPOST ONLCR # B2400 SANE IXANY TAB3
#login: #contty4
contty4# B4800 HUPCL OPOST ONLCR # B4800 SANE IXANY TAB3
#login: #contty5
```

Figure B-3. Typical **gettydefs** File (1 of 2)

```
contty5# B19200 HUPCL OPOST ONLCR # B19200 SANE IXANY TAB3
#login: #contty
pty# B9600 HUPCL OPOST ONLCR # B9600 SANE IXANY TAB3 #PC login: #pty
4800H# B4800 # B4800 SANE IXANY TAB3 HUPCL #login: #9600H
9600H# B9600 # B9600 SANE IXANY TAB3 HUPCL #login: #19200H
19200H# B19200 # B19200 SANE IXANY TAB3 HUPCL #login: #2400H
2400H# B2400 # B2400 SANE IXANY TAB3 HUPCL #login: #1200H
1200H# B1200 # B1200 SANE IXANY TAB3 HUPCL #login: #300H
300H# B300 # B300 SANE IXANY TAB3 HUPCL #login: #4800H
conttyH# B9600 OPOST ONLCR # B9600 HUPCL SANE IXANY TAB3
#login: #contty1H
contty1H# B1200 OPOST ONLCR # B1200 HUPCL SANE IXANY TAB3
#login: #contty2H
contty2H# B300 OPOST ONLCR # B300 HUPCL SANE IXANY TAB3
#login: #contty3H
contty3H# B2400 OPOST ONLCR # B2400 HUPCL SANE IXANY TAB3
#login: #contty4H
contty4H# B4800 OPOST ONLCR # B4800 HUPCL SANE IXANY TAB3
#login: #contty5H
contty5H# B19200 OPOST ONLCR # B19200 HUPCL SANE IXANY TAB3
#login: #conttyH
```

Figure B-3. Typical **gettydefs** File (2 of 2)

/etc/group

The **/etc/group** file describes each group to the system. An entry is added for each new group. Each entry in the file is one line and consists of four fields, separated by a colon (:):

group name:password:group id:login names

where:

group name
defines the group name. The group name is from three to eight characters long. The first character is alphabetic; the rest are alphanumeric. No uppercase characters appear.

password
contains the encrypted group password. The encrypted group password contains 13 bytes (characters). The actual password is limited to a maximum of 8 bytes. The encrypted password can be followed by a comma and up to 4 more bytes of password aging information. The use of group passwords is discouraged.

group id
contains the group identification number, which must be between 0 and 60,000. Group identification numbers 0 through 99 are reserved; 0 indicates the super-user (root). Commas are not entered in this field.

login names
contains a list of all login names in the group; names are separated by commas. The names listed may use the **/newgrp** command to become a member of the group.

Figure B-4 shows a typical **/etc/group** file.

```
root::0:root
other::1:
bin::2:root,bin,daemon
sys::3:root,bin,sys,adm
adm::4:root,adm,daemon
mail::6:root
rje::8:rje,shqer
daemon::12:root,daemon
```

Figure B-4. Typical **/etc/group** File

/etc/init.d Directory

The **/etc/init.d** directory contains executable files used in upward and downward transitions to all system run levels. These files are linked to files beginning with **S** (start) or **K** (stop) in **/etc/rc*n*.d**, where *n* is the appropriate run level. Files are not executed from this directory. They are only executed from **/etc/rc*n*.d** directories.

/etc/inittab

The **/etc/inittab** file contains instructions for the **/etc/init** command. The instructions define the processes that are to be created or terminated for each initialization state. Initialization states are called run levels or run-states. By convention, run level 1 (or S or s) is single-user mode; run levels 2 and 3 are multi-user modes. Chapter 3, *Processor Operations*, summarizes the various run levels and describes their uses. (See the **inittab**(4) manual page in the *Programmer's Reference Manual* for additional information.) Figure B-5 shows a typical **/etc/inittab** file. The typical entry is a series of fields separated by a colon (:):

identification:run-state:action:process

where:

identification
is a one- or two-character identifier for the line entry. The identifier is unique for a line.

run-state
defines the run level in which the entry is to be processed.

action
defines how **/etc/init** treats the process field. (Refer to the **inittab**(4) manual page in the *Programmer's Reference Manual* for complete information.)

process
defines the shell command that is to be executed.

```
zu::sysinit:/etc/bzapunix </dev/console >/dev/console 2>&1
fs::sysinit:/etc/bcheckrc </dev/console >/dev/console 2>&1
ck::sysinit:/etc/setclk </dev/console >/dev/console 2>&1
mt:23:bootwait:/etc/brc </dev/console >/dev/console 2>&1
pt:23:bootwait:/etc/ports </dev/console >/dev/console 2>&1
is:2:initdefault:
p1:s1234:powerfail:/etc/led -f  # start green LED flashing
p3:s1234:powerfail:/etc/shutdown -y -i0 -g0 >/dev/console 2>&1
s0:056:wait:/etc/rc0 >/dev/console 2>&1 </dev/console
s1:1:wait:/etc/shutdown -y -iS -g0 >/dev/console 2>&1 </dev/console
s2:23:wait:/etc/rc2 >/dev/console 2>&1 </dev/console
s3:3:wait:/etc/rc3 >/dev/console 2>&1 </dev/console
fl:056:wait:/etc/led -f >/dev/console 2>&1 </dev/console
of:0:wait:/etc/uadmin 2 0 >/dev/console 2>&1 </dev/console
fw:5:wait:/etc/uadmin 2 2 >/dev/console 2>&1 </dev/console
RB:6:wait:echo "The system is being restarted." >/dev/console 2>&1
rb:6:wait:/etc/uadmin 2 1 >/dev/console 2>&1 </dev/console
co:234:respawn:/etc/getty console console
ct:234:off:/etc/getty -t 60 contty 4800
he:234:respawn:sh -c 'sleep 20 ;exec /etc/hdelogger >/dev/console 2>&1'
11:234:off:/etc/getty tty11 9600
12:234:off:/etc/getty tty12 9600
13:234:off:/etc/getty tty13 9600
14:234:off:/etc/getty tty14 9600
15:234:off:/etc/getty tty15 9600
31:234:respawn:/usr/lib/uucp/uugetty -r -t 60 tty31 4800H
32:234:respawn:/usr/lib/uucp/uugetty -r -t 60 tty32 4800H
33:234:off:/etc/getty tty33 9600
34:234:off:/etc/getty tty34 9600
35:234:off:/etc/getty tty35 9600
```

Figure B-5. Typical **/etc/inittab** File

/etc/motd

The **/etc/motd** file contains the message-of-the-day. The message-of-the-day is output by instructions in the **/etc/profile** file after a successful login. Keep the message short and to the point. Use the **/usr/news** file(s) for lengthy, more explicit messages.

/etc/passwd

The **/etc/passwd** file identifies each user to the system. An entry is added for each new user. Each entry in the file is one line and consists of seven fields; the fields are separated by a colon (:):

login name:passwd:user:group:account:login directory:program

where:

login name
defines the login name. The login name is from three to six characters. The first character is alphabetic; the rest are alphanumeric. No uppercase characters appear.

passwd
contains the encrypted login password. The encrypted login password contains 13 bytes (characters). The actual password is limited to a maximum of 8 bytes. The encrypted password can be followed by a comma and up to 4 more bytes of password aging information.

user id
field contains the user identification number, which must be between 0 and 60,000. Group identification numbers 0 through 99 are reserved; 0 indicates the super-user (root). Commas are not entered in this field.

group id
contains the group identification number, which must be between 0 and 60,000. Group identification numbers 0 through 99 are reserved; 0 indicates the super-user (root). Commas are not entered in this field.

account
is used by accounting programs. This field typically contains the user name, department number, and bin number.

login directory
defines the full path name of the login directory.

program
defines the program to be executed after login. If it is null, the shell (**/bin/sh**) is invoked.

Figure B-6 shows a typical **/etc/passwd** file. (See the **passwd**(4) manual page in the *Programmer's Reference Manual* for additional information.)

```
root:CQCpoiY.hyx5M:0:1:0000-Admin(0000):/:
daemon:DT8E7TMqEGAE.:1:1:0000-Admin(0000):/:
bin:NTwEs.ajCTuQ6:2:2:0000-Admin(0000):/bin:
sys:ZQJgJg55l50HI:3:3:0000-Admin(0000):/usr/src:
adm:iQ0OMArb9NXZM:4:4:0000-Admin(0000):/usr/adm:
uucp:rQ5fM/1P8Z6yU:5:5:0000-uucp(0000):/usr/lib/uucp:
nuucp:4Rka.6jZpmZeI:10:10:0000-uucp(0000):
/usr/spool/uucppublic:/usr/lib/uucp/uuc
ico
rje:FRStaYWcMn0do:18:18:0000-rje(0000):/usr/rje:
trouble:PRS31ul156VCI:70:1:trouble(0000):/usr/lib/trouble:
lp:YR6Oh4EJQfTg2:71:2:0000-lp(0000):/usr/spool/lp:
setup:xEwCpdK/pnbHk:0:0:general system administration:
/usr/admin:/bin/rsh
powerdown:BF.j6TnLXfDjc:0:0:general system administration:
/usr/admin:/bin/rsh
sysadm:NFS9vmN./SOm2:0:0:general system administration:
/usr/admin:/bin/rsh
listen:np:37:4:Network Admin:/usr/net/nls:
ral::100:1:Robert Lippman:/usr/ral:
mcn::101:1:Chris Negus:/usr/mcn:
kol::102:1:Kathy O'Leary:/usr/kol:
jcb::103:1:Jim Blinn:/usr/jcb:
```

Figure B-6. Typical **/etc/passwd** File

/etc/profile

The default profile for all users is in the **/etc/profile** file. The standard (default) environment for all users is established by the instructions in the **/etc/profile** file. The system administrator can modify this file to set options for the **root** login. For example, the following can be added to the **/etc/profile** for the **root** login to cause the erase character to back up and to set the TERM variable:

```
if [ ${LOGNAME} = root ]
        then
                stty echoe
                echo "Enter TERM: \c"
                read TERM
                export TERM
```

Figure B-7 shows the default profile.

```
# The profile that all logins get before using their own .profile.
trap ""  2 3
export LOGNAME
. /etc/TIMEZONE
# Login and -su shells get /etc/profile services.
# -rsh is given its environment in its .profile.
case "$0" in
-su )
        export PATH
        ;;
-sh )
        export PATH
        # Allow the user to break the Message-Of-The-Day only.
        trap "trap '' 2"  2
        cat -s /etc/motd
        trap "" 2
        if mail -e
        then
                echo "you have mail"
        fi
        if [ ${LOGNAME} != root ]
        then
                news -n
        fi
        ;;
esac
umask 022
trap  2 3
```

Figure B-7. Standard **/etc/profile** File

/etc/rc0

The **/etc/rc0** file contains a shell script that is executed by **/etc/shutdown** for transitions to single-user state, and by **/etc/init** on transitions to run levels 0, 5, and 6. Files in the **/etc/shutdown.d** and **/etc/rc0.d** directories are executed when **/etc/rc0** is run. The file **K00ANNOUNCE** in **/etc/rc0.d** prints the message **"System services are now being stopped"**. Any task that you want executed when the system is taken to run levels 0, s, 5, or 6 can be done by adding a file to the **/etc/shutdown.d** directory. Figure B-8 shows a typical **/etc/rc0** file.

```
# "Run Commands" for init state 0
# Leaves the system in a state where it is safe to turn off the power
# or go to firmware.
stty sane tab3 2>/dev/null
echo 'The system is coming down.  Please wait.
if [ -d /etc/shutdown.d ]
then
        for f in /etc/shutdown.d/*
        { if [ -s $f ]
                then
                        /bin/sh ${f}
                fi
        }
fi
#       End of historical section
if [ -d /etc/rc0.d ]
then
        for f in /etc/rc0.d/K*
        {
                if [ -s ${f} ]
                then
                        /bin/sh ${f} stop
                fi
        }
.
.
.
```

Figure B-8. Typical **/etc/rc0** File (1 of 2)

```
.
.
.
# system cleanup functions ONLY (things that end fast!)
for f in /etc/rc0.d/S*
        {
                if [ -s ${f} ]
                then
                        /bin/sh ${f} start
                fi
        }
fi
trap "" 15
kill -15 -1
sleep 10
/etc/killall  9
sleep 10
sync;sync;sync
/etc/umountall
stty sane 2>/dev/null
sync;  sync
echo '
The system is down.'
sync
```

Figure B-8. Typical **/etc/rc0** File (2 of 2)

/etc/rc0.d Directory

The **/etc/rc0.d** directory contains files executed by **/etc/rc0** for transitions to system run levels 0, 5, and 6. Files in this directory are linked from the **/etc/init.d** directory, and begin with either a **K** or an **S**. **K** indicates processes that are stopped, and **S** indicates processes that are started when entering run levels 0, 5, or 6.

/etc/rc2

The **/etc/rc2** file contains a shell script that is executed by **/etc/init** on transitions to run level 2 (multi-user state). Executable files in the **/etc/init.d** and any executable files beginning with **S** or **K** in **/etc/rc2.d** directories are executed when **/etc/rc2** is run. All files in **rc2.d** are linked from files in the **/etc/init.d** directory. The following are descriptions of some of those files. These files are prefixed with an **S** or a **K** and a number in the **/etc/rc2.d** directory.

MOUNTFILESYS
Sets up and mounts file systems. Builds the mount table and mounts the **root** (*/*) and user (**/usr**) file systems. Makes the **/usr/tmp** directory, cleaning up (deleting) any previous files in that directory.

cron
Starts the **cron** daemon by executing **/etc/cron**.

sysetup
Removes the **/etc/ps_data** to force the **/bin/ps** command to read the **/unix** file. Outputs the system configuration if the **/etc/prtconf** command exists. Outputs the system trademark information.

uucp
When basic networking is added to the system, the **uucp** file is added to this directory. The **uucp** file deletes **uucp** locks (LCK*), status files (STST*), and temporary files (TM*) under the **/usr/spool/uucp** directory structure.

lp
When line printer spooling is added to the system, the **lp** file is added to the **init.d**. The **lp** file removes the spooler lock file and starts the scheduler.

Other files may also be added to **/etc/rc2.d** and **/etc/init.d** directories as a function of adding hardware or software to the system. Figure B-9 shows a typical **/etc/rc2** file.

```
# "Run Commands" executed when the system is changing to init state 2,
# traditionally called "multi-user".
. /etc/TIMEZONE
# Pickup start-up packages for mounts, daemons, services, etc.
set `who -r`
if [ $9 = "S" ]
then
    echo 'The system is coming up.  Please wait.'
    BOOT=yes
    if [ -f /etc/init.d/PRESERVE ]       # historical segment for vi and ex
        then
                mv /etc/init.d/PRESERVE /etc/init.d
                ln /etc/init.d/PRESERVE /etc/rc2.d/S02PRESERVE
        fi
elif [ $7 = "2" ]
then
        echo 'Changing to state 2.'
        if [ -d /etc/rc2.d ]
        then
                for f in /etc/rc2.d/K*
                {
                        if [ -s ${f} ]
                        then
                                /bin/sh ${f} stop
                        fi
                }
        fi
fi
  .
  .
  .
```

Figure B-9. Typical **/etc/rc2** File (1 of 2)

```
.
.
.
if [ -d /etc/rc2.d ]
then
        for f in /etc/rc2.d/S*
        {
                if [ -s ${f} ]
                then
                        /bin/sh ${f} start
                fi
        }
fi
if [ "${BOOT}" = "yes" ]
then
        stty sane tab3 2>/dev/null
fi
if [ "${BOOT}" = "yes" -a -d /etc/init.d ]
then
        for f in `ls /etc/init.d`
        {
                if [ ! -s /etc/init.d/${f} ]
                then
                        /bin/sh /etc/init.d/${f}
                fi
        }
fi
if [ "${BOOT}" = "yes" -a $7 = "2" ]
then
        echo 'The system is ready.'
elif [ $7 = "2" ]
then
        echo 'Change to state 2 has been completed.'
fi
```

Figure B-9. Typical **/etc/rc2** File (2 of 2)

/etc/rc2.d Directory

The **/etc/rc2.d** directory contains files executed by **/etc/rc2** for transitions to system run level 3. Files in this directory are linked from the **/etc/init.d** directory, and begin with either a **K** or an **S**. **K** indicates processes that should be stopped, and **S** indicates processes that should be started when entering run levels 2 or 3.

/etc/init.d Directory

The **/etc/init.d** directory contains executable files that do the various functions needed to initialize the system to run level 2. The files are executed when **/etc/rc2** is run. (This directory is only maintained for compatibility reasons.)

/etc/rc3

The **/etc/rc3** file is executed by **/etc/init**. It executes the shell scripts in **/etc/rc3.d** on transitions to system run level 3 (the RFS state).

/etc/rc3.d Directory

The **/etc/rc3.d** directory contains files executed by **/etc/rc3** for transitions to system run level 3 (multi-user mode). Files in this directory are linked from the **/etc/init.d** directory, and begin with either a **K** or an **S**. **K** indicates processes that should be stopped, and **S** indicates processes that should be started when entering run level 3.

/etc/rstab

The **/etc/rstab** file is used to specify the RFS resources from your machine that are automatically offered to remote machines upon entering system run level 3 (RFS state). Entries in this file should be entire **adv**(1M) command lines. (See Chapter 10 for details.)

/etc/save.d Directory

The **/etc/save.d** directory contains files that are used by the System Administration Menu commands associated with backing up data. The following files are included:

except
A list of the directories and files that should not be copied as part of a backup (**savefiles**) is maintained in this file.

timestamp/...
The date and time of the last backup (volume or incremental) is maintained for each file system in the **/etc/timestamp** directory.

/etc/shutdown

The **/etc/shutdown** file contains a shell script to shut down the system gracefully in preparation for system backup or scheduled downtime. After stopping all nonessential processes, the **shutdown** script executes files in the **/etc/shutdown.d** directory by calling **/etc/rc0** for transitions to run level s or S. For transitions to other run levels, the **shutdown** script calls **/etc/init**. Figure B-10 shows a typical **/etc/shutdown** file.

```
# Sequence performed to change the init stat of a machine.
# This procedure checks to see if you are permitted and allows an
# interactive shutdown.  The actual change of state, killing of
# processes and such are performed by the new init state, say 0,
# and its /etc/rc0.
# Usage:  shutdown [ -y ] [ -g<grace-period> ] [ -i<init-state> ]
#!      chmod +x ${file}
if [ `pwd` !- / ]
then
echo "$0:  You must be in the / directory to run /etc/shutdown."
exit 1
fi
#       Check the user id.
if [ -x /usr/bin/id ]
then
        eval `id  |  sed 's/[^a-z0-9].*//'`
        if [ "${uid:-0}" -ne 0 ]
        then
                echo "$0:  Only root can run /etc/shutdown."
                exit 2
        fi
fi
grace=60
askconfirmation=yes
initstate=s
while [ $# -gt 0 ]
do
        case $1 in
        -g[0-9]* )
                grace=`expr "$1" : '-g\(.*\)'`
                ;;
        -i[Ss0156] )
.
.
.
```

Figure B-10. Typical **/etc/shutdown** File (1 of 4)

```
.
.
.
        initstate=`expr "$1" : '-i\(.*\)'`
        ;;
   -i[234] )
        initstate=`expr "$1" : '-i\(.*\)'`
        echo "$0:  Initstate $i is not for system shutdown"
        exit 1
        ;;
   -y )
        askconfirmation=
        ;;
   -* )
        echo "Illegal flag argument '$1'"
        exit 1
        ;;
   * )
        echo "Usage:  $0 [ -y ] [ -g<grace> ] [ -i<initstate> ]"
        exit 1
        esac
        shift
done
if [ -n "${askconfirmation}" -a -x /etc/ckbupscd ]
        then
#       Check to see if backups are scheduled at this time
        BUPS=`/etc/ckbupscd`
        if [ "$BUPS" != "" ]
        then
        echo "$BUPS"
        echo "Do you wish to abort this shutdown and return to
command level to do these backups? [y, n]                    read YORN
.
.
.
```

Figure B-10. Typical **/etc/shutdown** File (2 of 4)

```
.
.
.
                if [ "$YORN" = "y" -o "$YORN" = "Y" ]
                        then
                        exit 1
                fi
        fi
fi
if [ -z "${TZ}"  -a  -r /etc/TIMEZONE ]
then
    . /etc/TIMEZONE
fi
echo '
Shutdown started.    date
echo
sync
cd /
trap "exit 1"  1 2 15
a="`who  |  wc -l`"
if [ ${a} -gt 1  -a  ${grace} -gt 0 ]
then
su adm -c /etc/wall<<-!
        -The system will be shut down in ${grace} seconds.
        Please log off now-.
!
        sleep ${grace}
fi
/etc/wall <<-!
        -THE SYSTEM IS BEING SHUT DOWN NOW ! ! !-
        -Log off now or risk your files being damaged.-
!
sleep ${grace}
.
.
.
```

Figure B-10. Typical **/etc/shutdown** File (3 of 4)

```
.
.
.
if [ ${askconfirmation} ]
then
        echo "Do you want to continue? (y or n):    \c"
        read b
else
        b=y
fi
if [ "$b" != "y" ]
then
        /etc/wall <<-\!
        False Alarm:  The system will not be brought down.
        !
        echo 'Shut down aborted.'
        exit 1
fi
case "${initstate}" in
s | S )
    . /etc/rc0
esac
/etc/init ${initstate}
.
.
.
```

Figure B-10. Typical **/etc/shutdown** File (4 of 4)

/etc/shutdown.d Directory

The executable files in **/etc/shutdown.d** perform the various functions required during the transition to the single-user state (run levels 1, s, or S). (This directory is only maintained for compatibility reasons.)

/etc/TIMEZONE

The **/etc/TIMEZONE** file sets the time zone shell variable TZ. The TZ variable is initially established for the system via the System Administration **setup** function. The TZ variable in the **TIMEZONE** file is changed by the System Administration **timezone** command (**sysadm timezone**). The TZ variable can be redefined on a user (login) basis by setting the variable in the associated **.profile**. The **TIMEZONE** file is executed by **/etc/rc2**. Figure B-11 shows a typical **/etc/TIMEZONE** file.

The format of **TZ** is:

TZ=*TTT#SSS*

where:

TTT
the three-character abbreviation for the local time zone.

#
the number of hours that the local time zone differs from Greenwich Mean Time (GMT). This field can be entered as a positive or negative number.

SSS
the three-character abbreviation for the local daylight savings time zone. This field is entered only if daylight savings time is observed.

```
# Set timezone environment to default for the machine
TZ=EST5EDT
export TZ
```

Figure B-11. Typical **/etc/TIMEZONE** File

/etc/utmp

The **/etc/utmp** file contains information on the run-state of the system. This information is accessed with a **who -a** command.

/etc/wtmp

The **/etc/wtmp** file contains a history of system logins. The owner and group of this file must be **adm**, and the access permissions must be 664. Each time **login** is run this file is updated. As the system is accessed, this file increases in size. Periodically, this file should be cleared or truncated. The command line **>/etc/wtmp** when executed by **root** creates the file with nothing in it. The following command line limits the size of the **/etc/wtmp** file to the last 3600 characters in the file:

```
tail -3600c /etc/wtmp > /tmp/wtmp; mv /tmp/wtmp /etc/wtmp
```

Note that **/etc/cron**, **/etc/rc0**, or **/etc/rc2** can be used to clean up the **wtmp** file. To use one of these functions, add the appropriate command line to the **/usr/spool/cron/crontab/root**, **/etc/shutdown.d/...**, or **/etc/init.d/**, **rc2.d**, **rc3.d ...** file.

/usr/adm/sulog

The **/usr/adm/sulog** file contains a history of substitute user (**su**) command usage. As a security measure, this file should not be readable by **others**. The **/usr/adm/sulog** file should be periodically truncated to keep the size of the file within a reasonable limit. Note that **/etc/cron**, **/etc/rc0**, or **/etc/rc2** can be used to clean up the **sulog** file. To use one of these functions, add the appropriate command line to the **/usr/spool/cron/crontab/root**, **/etc/shutdown.d/...**, or **/etc/init.d/**, **rc2.d**, **rc3.d ...** file. The following command lines limit the size of the log file to the last 100 lines in the file:

```
tail -100 /usr/adm/sulog > /tmp/sulog
mv /tmp/sulog /usr/adm/sulog
```

Figure B-12 shows the contents of a typical **/usr/adm/sulog** file.

```
SU 08/18 12:35 + console root-sysadm
SU 08/18 16:11 + console root-sysadm
SU 08/18 16:16 + console root-sysadm
SU 08/18 23:45 + tty?? root-uucp
SU 08/19 11:53 + console root-sysadm
SU 08/19 15:25 + console root-sysadm
SU 08/19 23:45 + tty?? root-uucp
SU 08/20 10:16 + console root-adm
SU 08/20 10:33 + tty24 rar-root
SU 08/20 10:42 + console root-sysadm
SU 08/20 10:59 + console root-root
SU 08/20 11:01 + console root-sysadm
SU 08/20 12:36 + tty11 bin-bin
SU 08/20 12:37 + tty11 tws-bin
SU 08/20 14:42 - tty24 awa-sys
SU 08/20 14:47 - tty24 awa-sys
SU 08/20 14:48 + tty24 awa-root
SU 08/20 15:44 + console root-sysadm
```

Figure B-12. Typical **/usr/adm/sulog** File

/usr/lib/cron/log

A history of all actions taken by **/etc/cron** is recorded in the **/usr/lib/cron/log** file. The **/usr/lib/cron/log** file should be periodically truncated to keep the size of the file within a reasonable limit. Note that **/etc/cron**, **/etc/rc0**, or **/etc/rc2** can be used to clean up the **/usr/lib/cron/log** file. To use one of these functions to limit the size of a log file, add the appropriate command line to the **/usr/spool/cron/crontab/root**, **/etc/shutdown.d/...**, or **/etc/init.d/ rc2.d, rc3.d ...** file, as applicable. The following command line limits the size of the log file to the last 100 lines in the file:

tail -100 /usr/lib/cron/log > /tmp/log; mv /tmp/log /usr/lib/cron/log

Figure B-13 shows the information typically found in the **/usr/lib/cron/log** file.

```
! *** cron started ***   pid = 237 Sun Aug 19 14:06:45 1984
>  CMD: /usr/lib/uucp/uudemon.hour > /dev/null
>  root 251 c Sun Aug 19 14:11:00 1984
<  root 251 c Sun Aug 19 14:11:01 1984
>  CMD: /usr/lib/uucp/uudemon.poll > /dev/null
>  root 370 c Sun Aug 19 14:30:00 1984
<  root 370 c Sun Aug 19 14:30:03 1984
>  CMD: /usr/lib/uucp/uudemon.hour > /dev/null
>  root 417 c Sun Aug 19 14:41:01 1984
<  root 417 c Sun Aug 19 14:41:02 1984
>  CMD: /usr/lib/uucp/uudemon.poll > /dev/null
>  root 452 c Sun Aug 19 15:01:00 1984
<  root 452 c Sun Aug 19 15:01:04 1984
>  CMD: /usr/lib/uucp/uudemon.hour > /dev/null
>  root 460 c Sun Aug 19 15:11:00 1984
<  root 460 c Sun Aug 19 15:11:00 1984
>  CMD: /usr/lib/uucp/uudemon.poll > /dev/null
>  root 541 c Sun Aug 19 15:30:00 1984
<  root 541 c Sun Aug 19 15:30:07 1984
```

Figure B-13. Typical **/usr/lib/cron/log** File

/usr/lib/help/HELPLOG

If **help** monitoring has been enabled, a history of all actions taken by the **/usr/bin/help** command is kept in the **/usr/lib/help/HELPLOG** file. The **HELPLOG** file is copied to **/usr/lib/help/oHELPLOG** and a new **/usr/lib/help/HELPLOG** file is created by the **/usr/lib/help/helpclean** command. Executing the **helpclean** command twice in succession zeros out the **HELPLOG** and the **oHELPLOG** files. You can also use **/etc/cron**, **/etc/rc0**, or **/etc/rc2** to clean up the **HELPLOG** file. To use one of these functions, add the appropriate command line to the **/usr/spool/cron/crontab/root**, **/etc/shutdown.d/...**, or **/etc/init.d/ rc2.d, rc3.d ...** file, as applicable. The following command lines limit the size of the log file to the last 100 lines in the file:

```
tail -100 /usr/lib/help/HELPLOG > /tmp/help
mv /tmp/help /usr/lib/help/HELPLOG
```

Figure B-14 shows the information typically found in the **/usr/lib/help/HELPLOG** file.

```
login~bin       uname~jundland     date~Mon Aug 20 12:51:03 EDT 1984
name~locate     response~'l'       status~OK
name~locate     response~'d'       status~ERROR
name~getkey     response~'k'       status~OK
name~keysrch    response~'list'    status~OK
name~quit       response~'q'       status~OK
login~bin       uname~jundland     date~Mon Aug 20 12:51:41 EDT 1984
```

Figure B-14. Typical **/usr/lib/help/HELPLOG** File

/usr/lib/spell/spellhist

If the Spell Utilities are installed, a history of all words that **spell**(1) fails to match is kept in the **/usr/lib/spell/spellhist** file. Periodically, this file should be reviewed for words that should be added to the dictionary. After the **spellhist** file is reviewed, it can be cleared.

/usr/news Directory

The **/usr/news** directory contains news files. The file names are descriptive of the contents of the files; they are analogous to headlines. When a user reads the news, using the **news** command, an empty file named **.news_time** is created in his or her login directory. The date (time) of this file is used by the **news** command to determine if a user has read the latest news file(s).

/usr/spool/cron/crontabs Directory

The **/usr/spool/cron/crontabs** directory contains crontab files for **adm**, **root**, and **sys** logins. Providing their lognames are in the **/usr/lib/cron/cron.allow** file, users can establish their own **crontabs** file using the **crontab** command. If the **cron.allow** file does not exist, the **/usr/lib/cron/cron.deny** file is checked to determine if the user is denied the use of the **crontab** command.

As **root**, you can either use the **crontab**(1) command or edit the appropriate file under **/usr/spool/cron/crontabs** to make the desired entries. Revisions to the file take effect at the next reboot. The line entry format of a **/usr/spool/cron/crontabs/***logname* file is as follows:

minute hour day month day-of-week command

where:

minute
is a one- or two-digit number; 0 through 59.

hour
is a one- or two-digit number; 0 through 24.

day
is the numerical day of the month; 1 through 31.

month
is the numerical month of the year; 1 through 12.

day-of-week
is the numerical day of the week where Sunday is 0, Monday is 1, . . . and Saturday is 6.

command
is the program or command that is executed at the time specified by the first five fields.

The following syntax applies to the first five fields:

- Two numbers separated by a minus indicates an inclusive range of numbers between the two specified numbers.
- A list of numbers separated by commas specifies all of the numbers listed.
- An asterisk specifies all legal values.

In the command field (sixth field), a percent sign (%) is translated to 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.

Figure B-15 shows a typical **/usr/spool/cron/crontabs/***logname* file. The data shown is the **root** file. The file entries support the **calendar** reminder service and basic networking. Remember, you can use the **cron** function to decrease the number of data terminal driven system administration tasks; include recurring and habitual tasks in your **crontab** file.

```
0 1 * * * /usr/bin/calendar -
41,11 * * * * /usr/lib/uucp/uudemon.hour > /dev/null
45 23 * * * ulimit 5000; /bin/su uucp -c
"/usr/lib/uucp/uudemon.cleanup"
  > /dev/null 2>&1
1,30 * * * * /usr/lib/uucp/uudemon.poll > /dev/null
```

Figure B-15. Typical **/usr/spool/cron/crontabs/root** File

(Refer to the **crontab**(1) manual page in the *User's Reference Manual* for additional information.)

C Error Messages

Error Messages

SYSTEM V/88 error messages are divided into three severity classes: **NOTICE, WARNING,** and **PANIC.** When an error message displays, its severity class displays as the first part of the message. The following operating system error message tables are divided into severity classes. A description of each severity class is given with each table.

NOTICE Messages

NOTICE error messages provide information on the system status. These messages may help you to anticipate problems before troubles occur. These error messages are defined alphabetically in the following table.

Error Message (Prefaced by NOTICE)	Description/Action
bn = # er = #, #	A device error occurred during a read/write operation. Log that the message occurred. No action is required unless the problem persists.
Can't allocate message buffer	System is out of message buffers. Either try again later, or send fewer interprocess communication messages.
Configured value of NOFILES (#) is greater than max (#) NOFILES set to #.	The value of NOFILES in **sysgen**s kernel file exceeds the allowed maximum. No immediate action required. Maximum value has been used. To avoid repetitions on future config boots, change **sysgen**s kernel file.

Error Message (Prefaced by NOTICE)	Description/Action
Configured value of NOFILES (#) is less than min (#) NOFILES set to #.	The value of NOFILES in **sysgen**s kernel file is less than the allowed minimum. No immediate action required. Minimum value has been used. To avoid repetitions on future config boots, change **sysgen**s kernel file.
contmemall - insufficient memory to allocate *xx* pages (system call failed)	Operating system did not have enough memory to run a process. Increase swap space.
contmemall - insufficient memory to lock *xx* pages (system call failed)	Operating system did not have enough memory to run a process. Increase physical memory.
Device Error bn = # er = #, #	Device error occurred during a read/write. Log that the error message occurred. Call your Service Representative if the problem persists.
dupreg - insufficient memory to allocate *xx* pages (system call failed)	Operating system did not have enough memory to run a process. Increase swap space.
dupreg - insufficient memory to lock *xx* pages (system call failed)	Operating system did not have enough memory to run a process. Increase physical memory.
File table overflow	The system file access table has overflowed. Reconfigure the system with larger NFILE turnable parameter if this is a frequent problem. Call your Service Representative if the problem continues.

Error Message (Prefaced by NOTICE)	Description/Action
getc pages - waiting for *nnn* contiguous pages	User process failed to allocate *nnn* contiguous pages. Run this user process immediately after booting the operating system.
growreg - insufficient memory to allocate *xx* pages (system call failed)	Operating system did not have enough memory to run a process. Increase swap space.
growreg - insufficient memory to lock *xx* pages (system call failed)	Operating system did not have enough memory to run a process. Increase physical memory.
iaddress > 2 24	Block number in an i-node was found to be greater than permissible when updating the file control block. Take the system to single-user mode (Procedure 3.3), and check the file system (Procedure 5.3). Call your Service Representative if you cannot resolve the problem.
MVME350_#: not online	
MVME350_#: write protected	
MVME350: DMA buffers discarded	
MVME350: DMA buffers still active	
MVME350: *str* on controller #, drive #	
MVME350: Unknown command #	
no space on floppy drive, slice #	The disk has reached its capacity.

Error Message (Prefaced by NOTICE)	Description/Action
no space on integral hard disk drive #, partition #	Clean up file system indicated by the partition number. Repartition if necessary.
Out of inodes on floppy drive, slice #	Backup the file system (Procedure 5.5). Run Procedure 5.2 increasing the number of inodes. (This destroys what is on the disk.)
Out of inodes on integral hard disk, drive #, partition #	Backup the file system (Procedure 5.5). Run Procedure 5.2 increasing the number of inodes. (This destroys what is on the disk)
page read error on floppy drive, slice #	Probable bad diskette. Try a different one. If the problem persists, call your Service Representative.
page read error on integral hard disk, partition #	An I/O error has occurred while trying to read. Probable fault in a page from a file. Take the system to single-user mode (Procedure 3.3) and run **sysadm badtrack**.
shmctl - couldn't lock # pages into memory	Memory overcommitted. Try again later.
sptmemall - insufficient memory to allocate *xx* pages (system call failed)	Operating system did not have enough memory to run a process. Increase swap space.
sptmemall - insufficient memory to lock *xx* pages (system call failed)	Operating system did not have enough memory to run a process. Increase physical memory.
stray interrupt at #	No action required.

Error Message (Prefaced by NOTICE)	Description/Action
str - Insufficient memory to allocate # pages - system call failed	*str* may be **growreg, exec, fork,** or other kernel function name. Try again later.
str - swpuse count overflow	More than 256 processes are sharing the same page of swap. A copy has been made. No action required.
str on MVME323: ctl #, drive #, vol #, slice #	
swapdel - too few free pages	An attempt to delete a swap file has failed because it would result in too little remaining space. No action required.
swap space running out: needed # blocks	The system had to do more work than normal to swap out a process. If this occurs frequently, try to run fewer simultaneous processes. The computer may hang in these circumstances.
useracc - couldn't lock page	Insufficient main memory available to lock a user data page in memory to service a read or write system call to a raw device. Reduce system load, reduce size of raw I/O buffer in user program, or add more memory.
System Halt Requested. System secured for powerdown.	Indicates a successful **sysadm powerdown** or **uadmin 2 0**. The system is idle (halted).
System Reboot Requested. System secured for RESET.	Indicates a successful **sysadm reboot** or **uadmin 2 1**. The system is idle (halted).
Return to Firmware Requested. System secured for RESET.	Indicates a successful **sysadm firmware** or **uadmin 2 2**. The system is idle (halted).

WARNING Messages

WARNING error messages indicate that the system may stop functioning if corrective action is not taken. These error messages are defined alphabetically in the following table.

Error Message (Prefaced by WARNING)	Description/Action
floppy disk timeout: work list flushed	Make sure diskette is in the drive and the door is closed. Contact your Service Representative if the problem occurs again.
iget - inode table overflow	Out of free slots in the inode table. Too many files open at once. Reduce number of simultaneous processes, or increase number of inode table entries.
inode table overflow	If recurring problem, reconfigure the system with a bigger i-node table (NINODE).
mfree map overflow #. Lost # items at #	If recurring problem, reconfigure the system with a larger core map size (SPTMAP).
No kernel virtual space. size= #, mode= #, base=#	If the problem persists, take dump and reboot.
*****STRAY INTERRUPT, ISR=#, IMR=#	No action required.

Error Message (Prefaced by WARNING)	Description/Action
out of swap space: needed # blocks	A process was left in memory because there was no room to swap it out. It will be swapped out if room becomes available. More room can be made by killing some processes. Run fewer simultaneous processes to avoid this problem. The system may hang in these circumstances.
Region table overflow	Each text, data, stack, and shmem process segment requires one entry in the region table. The system call that tried to allocate another region failed. Reduce the number of processes, or increase the number of region table overflows (NREGION).

PANIC Messages

PANIC error messages indicate a problem severe enough that the operating system must stop. The cause can be a hardware, software, or configuration problem.

NOTE

When the hardware is at fault, the PANIC error message does not always reflect the immediate problem.

The file system is updated (**sync**(1M) when a system PANIC occurs. However, you should still check it via **fsck**(1M) when the system is brought back up.

The System Administrator should keep a log of all PANICs seen on a system. The log should include a record of such things as special devices in use and special programs running when a panic occurs.

If a particular PANIC error message occurs repeatedly (or predictably), you should contact your Service Representative.

NOTE

After the panic completes, you can take a crash dump if you wish; then reboot the system. (See Procedure 3.6).

Error Message (Prefaced by PANIC)	Description/Action
assertion failed: *str*, file: *str* line: #	Your system has **debug** enabled. A debug assertion has failed, indicating something is wrong. Record the message; a standard release of SYSTEM V/88 does not have **debug** enabled.
blkdev	The system description file may be incorrect.
bumprcnt — region count list overflow	Ran out of region count entries.
cannot mount root	The root file system was corrupted or nonexistent when trying to boot. Reboot from the SYSTEM V/88 boot tape and attempt to **fsck** and mount the root file system. If reboot fails, do a partial restore from the core diskettes.
devtab	The list header for the chain of buffers attached to the block-type device cannot be found.

Error Message (Prefaced by PANIC)	Description/Action
dupreg - pbremove	Paging problem in the operating system. The kernel was trying to remove a page from the page cache but could not find it in the cache.
exec - bad magic	Internal operating system problem. May be caused by a bad object file reboot system. Log the error and contact your Service Representative.
getpages — pbremove	The kernel was attempting to remove a page from the page cache but could not find it in the cache. Probable software bug.
getxfile - bad magic	Internal operating system problem. May be caused by a bad object file reboot system. Log the error and contact your Service Representative.
iget — mounted on inode not in mount table.	Inode has mount flag set but not in mount table. Probable software bug.
i/o error in swap	An access error occurred on the swap device. Check the hard disk error log.
iput — bad mount count	The count of the number of inodes in use on a particular file system is incorrect. Probable software bug.
iupdat — fifo iaddress > 2 24	Block number in inode is too big.
iupdat — iaddress > 224	Block number in inode is too big.
kernel bus error system panic	SYSTEM V/88 took a bus error while running in system mode. Record message and all activities on system. Reboot.

Error Message (Prefaced by PANIC)	Description/Action
main — copyout of icode failed	The kernel was not able to copy the assembly code that is used to start up the system.
MVME323: Can't get cpages	
MVME350: Can't get cpages	
MVME350: null status	
main — swapadd failed	The kernel was not able to attach to the first swap area. Check to see if there is a swap area available on the boot disk.
newproc — fork failed	The kernel was not able to create one of the kernel processes while booting. Check tunables.
newproc — no procs	The kernel ran out of process table slots while creating kernel processes at boot time. Check value of NPROCS.
no procs	A process table entry was not found during a fork when an entry is available.
not a valid root	The root file system magic number is incorrect or the root device is improperly specified.
pinsert — pinsert dup	The kernel was attempting to add a page to the page cache but the page was found to be in the cache already. Probable kernel bug.
setrq — proc on q	The kernel was attempting to add a process to the run queue but the process was found to be on the queue already.

Error Message (Prefaced by PANIC)	Description/Action
srmount — cannot mount root	The kernel was not able to mount the root file system when booting. Check the VTOC on the disk.
srmount — not a valid root	When the root file system was being mounted during the kernel boot, it did not contain the correct "magic number". Check VTOC. Boot off another disk and run **fsck**(1M) on root.
swapseg — i/o error in swap	An I/O error occurred during a transfer to or from the swap area.
Timeout table overflow	The queue for timeout requests has overflowed while attempting to add another entry.
unknown level in cmn_err (level=#, msg = ...)	The common error software was invoked to process an error, but given an invalid severity level. This problem is secondary; the original problem is given by the msg.
vfault — bad dbd_type	The page being faulted is not a recognized type (? demand fill, demand zero, in file or on swap).
xalloc — bad magic	An invalid magic number was found in an a.out header during an exec system call. This should have been detected earlier by the kernel, so this is a probable kernel bug.

Call Error Messages

A system call that is unsuccessful returns an impossible value to the calling process. This impossible value is almost always a -1. When a system call is successful, a value of 0 is returned to the calling process. Any time a system call is unsuccessful, an external variable called *errno* is set to one of the numbers in the following table.

When a -1 value is returned, the *errno* variable contains the number corresponding to the reason of the failure. The *errno* variable is only valid immediately after a system call failure. It is not cleared on successful system calls. These error numbers are defined in the **/usr/include/sys/errno.h** header file.

Error		Description
Number	**Name**	
1	EPERM	**Not Owner.** Typically this error indicates an attempt to modify a file in some way forbidden except to its owner or superuser. It is also returned for attempts by ordinary users to do things allowed only to the superuser.
2	ENOENT	**No such file or directory.** This error occurs when a file name is specified and the file should exist but does not, or when one of the directories in a path name does not exist.
3	ESRCH	**No such process.** No process can be found corresponding to that specified by **pid** in **kill** or **ptrace**.

Error		Description
Number	Name	
4	EINTR	**Interrupted system call.** An asynchronous signal (e.g., interrupt or quit), which the user has elected to catch, occurred during a system call. If execution is resumed after processing the signal, it appears as if the interrupted system call returned this error condition.
5	EIO	**I/O error.** Some physical I/O error. This error may in some cases occur on a call following the one to which it actually applies.
6	ENXIO	**No such device or address.** I/O on a special file refers to a subdevice that does not exist, or beyond the limits of the device. It may also occur when, for example, a tape drive is not online or no disk pack is loaded on a drive.
7	E2BIG	**Arg list too long.** An argument list longer than 5,120 bytes is presented to a member of the *exec* family.
8	ENOEXEC	**Exec format error.** A request is made to execute a file which, although it has the appropriate permissions, does not start with a valid magic number (see **a.out**(4)).
9	EBADF	**Bad file number.** Either a file descriptor refers to no open file, or a read (respectively write) request is made to a file which is open only for writing (respectively reading).
10	ECHILD	**No child processes.** A **wait,** was executed by a process that had no existing or unwaited-for child processes.

Error		Description
Number	Name	
11	EAGAIN	**No more processes.** A **fork** failed because the systems process table is full or the user is not allowed to create any more processes.
12	ENOMEM	**Not enough space.** During an **exec, brk,** or **sbrk,** a program asks for more space than the system is able to supply. This is not a temporary condition; the maximum space size is a system parameter. The error may also occur if the arrangement of text, data, and stack segments requires too many segmentation registers, or if there is not enough swap space during a **fork.**
13	EACCES	**Permission denied.** An attempt was made to access a file in a way forbidden by the protection system.
14	EFAULT	**Bad address.** The system encountered a hardware fault in attempting to use an argument of a system call.
15	ENOTBLK	**Block device required.** A non-block file was mentioned where a block device was required, e.g., in **mount.**
16	EBUSY	**Mount device busy.** An attempt to mount a device that was already mounted or an attempt was made to dismount a device on which there is an active file (open file, current directory, mounted-on file, active text segment). It also occurs if an attempt is made to enable accounting when it is already enabled.
17	EEXIST	**File exists.** An existing file was mentioned in an inappropriate context, e.g., **link.**
18	EXDEV	**Cross-device link.** A link to a file on another device was attempted.

Error		Description
Number	Name	
19	ENODEV	**No such device.** An attempt was made to apply an inappropriate system call to a device; e.g., read a write-only device.
20	ENOTDIR	**Not a directory.** A non-directory was specified where a directory is required, for example in a path prefix or as an argument to **chdir**(2).
21	EISDIR	**Is a directory.** An attempt to write on a directory.
22	EINVAL	**Invalid argument.** Some invalid argument (e.g., dismounting a non-mounted device; mentioning an undefined signal in **signal**, or **kill**, reading or writing a file for which **lseek** has generated a negative pointer). Also set by the math functions described in the (3M) entries of the *Programmer's Reference Manual.*
23	ENFILE	**File table overflow.** The systems table of open files is full, and temporarily no more **open**s can be accepted.
24	EMFILE	**Too many open files.** No process may have more than 20 file descriptors open at a time.
25	ENOTTY	**Not a typewriter.**
26	ETXTBSY	**Text file busy.** An attempt to execute a pure-procedure program that is currently open for writing (or reading). Also, an attempt to open for writing a pure-procedure program that is being executed.
27	EFBIG	**File too large.** The size of a file exceeded the maximum file size (1,082,201,088 bytes) or ULIMIT; see **ulimit**(2).

Error		Description
Number	Name	
28	ENOSPC	**No space left on device.** During a write to an ordinary file, there is no free space left on the device.
29	ESPIPE	**Illegal seek.** An **lseek** was issued to a pipe.
30	EROFS	**Read-only file system.** An attempt to modify a file or directory was made on a device mounted read-only.
31	EMLINK	**Too many links.** An attempt to make more than the maximum number of links (1000) to a file.
32	EPIPE	**Broken pipe.** A write on a pipe for which there is no process to read the data. This condition normally generates a signal; the error is returned if the signal is ignored.
33	EDOM	**Math argument.** The argument of a function in the math library (3M) is out of the domain of the function.
34	ERANGE	**Result too large.** The value of a function in the math package (3M) is not representable within machine precision.
35	ENOMSG	**No message of desired type.** An attempt was made to receive a message of a type that does not exist on the specified message queue (see **msgop**(2)).
36	EIDRM	**Identifier removed.** This error is returned to processes that resume execution due to the removal of an identifier from the file system's name space (see **msgct**(2), **semctl**(2), and **shmctl**(2)).
37	ECHRNG	**Channel number out of range.**
38	EL2NSYNC	**Level 2 not synchronized.**
39	EL3HLT	**Level 3 halted.**

Error		Description
Number	Name	
40	EL3RST	Level 3 reset.
41	ELNRNB	Link number out of range.
42	EUNATCH	Protocol driver not attached.
43	ENOCSI	No CSI structure available.
44	EL2HLT	Level 2 halted.
45	EWOULDBLOCK	Operation would block.
46	ENOLCK	No record locks available.
50	EBADE	Invalid exchange.
51	EBADR	Invalid request descriptor.
52	EXFULL	Exchange full.
53	ENOANO	No anode.
54	EBADRQC	Invalid request code.
55	EBADSLT	Invalid slot.
56	EDEADLOCK	File-locking deadlock error.
57	EBFONT	Bad font file fmt.
60	ENOSTR	Device not a stream.
61	ENODATA	No data (for no-delay i/o).
62	ETIME	Timer expired.
63	ENOSR	Out of streams resources.
64	ENONET	Machine is not on network.
65	ENOPKG	Package not installed.
66	EREMOTE	The object is remote.
67	ENOLINK	The link has been severed.

Error		Description
Number	Name	
68	EADV	Advertise error.
69	ESRMNT	Srmount error.
70	ECOMM	Communication error on send.
71	EPROTO	Protocol error.
74	EMULTIHOP	Multihop attempted.
75	ELBIN	Inode is remote (not really an error).
76	EDOTDOT	Cross mount point (not really error).
77	EBADMSG	Trying to read unreadable message.
78	ENAMETOOLONG	Filename too long.
80	ENOTUNIQ	Given log.name not unique.
81	EBADFD	F. d. invalid for this operation.
82	EREMCHG	Remote address changed.
83	ELIBACC	Cannot access a needed shared lib.
84	ELIBBAD	Accessing a corrupted shared lib.
85	ELIBSCN	The .lib section in a.out corrupted.
86	ELIBMAX	Attempting to link in too many libs.
87	ELIBEXEC	Attempting to exec a shared lib.
89	ENOSYS	System call not supported.
90	ELOOP	Too many levels of symbolic links.
91	ERESTART	Unsupported file system operation.
128	EINPROGRESS	Operation now in progress.
129	EALREADY	Operation already in progress.
130	ENOTSOCK	Socket operation on non-socket.
131	EDESTADDRREQ	Destination address required.

Error		Description
Number	Name	
132	EMSGSIZE	**Message too long.**
133	EPROTOTYPE	**Protocol wrong type for socket.**
134	ENOPROTOOPT	**Protocol not available.**
135	EPROTONOSUPPORT	**Protocol not supported.**
136	ESOCKTNOSUPPORT	**Socket type not supported.**
137	EOPNOTSUPP	**Operation not supported on socket.**
138	EPFNOSUPPORT	**Protocol family not supported.**
139	EAFNOSUPPORT	**Protocol family does not support address family.**
140	EADDRINUSE	**Address already in use.**
141	EADDRNOTAVAIL	**Cannot assign requested address.**
142	ENETDOWN	**Network is down.**
143	ENETUNREACH	**Network is unreachable.**
144	ENETRESET	**Network dropped connection on reset.**
145	ECONNABORTED	**Software caused connection abort.**
146	ECONNRESET	**Connection reset by peer.**
147	ENOBUFS	**No buffer space available.**
148	EISCONN	**Socket is already connected.**
149	ENOTCONN	**Socket is not connected.**
150	ESHUTDOWN	**Cannot send after socket shutdown.**
151	ETOOMANYREFS	**Too many references: cannot splice.**
152	ETIMEDOUT	**Connection timed out.**
153	ECONNREFUSED	**Connection refused.**
158	ENOTEMPTY	**Directory not empty.**

Firmware Error Messages

The firmware mode is the state your computer must be in for you to interface with several software programs. If a problem occurs while in this state, a firmware error message displays on the console terminal.

LP Print Service Error Messages

This section provides a description of the error messages that are associated with LP commands. The following variables are used in the error messages:

file(s)
Indicates the file or files that are to be printed.

dest
Indicates the name of the destination printer.

printer-id
Indicates the request identification number of the printout. For example, *dqp10_2-46* is the printer name followed by the request identification number.

printer-name
Indicates the name of the printer.

program-name
Indicates the program name that was executed.

user
Indicates the user who requested the printout.

Following each message is an explanation of the probable cause of the error and the corrective action to take. If you are not able to correct all the error conditions you encounter, call your Service Representative for assistance.

Some lengthy error messages that appear all on one line on the display are too long to be shown as one line in the documentation. When more than one line is required in the documentation to show a one line error message, a "\" is used to split the message.

Error Message	Description/Action
dest is an illegal destination name	The *dest* you used is not a valid destination name. Use the **lpstat -p** command to list valid destination names.
file is a directory	The file name you typed is a directory and cannot be printed.
xx is not a request id or a printer	The argument you used with the **cancel** command is not a valid request identification number or a printer name. Use the **lpstat -t** command to give you all the printers and requests waiting to get printed.
xx is not a request id	The request identification number you used with the **lpmove** command is not a valid request identification number. To find out what requests are valid, use the **lpstat -u** command.
xx not a request id or a destination	You used an invalid request identification number or destination with the **lpstat** command. To find out what is valid, use the **lpstat -t** command.
dest not accepting requests since *date*	Requests to the printer that you are trying to use have been stopped by the **reject** command.
Can't access FIFO	The named pipe file **/usr/spool/lp/FIFO** is incorrect. The mode should be 600.
LP Administrator not in password file	You must have an entry in the **/etc/passwd** file for "lp," and you must belong to the group "bin."
acceptance status of destination "printer-name" unknown	Use the **accept** command to enable the printer so it will accept requests.
can't access file "xx"	The mode could be wrong on your directory or the file that you are trying to access.

Error Message	Description/Action
can't create class "xx"-it is an existing printer name	The class name you are trying to use has already been given to a printer. You will have to use another name or remove the printer to use the class name.
can't create new acceptance status file	The mode may be wrong on the **/usr/spool/lp** directory. It should be 755 with the owner "lp" and the group "bin."
can't create new class file	The mode may be wrong on the **/usr/spool/lp** directory. It should be 755 with the owner "lp" and the group "bin."
can't create new interface program	The mode may be wrong on the **/usr/spool/lp/interface** directory. It should be 755 with the owner "lp" and the group "bin."
can't create new member file	The mode may be wrong on the **/usr/spool/lp/member** directory. It should be 755 with the owner "lp" and the group "bin."
can't create new printer status file	The mode may be wrong on the **/usr/spool/lp/pstatus**. It should be 644 with the owner "lp" and the group "bin."
can't create new request directory	The mode may be wrong on the **/usr/spool/lp/request** directory. It should be 755 with the owner "lp" and the group "bin."
can't create printer "printer-name" -- it is an existing class name	The printer-name you are trying to use has already been used as a class name. You will have to assign another name for the printer.

Error Message	Description/Action
can't create new output queue	The mode on the file **/usr/spool/lp/seqfile** is incorrect. It should be 644, and the mode on the directory should be 755. The owner should be "lp," and the group should be "bin." This may be corrected by typing the command at a later time.
can't create new sequence number file	The mode on the file **/usr/spool/lp/seqfile** is incorrect. The mode of the file should be 644, and the mode of the directory should be 755. The owner should be "lp," and the group should be "bin." This may be corrected by typing the command at a later time.
can't create request file "xx"	The mode on the file **/usr/spool/lp/request/***printer-name*/r-id is incorrect. *Printer-name* is the name of the printer e.g., dqp10, and r-id is the request identification number. The mode of the file should be 444, and the mode of the directory should be 755. The owner should be "lp," and the group should be "bin." This may be corrected by typing the command at a later time.
can't fork	You either have several processes running and are not allowed to run anymore, or the system has all the processes running that it can handle. You will have to rerun this command later.
can't lock acceptance status	This is a temporary file in **/usr/spool/lp** that prevents more than one "lp" request from being taken at any given instant. You must rerun this command later.

Error Message	Description/Action
can't lock output queue	The file **/usr/spool/lp/QSTATLOCK** prevents more than one "lp" request from being printed on a printer at a time. You will have to rerun this command later.
can't lock printer status	The temporary file **/usr/spool/lp/PSTATLOCK** prevents more than one "lp" request from being printed on a printer at a time. You must rerun this command later.
can't lock sequence number file	The file **/usr/spool/lp/SEQLOCK** prevents more than one "lp" request from getting the next printer-id (request identification) number at a time. You must rerun this command later.
can't move request printer-id	*Printer-id* is the request identification number that cannot be moved. You will probably have to change the modes on the files and directories in **/usr/spool/lp/request**. Also, you will have to manually move the request from the disabled printer directory to the new destination after you shut down the LP scheduler.
can't open class file	The **lp** program is trying to access the list of classes for printers. One reason it may not be able to open the class file is that the system could have the maximum number of files open that are allowed at any time. This can be corrected by typing the command at a later time.

Error Message	Description/Action
can't open member file	The **lp** program is trying to access the list of members in the directory **/usr/spool/lp /member**. The system could have the maximum number of files open that are allowed at any time. This can be corrected by typing the command at a later time.
can't open xx file in MEMBER directory	There are a couple of reasons why file *xx* in the **/usr/spool/lp /member** directory cannot be opened. The mode on the file could be incorrect. It should be 644. Another possibility is that the system could have the maximum number of files open that are allowed at any time. This can be corrected by typing the command at a later time.
can't open xx file in class directory	One possibility why file *xx* cannot be opened is that the mode on the file or directory is incorrect. The file mode should be 644, and the directory mode should be 755. Another possibility is that the system has the maximum number of files open that are allowed at any time. The latter problem can be corrected by typing the command at a later time.
can't open xx	You cannot print on printer *xx* because the mode is incorrect on **/dev/tty**. The mode should be 622.

Error Message	Description/Action
can't open FIFO	The mode on the named pipe file **/usr/spool/lp/FIFO** may be incorrect. It should be 600. Or, the system could have the maximum number of files open that are allowed at any time. The latter problem can be corrected by typing the command at a later time.
can't open MEMBER directory	The mode on the directory **/usr/spool/lp/member** could be incorrect. It should be 755. Another possibility is that the system could have the maximum number of files open that are allowed at any time. If the maximum number of files are open, try typing the command at a later time.
can't open acceptance status file	The mode on the file **/usr/spool/lp/qstatus** may not be correct. It should be 644. Another possibility is that the system could have the maximum number of files open that are allowed at any time. The latter problem can be corrected by typing the command at a later time.
can't open default destination file	Check the mode on the file **/usr/spool/lp/default**. The mode should be 644. If the mode is okay, it could be that the system has the maximum number of files open that are allowed at any one time. This can be corrected by trying the command at a later time.

Error Message	Description/Action
can't open file filename	The *filename* was incorrectly typed or you don't have the correct modes set. The mode should be at least 400 if you are the owner.
can't open output queue file	Check the mode on the file **/usr/spool/lp/outputq**. It should be 644. This error message could also be generated if the system has the maximum number of files open that are allowed at any one time. Try entering the command at a later time.
can't open printer status file	The mode on the file **/usr/spool/lp/pstatus** is incorrect. The mode should be 644. It could also be that the system has the maximum number of files open that are allowed at any one time. This can be corrected by trying the command at a later time.
can't open request directory directory name	The mode on the directory **/usr/spool/lp/request** is incorrect. The mode should be 655. It could also be that the system has the maximum number of files open that are allowed at any one time. This can be corrected by trying the command at a later time.
can't open request file xx	The mode on the file **/usr/spool/lp/member/request/***xx* is incorrect. The mode should be 644. It could also be that the system has the maximum number of files open that are allowed at any one time. This can be corrected by trying the **lpmove** command at a later time.

Error Message	Description/Action
can't open system default destination file	The mode on the file **/usr/spool/lp/default** is incorrect. The mode should be 644. It could also be that the system has the maximum number of files open that are allowed at any one time. This can be corrected by trying the command again at a later time.
can't open temporary output queue	The mode on the file **/usr/spool/lp/outputq** is incorrect. The mode should be 644. It could also be that the system has the maximum number of files open that are allowed at any one time. This can be corrected by trying the command at a later time.
can't proceed -- scheduler running	Many of the **lpadmin** command options cannot be executed while the scheduler is running. Stop the scheduler using the **lpshut** command and invoking the command again.
can't read current directory	The **lp** and **lpadmin** commands cannot read the directory containing the file to be printed. The directory name may be incorrect or you do not have read permission on that directory.
can't remove class file	The mode may be wrong on the **/usr/spool/lp/class**. It should be 755. The owner should be "lp," and the group should be "bin." Another possibility is the file in that directory may have the wrong mode. It should be 644.

Error Message	Description/Action
can't remove printer	The mode may be wrong on the **/usr/spool/lp/member** directory. It should be 755, and the files in that directory should be 644. Both the directory and the files should be owned by "lp," and the group should be "bin."
can't remove request directory	The mode may be wrong on the **/usr/spool/lp/request** directory. It should be 755 and should be owned by "lp," and the group should be "bin." The directory may still have pending requests to be printed that must be removed before the directory can be removed.
can't set user id to LP Administrator's user id	The **lpsched** and **lpadmin** commands can only be used when you are logged in as "lp" or "root."
can't unlink old output queue	The **lpsched** program cannot remove the old output queue. You must remove it manually by using the command **rm/usr/spool/lp/outputq**.
can't write to xx	The **lpadmin** command cannot write to device *xx*. The mode is probably wrong on the /dev/ttyxx file. It should be 622 and owned by "lp."
cannot create temp file filename	The system may be out of free space on the usr file system. Use the command **df /usr** to determine the number of free blocks. Several hundred blocks are required to insure that the system performs correctly.

Error Message	Description/Action
class "xx" has disappeared!	Class *xx* was probably removed since the scheduler was started. The system may be out of free space on the usr file system. Use the command **df /usr** to find out. Use the **lpshut** command to stop the scheduler and restore the class from a backup.
class "xx" non-existent	The class *xx* may have been removed because the system is out of free space on the usr file system. Use the command **df /usr** to find out how much free space is available. The class probably has to be restored from a backup.
class directory has disappeared!	The **/usr/spool/lp/class** directory has been removed. The system may be out of free space on **/usr**; use the **df /usr** command to find out. The class directory contains all the data for each printer class. To restore this directory, get these files and directory from a backup.
corrupted member file	The **/usr/spool/lp/member** directory has a corrupted file in it. You should restore the directory from backup.
default destination "dest" non-existent	Either the default destination is not assigned or the printer *dest* has been removed. Use the **lpadmin** to set up a default destination or set LPDEST to the value of the destination.

Error Message	Description/Action
destination "dest"has disappeared!	A destination printer, *dest* has been removed since **lpsched** was started. Use the **lpadmin** command to remove the printer.
destination "printer-name" is no longer accepting requests	The printer has been disabled using the **reject** command. The printer can be reenabled using the **accept** command.
destination dest non-existent	The destination printer you specified as an argument to the **accept** or **lpadmin** command is not a valid destination name, or it has been removed since the scheduler was started.
destination "printer-name" was already accepting requests	The destination printer was previously "enabled." Once a printer is accepting requests, issuing any more **accept** commands to it are ignored.
destination "printer-name" was already not accepting requests	A **reject** command was already sent to the printer. Use the **accept** command to allow the printer to start accepting requests again.
destination printer-name is not accepting requests move in progress ...	The printer has been disabled by the **reject** command, and requests are being moved from the disabled printer to another printer. The printer can be enabled again by the **accept** command.
destinations are identical	When using the **lpmove** command, you must specify a printer to move the print requests from and a different printer to move the requests to.

Error Message	Description/Action
disabled by scheduler: login terminal	The login terminal has been disabled by the LP scheduler. The printer can be reenabled by using the **enable** command.
error in printer request printer-id	*Printer-id* is the actual request identification number. The error was most likely due to an error in the printer. Check the printer, and reset it if needed.
illegal keyletter "xx"	An invalid option, *xx*, was used. See the manual page for the correct options.
keyletters "-xx" and "-yy" are contradictory	This combination of options to the **lpadmin** program cannot be used together.
keyletter "xx" requires a value	The option *xx* requires an argument. For example, in the following command line the argument to the **-m** option is the name of a model interface program: **lpadmin -m** *model*
keyletters -e, -i, and -m are mutually exclusive	These options to the **lpadmin**(1M) command cannot be used together. Refer to the manual page for more information on usage.
lp: xx	In this message the variable *xx* could be one of several arguments. Typically, it is a message telling you the default destination is not assigned.
member directory has disappeared!	The **/usr/spool/lp/member** directory has been removed. The system is probably out of free disk space in the usr file system. You need to clean up the usr file system, and then install the LP commands or retrieve them from a backup.

Error Message	Description/Action
model "xx" non-existent	The name that you are using for a model interface program is not a valid one. A list of valid models is in the **/usr/spool/lp/model** directory.
new printers require -v and either -e, -i, or -m	A printer must have an interface program, and this is specified by **-e, -i,** or **-m** options. The **-v** option specifies the device file for the printer. For more information on these options, refer to the **lpadmin**(1M) manual page.
no destinations specified	There are no destination printers specified. Use the **lpadmin** command to set one up.
no printers specified	There are no printers specified. Use the **lpadmin** command to set one up.
non-existent printer xx in PSTATUS	A printer with the name *xx* is in the **/usr/spool/lp/pstatus** file but no longer exists. The printer should be removed using the **lpadmin** command.
non-existent printer printer-name in class xx	The printer that you are trying to address in class *xx* has been removed from that class.
out of memory	Implies the system is in trouble. The message implies that there is not enough memory to contain the text to be printed.
printer "printer-name" already in class "xxP"	The printer you are trying to move to class *xx* is already in that class. You cannot move a printer to a class that it is already in.
printer "printer-name" has disappeared! or printer "printer-name" has disappeared	The printer has been removed and the **enable** command cannot find it. The printer was most likely removed since the machine was rebooted or since the scheduler was started.

Error Message	Description/Action
printer "printer-name" non-existent	*Printer-name* is the name of a printer that has been removed since the scheduler has been started. You must use the **lpadmin -x***printer-name*.
printer status entry for "printer-name" has disappeared	The **/usr/spool/lp/pstatus** file must have been corrupted. You will have to resubmit the printer request.
printer "printer-name" was not busy	The printer is not printing a request at this time. Either the request you wanted to cancel is finished printing or you have specified the wrong printer.
request "printer-id" non-existent	You are attempting to cancel a request that does not exist. You may have given the wrong printer name or wrong request id number or the request may have finished printing.
request not accepted	The request was not accepted by **lp**. The scheduler may not be running. Use the **lpstat -t** command to find out more information.
requests still queued for "printer-name" -- use lpmove	*Printer-name* is the printer that still has requests waiting to get printed. You need to use the **lpmove** command to get those requests moved to another printer.
scheduler is still running -- can't proceed	You cannot perform this command while the scheduler is running. You must use the **lpshut** command first.
spool directory non-existent	The directory **/usr/spool** has been removed. You must use the **mkdir** command to restore the directory. This has probably removed some of the necessary LP files. You must reinstall the LP commands.

Error Message	Description/Action
standard input is empty	You specified an invalid file name either by incorrectly typing a name or by specifying a nonexistent file. Nothing is printed on the printers from this request.
this command for use only by LP Administrators	This command is restricted to someone logged in as root or lp.
too many options for interface program	The **lp** command called the appropriate interface program with too many arguments. For more information on the options and arguments that can be used with the **lp** command, see the **lp** manual page.
unknown keyletter "xx" or unknown keyletter "-xx"	An invalid option was supplied to the **lp** or **lpadmin** commands. See the manual pages for all the correct usages.
unknown option "xx"	This message displays in response to an invalid option supplied to the **disable**, **lpstat**, or **reject** commands. See the manual pages for all the correct usages.
usage: disable [-c] [-r[reason]] printer	The syntax for the **disable** command is not correct. The valid options are: **-c** to cancel the currently printing request, and **-r** followed by the reason that you are disabling the printer.
usage: reject [-r[reason]] dest ...	The syntax for the **reject** command is not correct. The proper format is to specify the reason why the printer is not taking any more print requests and to identify the destination printer.

Error Message	Description/Action
usage: accept dest	The syntax for the **accept** command is to specify a destination printer. You are setting up a printer to accept requests, and you did not specify what printer should accept requests.
usage: enable printer	The syntax for the **enable** program is to specify a destination printer.
usage: cancel id .... printer ...	The syntax for the **cancel** command is not correct. The proper format is to specify the request identification number or the printer name.
usages: lpadmin -pprinter [-vdevice] [-cclass] [-rclass] [-eprinter | -iinterface | -mmodel] [-h | -l] -or- lpadmin -d[destination] -or- lpadmin -xdestination	The correct syntax for the **lpadmin** command is to specify at least one of the options mentioned above.
your printer request printer-id was canceled by user	The printer request did not finish printing because another *user* cancelled it. Typically, you get this message in your mail. One reason a person may cancel a request other than their own is because the request is not printing correctly.

Basic Networking Utilities Error Messages

This section lists the error messages associated with Basic Networking Utilities. There are two types of error messages. ASSERT errors are recorded in the **/usr/spool/uucp/.Admin/errors** file. STATUS errors are recorded in individual machine files found in the **/usr/spool/uucp/.Status** directory.

BNU ASSERT Error Messages

When a process is aborted, ASSERT error messages are recorded in **/usr/spool/uucp/.Admin/errors**. These messages include the file name, sccsid, line number, and the text listed below. In most cases, these errors are the result of file system problems. The "errno" (when present) should be used to investigate the problem. If "errno" is present in a message, it is shown as () in the following list.

Error Message	Description/Action
CAN'T OPEN	An **open**() or **fopen**() failed.
CAN'T WRITE	A **write**(), **fwrite**(), **fprint**(), failed.
CAN'T READ	A **read**(), **fgets**(), etc. failed.
CAN'T CREATE	A **create**() call failed.
CAN'T ALLOCATE	A dynamic allocation failed.
CAN'T LOCK	An attempt to make a LCK (lock) file failed. In some cases, this is a fatal error.
CAN'T STAT	A **stat**() call failed.
CAN'T CHMOD	A **chmod**() call failed.
CAN'T LINK	A **link**() call failed.
CAN'T CHDIR	A **chdir**() call failed.
CAN'T UNLINK	A **unlink**() call failed.

Error Message	Description/Action
WRONG ROLE	This is an internal logic problem.
CAN'T MOVE TO CORRUPTDIR	An attempt to move some bad C. or X. files to the **/usr/spool/uucp/.Corrupt** directory failed. The directory is probably missing or has wrong modes or owner.
CAN'T CLOSE	A **close**() or **fclose**() call failed.
FILE EXISTS	The creation of a C. or D. file is attempted, but the file exists. This occurs when there is a problem with the sequence file access. Usually indicates a software error.
No uucp server	A tcp/ip call is attempted, but there is no server for UUCP.
BAD UID	The uid cannot be found in the **/etc/passwd** file. The file system is in trouble, or the **/etc/passwd** file is inconsistent.
BAD LOGIN_UID	Same as previous.
ULIMIT TOO SMALL	The ulimit for the current user process is too small. File transfers may fail, so transfer is not attempted.
BAD LINE	There is a bad line in the **Devices** file; there are not enough arguments on one or more lines.
FSTAT FAILED IN EWRDATA	There is something wrong with the ethernet media.
SYSLST OVERFLOW	An internal table in gename.c overflowed. A big/strange request was attempted. Contact your Service Representative.
TOO MANY SAVED C FILES	Same as previous.

Error Message	Description/Action
RETURN FROM fixline ioctl	An **ioctl**, which should never fail, failed. There is a system driver problem.
BAD SPEED	A bad line speed appears in the **Devices/Systems** files (Class field).
PERMISSIONS file: BAD OPTION	There is a bad line or option in the **Permissions** file. Fix it immediately!
PKCGET READ	The remote machine probably hung up. No action.
PKXSTART	The remote machine aborted in a non-recoverable way. This can generally be ignored.
SYSTAT OPEN FAIL	There is a problem with the modes of **/usr/lib/uucp/.Status**, or there is a file with bad modes in the directory.
TOO MANY LOCKS	There is an internal problem! Contact your Service Representative.
XMV ERROR	There is a problem with some file or directory. It is likely the spool directory, since the modes of the destinations were suppose to be checked before this process was attempted.
CAN'T FORK	An attempt to fork and exec failed. The current job should not be lost, but will be attempted later (**uuxqt**). No action need be taken.

BNU STATUS Error Messages

STATUS error messages are messages that are stored in the **/usr/spool/uucp/.Status** directory. This directory contains a separate file for each remote machine that your computer attempts to communicate with. These individual machine files contain status information on the attempted communication, whether it was successful or not. What follows is a list of the most common error messages that may appear in these files.

Error Message	Description/Action
OK	Things are OK.
NO DEVICES AVAILABLE	There is currently no device available for the call. Check to see that there is a valid device in the **Devices** file for the particular system. Check the **Systems** file for the device to be used to call the system.
WRONG TIME TO CALL	A call was placed to the system at a time other than what is specified in the **Systems** file.
TALKING	Self explanatory.
LOGIN FAILED	The login for the given machine failed. It could be a wrong login/password, wrong number, a very slow machine, or failure in getting through the Dialer-Token-Pairs script.
CONVERSATION FAILED	The conversation failed after successful startup. This usually means that one side went down, the program aborted, or the line (link) was dropped.
DIAL FAILED	The remote machine never answered. It could be a bad dialer or the wrong phone number.

Error Message	Description/Action
BAD LOGIN/MACHINE COMBINATION	The machine called us with a login/machine name that does not agree with the **Permissions** file. This could be an attempt to masquerade!
DEVICE LOCKED	The calling device to be used is currently locked and in use by another process.
ASSERT ERROR	An ASSERT error occurred. Check the **/usr/spool/uucp/.Admin/errors** file for the error message and see the section *BNU ASSERT Error Messages*.
SYSTEM NOT IN Systems	The system is not in the **Systems** file.
CAN'T ACCESS DEVICE	The device tried does not exist or the modes are wrong. Check the appropriate entries in the **Systems** and **Devices** files.
DEVICE FAILED	The open of the device failed.
WRONG MACHINE NAME	The called machine is reporting a different name than expected.
CALLBACK REQUIRED	The called machine requires that it calls your computer.
REMOTE HAS A LCK FILE FOR ME	The remote site has a LCK file for your computer. They could be trying to call your machine. If they have an older version of Basic Networking, the process that was talking to your machine may have failed leaving the LCK file. If they have the new version of Basic Networking, and they are not communicating with your computer, then the process that has a LCK file is hung.

Error Message	Description/Action
REMOTE DOES NOT KNOW ME	The remote machine does not have the node name of your computer in its **Systems** file.
REMOTE REJECT AFTER LOGIN	The login used by your computer to login does not agree with what the remote machine was expecting.
REMOTE REJECT, UNKNOWN MESSAGE	The remote machine rejected the communication with your computer for an unknown reason. The remote machine may not be running a standard version of Basic Networking.
STARTUP FAILED	Login succeeded, but initial handshake failed.
CALLER SCRIPT FAILED	This is usually the same as "DIAL FAILED." However, if it occurs often, suspect the caller script in the **dialers** file. Use **uutry** to check.

D System Administration Menu Package

System Administration Main Menu

The System Administration Menu Package, also described in **sysadm**(1), provides a menu interface to perform system administration commands. Typing **sysadm** without an argument causes the computer to display the menu of system administration subcommands shown in Figure D-1.

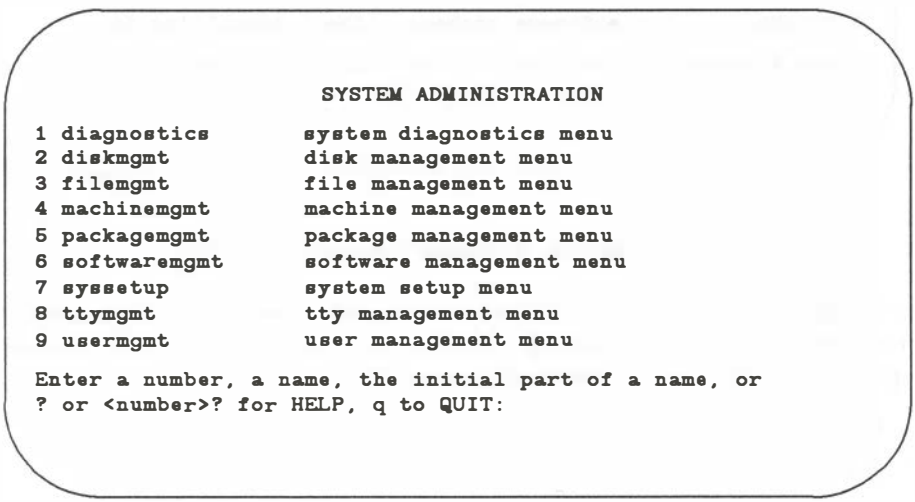


Figure D-1. System Administration Main Menu

As the main menu indicates, typing **q** <**CR**> returns you to the shell and **?** <**CR**> displays a help screen.

To access one of the submenus listed in Figure D-1, enter the menu item number, the menu name, or the initial part of the menu name followed by <**CR**>. If, for example, you want to install a new software package, type any one of the following responses:

6 <**CR**>

softwaremgmt <**CR**>

software <**CR**>

The Software Management menu then displays:

```
                    SOFTWARE MANAGEMENT

1 installpkg   install new software package onto built-in disk
2 listpkg      list packages already installed
3 removepkg    remove previously installed package from built-in
               disk
4 runpkg       run software package without installing it

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

Figure D-2. Software Management Menu

The **installpkg** subcommand will help you install a new software package. To enter the interactive **installpkg** routine, type any one of the following responses and follow the instructions provided.

`1 <CR>`

`installpkg <CR>`

`install <CR>`

System Administration Submenus

The following section briefly describes the nine system administration submenus. For details, see **sysadm**(1).

System Diagnostics Menu

The System Diagnostics Menu contains a subcommand that issues reports and allows you to determine if there are detectable problems and a submenu which lets you attempt repairs of the disk. See **sysadm**(1) for details.

```
                    SYSTEM DIAGNOSTICS

1 diskrepair        disk repair
2 diskreport        report on built-in disk errors

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

Typing **1** (diskrepair) causes a disk repair submenu to display. Currently the only disk repair option supported is bad track redirection.

```
                      DISK REPAIR

1 Badtrack        BAD TRACK HANDLING

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

Typing **1** (Badtrack) causes a track redirection submenu to display. This menu contains subcommands which perform the various aspects of track redirection.

```
                    BAD TRACK HANDLING

1 delfsckfiles    clear all fsck input files
2 fixfsys         run fsck using fsck input files
3 redirect        redirect bad tracks on the disk
4 updbadlist      update the bad track file with the correct list

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

Disk Management Menu

The subcommands in this menu provide functions for using removable disks. With these subcommands you can format and copy disks and use disks as mountable file systems. See **sysadm**(1) for details.

```
                    DISK MANAGEMENT

1 checkfsys     check a removable medium file system for errors
2 cpdisk        make exact copies of a removable medium
3 erase         erase data from removable media
4 format        format new removable diskettes
5 hdsetup       hard disk set up
6 makefsys      create a new file system on a removable medium
7 mountfsys     mount a removable medium file system
8 umountfsys    unmount a removable medium file system

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

Typing **5** (hdsetup) causes an hdsetup submenu to display. This menu contains subcommands which perform the various aspects of hard disk formatting.

```
                    HARD DISK SETUP

1 bootloader      install a new bootloader
2 fmthdisk        format a hard disk
3 inithdisk       initialize configuration params

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

File Management Menu

The subcommands in this menu allow you to protect files on the hard disk file systems by copying them onto removable media and later restoring them to the hard disk by copying them back. Subcommands are also provided to determine which files might be best stored permanently on tape or diskette based on age or size. See **sysadm**(1) for details.

```
                    FILE MANAGEMENT

1 backup          backup files from built-in disk to removable media
2 backup_rest     backup files to and restore files from removable
                  media
3 diskuse         display how much of the hard disk is being used
4 fileage         list file older than a particular date
5 filesize        list the largest files in a particular directory
6 restore         restore files from "backup" and "store" media
                  to built-in disk
7 store           store files and directories of files onto
                  removable media

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

Choosing menu item 2 causes the **backup_restore** submenu to display.

```
Backup & Restore functions are:
  1. INCREMENTAL backup
  2. FULL backup
  3. SELECTED backup
  4. RESTORE files from archive
  5. ARCHIVE description change
  6. PREFERENCES
  0. QUIT
-- Your choice? [0-6]
```

Note that the format for the **backup_restore** submenu differs somewhat from the other **sysadm** menus. See *File System Backup and Restore* in Chapter 5 for details on this menu.

Machine Management Menu

Machine management functions are tools used to operate the machine, e.g., turn it off, reboot, or go to the firmware monitor. See **sysadm**(1) for details.

```
                    FILE MANAGEMENT

1 firmware  stop all running programs then enter firmware mode
2 powerdown stop all running programs then turn off machine
3 reboot    stop all running programs then reboot the machine
4 whoson    print list of users currently logged onto the system.

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

Package Management Menu

These submenus and subcommands manage various software and hardware packages that you install on your machine. Not all optional packages add subcommands here. Adding the NSE package to a system produces the following package management menu:

```
                    PACKAGE MANAGEMENT
1 rfsmgmt        remote file sharing management menu
2 transpmgmt     transport provider management menu
3 uucpmgmt       basic networking utilities menu

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

The following is the **uucpmgmt** submenu:

```
                    BASIC NETWORKING UTILITIES MANAGEMENT

1 devicemgmt    manage devices (list, add, delete)
2 pollmgmt      manage poll entries (list, add, delete)
3 portmgmt      manage I/O ports (list, modify)
4 systemmgmt    manage remote systems entries (list, add, delete,
                call)

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

Software Management Menu

These subcommands permit you to install new software, remove software, and run software directly from the removable diskette if that is the medium on which it is delivered. The "remove" and "run" capabilities are dependent on the particular software packages. See the instructions delivered with each package and **sysadm**(1).

```
                    SOFTWARE MANAGEMENT

1 installpkg  install new software package onto built-in disk
2 listpkg     list packages already installed
3 removepkg   remove previously installed package from built-in
              disk
4 runpkg      run software package without installing it

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

System Setup Menu

System setup routines allow you to describe the computer's environment: date, time, and time zone; administration and system capabilities under password control; the machine's name, etc. The first-time setup sequence is also part of this menu. See **sysadm**(1) for details.

```
                    SYSTEM SETUP

1 admpasswd    assign or change administrative passwords
2 datetime     set the date, time, time zone and daylight savings
               time
3 devsetup     floppy device setup menu
4 nodename     set the node name of this machine
5 setup        set up your machine the very first time
6 syspasswd    assign system passwords

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

TTY Management Menu

These subcommands allow you to manage the computer's terminal functions.

```
                    TTY MANAGEMENT

1 lineset    show tty line settings and hunt sequences
2 mklineset create new tty line settings and hunt sequences
3 modtty     show and optionally modify characteristics of tty lines

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

User Management Menu

These subcommands allow you to add, modify, and delete the list of users that have access to your machine. You can also place the users in separate groups so that they can share access to files within the group while protecting their files from other groups. See **sysadm**(1) for details.

```
                    USER MANAGEMENT

1 addgroup     add a group to the system
2 adduser      add a user to the system
3 delgroup     delete a group from the system
4 deluser      delete a user from the system
5 lsgroup      list groups in the system
6 lsuser       list users in the system
7 modadduser   modify defaults used by adduser
8 modgroup-make changes to a group on the system <not available>
9 moduser-menu of commands to modify a user's login

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

Typing menu item 9, **moduser**, causes a **moduser** submenu to display. This menu contains subcommands that modify the various aspects of a user's login.

```
                  MODIFY USER'S LOGIN

1 chloginid    change a user's login ID
2 chgpasswd    change a user's password
3 chgshell     change a user's login shell

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

E Accounting

SYSTEM V/88 Accounting provides methods to collect per-process resource utilization data, record connect sessions, monitor disk utilization, and charge fees to specific logins. A set of C language programs and shell procedures is provided to reduce this accounting data into summary files and reports. This appendix describes the structure, implementation, and management of the SYSTEM V/88 accounting system, as well as a discussion of the reports generated and the meaning of the columnar data.

General

The following list is a synopsis of the actions of the accounting system:

- At process termination, the system kernel writes one record per process in **/usr/adm/pacct** in the form of **acct.h**.
- The **login** and **init** programs record connect sessions by writing records into **/etc/wtmp**. Date changes, reboots, and shutdowns are also recorded in this file.
- The disk utilization programs **acctdusg** and **diskusg** break down disk usage by login.
- Fees for file restores can be charged to specific logins with the **chargefee** shell procedure.
- Each day the **runacct** shell procedure is executed via **cron** to reduce accounting data and produce summary files and reports.
- The **monacct** procedure can be executed on a monthly or fiscal period basis. It saves and restarts summary files, generates a report, and cleans up the **sum** directory.

Files and Directories

The **/usr/lib/acct** directory contains all of the C language programs and shell procedures necessary to run the accounting system. The **adm** login (currently user ID of four) is used by the accounting system and has the directory structure shown in Figure E-1.

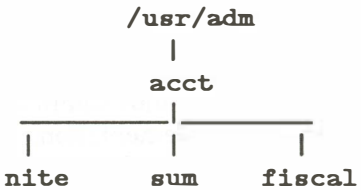


Figure E-1. Directory Structure of the **adm** Login

The **/usr/adm** directory contains the active data collection files. The **nite** directory contains files that are reused daily by the **runacct** procedure. The **sum** directory contains the cumulative summary files updated by **runacct**. The **fiscal** directory contains periodic summary files created by **monacct**.

Daily Operation

When the system is switched into multi-user mode, **/usr/lib/acct/startup** is executed, and the following occurs:

- The **acctwtmp** program adds a "boot" record to **/etc/wtmp**. This record is signified by using the system name as the login name in the **wtmp** record.
- Process accounting is started via **turnacct**. **turnacct on** executes the **accton** program with the argument **/usr/adm/pacct.**
- The **remove** shell procedure is executed to clean up the saved **pacct** and **wtmp** files left in the **sum** directory by **runacct**.

The **ckpacct** procedure is run via **cron**(1M) every hour to check the size of **/usr/adm/pacct**. If the file grows larger than 1000 blocks (default), **turnacct switch** is executed. While **ckpacct** is not absolutely necessary, the advantage of having several smaller **pacct** files becomes apparent when trying to restart **runacct** after a failure processing these records.

The **chargefee** program can be used to bill users for file restores. It adds records to **/usr/adm/fee**, which are picked up and processed by the next execution of **runacct** and merged into the total accounting records.

runacct is executed via **cron** each night. It processes the active accounting files, **/usr/adm/pacct**, **/etc/wtmp**, **/usr/adm/acct/nite/disktacct**, and **/usr/adm/fee**. It produces command summaries and usage summaries by login.

When the system is shut down using **shutdown**, the **shutacct** shell procedure is executed. It writes a shutdown reason record into **/etc/wtmp** and turns process accounting off.

After the first reboot each morning, the computer operator should execute **/usr/lib/acct/prdaily** to print the previous day's accounting report.

Setting Up the Accounting System

To automate the operation of this accounting system, perform the following tasks:

1. If not already present, add this line to the **/etc/rc** file in the state 2 section:

   /bin/su -adm -c /usr/lib/acct/startup

2. If not already present, add this line to **/etc/shutdown** to turn off the accounting before the system is brought down:

   /usr/lib/acct/shutacct

3. For most installations, the following three entries should be made in **/usr/spool/cron/crontabs/adm** so that **cron** automatically runs the daily accounting:

   ```
   0 4 * * 1-6 /usr/lib/acct/runacct 2 > /usr/adm/acct/nite/fd2log
   0 2 * * 1-6 /usr/lib/acct/dodisk
   0 * * * *  /usr/lib/acct/ckpacct
   ```

4. To facilitate monthly merging of accounting data, the following entry in **/usr/spool/cron/crontabs/adm** allows **monacct** to clean up all daily reports and daily total accounting files and deposit one monthly total report and one monthly total accounting file in the **fiscal** directory.

   15 5 1 * * /usr/lib/acct/monacct

The above entry takes advantage of the default action of **monacct** that uses the current month's date as the suffix for the file names. Notice that the entry is executed at a time that allows **runacct** sufficient time to complete. This creates, on the first day of each month, monthly accounting files with the entire month's data.

5. The **PATH** shell variable should be set in **/usr/adm/.profile** to:

PATH=/usr/lib/acct:/bin:/usr/bin

runacct

runacct is the main daily accounting shell procedure. It is normally initiated via **cron** during non-prime time hours. **runacct** processes connect, fee, disk, and process accounting files. It also prepares daily and cumulative summary files for use by **prdaily** or for billing purposes. The following files produced by **runacct** are of particular interest:

nite/lineuse
Produced by **acctcon**, which reads the **wtmp** file. It produces usage statistics for each terminal line on the system. This report is useful for detecting bad lines. If the ratio of logoffs to logins exceeds about 3:1, the line is probably failing.

nite/daytacct
This file is the total accounting file for the previous day in **tacct.h** format.

sum/tacct
This file is the accumulation of each day's **nite/daytacct**, which can be used for billing purposes. It is restarted each month or fiscal period by the **monacct** procedure.

sum/daycms
Produced by the **acctcms** program, it contains the daily command summary. The ASCII version of this file is **nite/daycms**.

sum/cms
The accumulation of each day's command summaries. It is restarted by the execution of **monacct**. The ASCII version is **nite/cms**.

sum/loginlog
Produced by the **lastlogin** shell procedure; it maintains a record of the last time each login was used.

sum/rprt.*MMDD*

Each execution of **runacct** saves a copy of the output of **prdaily**.

runacct takes care not to damage files in the event of errors. A series of protection mechanisms is used that attempt to recognize an error, provide intelligent diagnostics, and terminate processing in such a way that **runacct** can be restarted with minimal intervention. It records its progress by writing descriptive messages into the file **active**. (Files used by **runacct** are assumed to be in the **nite** directory unless otherwise noted.) All diagnostics output during the execution of **runacct** is written to **fd2log**.

To prevent multiple invocations, in the event of two **cron**s or other problems, **runacct** complains if the files **lock** and **lock1** exist when invoked. The **lastdate** file contains the month and day **runacct** was last invoked and is used to prevent more than one execution per day. If **runacct** detects an error, a message is written to **/dev/console**, mail is sent to **root** and **adm**, the locks are removed, diagnostic files are saved, and execution is terminated.

To allow **runacct** to be restartable, processing is broken down into separate reentrant states. This is accomplished by using a *case* statement inside an endless *while* loop. Each state is one case of the *case* statement. A file is used to remember the last state completed. When each state completes, **statefile** is updated to reflect the next state. When **runacct** reaches the CLEANUP state, it removes the locks and terminates. States are executed as follows:

SETUP

The command **turnacct switch** is executed. The process accounting files, **/usr/adm/pacct***?*, are moved to **/usr/adm/Spacct***?.MMDD*. The **/etc/wtmp** file is moved to **/usr/adm/acct/nite/wtmp.***MMDD* with the current time added on the end.

WTMPFIX

The **wtmp** file in the **nite** directory is checked for correctness by the **wtmpfix** program. Some date changes cause **acctcon1** to fail, so **wtmpfix** attempts to adjust the time stamps in the **wtmp** file if a date change record appears.

CONNECT1

Connect session records are written to **ctmp** in the form **ctmp.h**. The **lineuse** file is created, and the **reboots** file is created showing all of the boot records found in the **wtmp** file.

CONNECT2

ctmp is converted to **ctacct.***MMDD*, which are connect accounting records. (Accounting records are in **tacct.h** format.)

PROCESS

The **acctprc1** and **acctprc2** programs are used to convert the process accounting files, **/usr/adm/Spacct***?.MMDD*, into total accounting records in **ptacct***?.MMDD*. The **Spacct** and **ptacct** files are correlated by number so that if **runacct** fails, the unnecessary reprocessing of **Spacct** files does not occur. One precaution should be noted: When restarting **runacct** in this state, remove the last **ptacct** file because it is not complete.

MERGE

Merge the process accounting records with the connect accounting records to form **daytacct**.

FEES

Merge in any ASCII **tacct** records from the file **fee** into **daytacct**.

DISK

On the day after **dodisk** procedure runs, merge **disktacct** with **daytacct**.

MERGETACCT

Merge **daytacct** with **sum/tacct,** the cumulative total accounting file. Each day **daytacct** is saved in **sum/tacct***MMDD*, so that **sum/tacct** can be recreated in the event it becomes corrupted or lost.

CMS

Merge in today's command summary with the cumulative command summary file **sum/cms**. Produce ASCII and internal format command summary files.

USEREXIT

Any installation-dependent (local) accounting programs can be included here.

CLEANUP

Clean up temporary files, run **prdaily** and save its output in **sum/rprt***MMDD*, remove the locks, then exit.

Recovering from Failure

The **runacct** procedure can fail for a variety of reasons: a system crash, **/usr** running out of space, or a corrupted **wtmp** file. If the **active***MMDD* file exists, check it first for error messages. If the **active** file and **lock** files exist, check **fd2log** for any mysterious messages. The following are error messages produced by **runacct**, and the recommended recovery actions:

ERROR: locks found, run aborted
The files **lock** and **lock1** were found. These files must be removed before **runacct** can restart.

ERROR:acctg already run for *date*:**check /usr/adm/acct/nite /lastdate**
The date in **lastdate** and today's date are the same. Remove **lastdate**.

ERROR: turnacct switch returned rc=?
Check the integrity of **turnacct** and **accton**. The **accton** program must be owned by **root** and have the **setuid** bit set.

ERROR: Spacct?.*MMDD* **already exists**
File setups probably already run. Check status of files, then run setups manually.

ERROR:/usr/adm/acct/nite/wtmp.*MMDD* **already exists, run setup manually**
Self-explanatory.

ERROR: wtmpfix errors refer to /usr/adm/acct/nite/wtmperror
wtmpfix detected a corrupted **wtmp** file. Use **fwtmp** to correct the corrupted file.

ERROR: connect acctg failed: check /usr/adm/acct/nite/log
The **acctcon1** program encountered a bad **wtmp** file. Use **fwtmp** to correct the bad file.

ERROR: Invalid state, check /usr/adm/acct/nite/active
The file **statefile** is probably corrupted. Check **statefile** and read **active** before restarting.

Restarting runacct

runacct called without arguments assumes that this is the first invocation of the day. The argument *MMDD* is necessary if **runacct** is being restarted and specifies the month and day for which **runacct** reruns the accounting. The entry point for processing is based on the contents of **statefile**. To override **statefile**, include the desired state on the command line. For example:

To start **runacct**:

nohup runacct 2> /usr/adm/acct/nite/fd2log&

To restart **runacct**:

nohup runacct 0601 2> /usr/adm/acct/nite/fd2log&

To restart **runacct** at a specific state:

nohup runacct 0601 WTMPFIX 2> /usr/adm/acct/nite/fd2log&

Fixing Corrupted Files

Occasionally, a file becomes corrupted or lost. Some of the files can simply be ignored or restored from the file save backup. However, certain files must be fixed to maintain the integrity of the accounting system.

Fixing wtmp Errors

The **wtmp** files seem to cause the most problems in the day-to-day operation of the accounting system. When the date is changed and the system is in multi-user mode, a set of date change records is written into **/etc/wtmp**. The **wtmpfix** program is designed to adjust the time stamps in the **wtmp** records when a date change is encountered. Some combinations of date changes and reboots, however, slip through **wtmpfix** and cause **acctcon1** to fail.

Perform the following steps to patch up a **wtmp** file:

cd /usr/adm/acct/nite
fwtmp < wtmp.*MMDD* **> xwtmp**
ed xwtmp
(delete corrupted records or delete all records from beginning up to the date change)
fwtmp -ic < xwtmp > wtmp.*MMDD*

If the **wtmp** file is beyond repair, create a null **wtmp** file. This prevents any charging of connect time. **acctprc1** cannot determine which login owned a particular process, but it is charged to the login that is first in the password file for that user ID.

Fixing tacct Errors

If the installation is using the accounting system to charge users for system resources, the integrity of **sum/tacct** is quite important. Occasionally, mysterious **tacct** records appear with negative numbers, duplicate user IDs, or a user ID of 65,535. First check **sum/tacctprev** with **prtacct**. If it seems in order, patch up the latest **sum/tacct***MMDD*, then recreate **sum/tacct**. A simple patchup procedure would be:

```
cd /usr/adm/acct/sum
acctmerg -v < tacct.MMDD > xtacct
ed xtacct
(remove the bad records; write duplicate uid records to another file)
acctmerg -i < xtacct > tacct.MMDD
acctmerg tacctprev < tacct.MMDD > tacct
```

Remember that the monacct procedure removes all **tacct.***MMDD* files; therefore, **sum/tacct** can be recreated by merging these files together.

Updating for Holidays

The file **/usr/lib/acct/holidays** contains the prime/non-prime table for the accounting system. The table should be edited to reflect your location's holiday schedule for the year. The format is composed of three types of entries:

Comment Lines

Comment lines may appear anywhere in the file as long as the first character in the line is an asterisk.

Year Designation Line

This line should be the first data line (non-comment line) in the file and must appear only once. The line contains three fields of four digits each (leading white space is ignored). For example, to specify the year 1989, prime time at 9:00 a.m., and non-prime time at 4:30 p.m, the year designation line is:

```
1989        0900        1630
```

A special condition in the time field automatically converts 2400 to 0000.

Company Holiday Lines

These entries follow the year designation line and have the following general format:

day_of_year *month* *day* *holiday_description*

The *day_of_year* field is a number from 1 through 366 that indicates the day of the holiday (leading white space is ignored). The three fields that follow are commentary and are not currently used by other programs.

Reports

runacct generates five basic reports upon each invocation. They cover the areas of connect accounting, usage by person on a daily basis, command usage reported by daily and monthly totals, and a report of the last time that users were logged in.

The following sections describe the reports and the meanings of their tabulated data.

Daily Report

In the first part of the report, the "from/to" banner indicates the period reported on. The times are the time the last accounting report was generated until the time the current accounting report was generated. It is followed by a log of system reboots, shutdowns, power fail recoveries, and any other record dumped into **/etc/wtmp** by the **acctwtmp** program. Refer to **acct**(1M) in the *System Administrator's Manual.*

The second part of the report is a breakdown of line utilization. The TOTAL DURATION tells how long the system was in multi-user mode (able to be accessed through the terminal lines). The columns are:

LINE

The terminal line or access port.

MINUTES

The total number of minutes the line was in use during the accounting period.

PERCENT
The total number of MINUTES the line was in use divided into the TOTAL DURATION.

 # SESS
The number of times this port was accessed for a **login**(1) session.

 # ON
This column does not have much meaning. At one time it gave the number of times the port was used to log in a user; but since **login**(1) can no longer be executed explicitly to log in a new user, this column should be identical to SESS.

 # OFF
This column lists both the number of times a user logged off and any interrupts that occurred on that line. Generally, interrupts occur on a port when the **getty**(1M) is first invoked when the system is brought to multi-user mode. These interrupts occur at a rate of about two per event; therefore, there may be more than twice as many # OFF as # ON or # SESS. This column is important when the # OFF exceeds the # ON by a large factor. It usually indicates that the multiplexer, modem, or cable is going bad, or that there is a bad connection somewhere. The most common cause of this is an unconnected cable dangling from the multiplexer.

During real time, **/etc/wtmp** file should be monitored because this is the file that generates connect accounting. If it grows rapidly, execute **acctcon1** to see which tty line is the noisiest. If the interrupting is occurring at a furious rate, general system performance is affected.

Daily Usage Report

This report gives a by-user breakdown of system resource utilization and consists of the following:

UID
User ID.

LOGIN NAME
Login name of the user. No more than one login name is allowed for a single user ID; this identifies which one.

CPU (MINS)

Amount of time the user's process used the central processing unit. This category is divided into PRIME and NPRIME (non-prime) utilization. The accounting system's definition of this breakdown is located in the **/usr/lib/acct/holidays** file. As delivered, prime time is defined to be 0830-1700 hours.

KCORE-MINS

Represents a cumulative measure of the amount of memory a process uses while running. The amount shown reflects kilobyte segments of memory used per minute. This measurement is also divided into PRIME and NPRIME amounts.

CONNECT (MINS)

Identifies "real time" used. What the column identifies is the amount of time that a user was logged into the system. If this time is rather high and the # OF PROCS column is low, the user is considered a "line hog," i.e., a person who logs in early in the morning and rarely uses the terminal the rest of the day. This column is also subdivided into PRIME and NPRIME utilization.

DISK BLOCKS

When the disk accounting programs have been run, their output is merged into the total accounting record (**tacct.h**) and appears in this column. This disk accounting is accomplished by the program **acctdusg**.

 # OF PROCS

Reflects the number of processes invoked by the user. Check this column for large numbers, indicating that a user may have a shell procedure that is faulty.

 # OF SESS

How many times the user logged onto the system.

 # OF DISK SAMPLES

Indicates how many times the disk accounting was run to obtain the average number of DISK BLOCKS listed earlier.

FEE
An often unused field in the total accounting record, the FEE represents the total accumulation of items charged against the user by the **chargefee** shell procedure (refer to **acctsh**(1M)). The **chargefee** procedure is used to levy charges against a user for special services performed, e.g., as file restores or tape manipulation by operators.

Daily Command and Monthly Total Command Summaries

These two reports are virtually the same except that the Daily Command Summary only reports on the current accounting period, while the Monthly Total Command Summary reports on the start of the fiscal period to the current date. In other words, the monthly report reflects the data accumulated since the last invocation of **monacct**.

The data included in these reports gives an administrator information about which commands are most heavily used, and based on that, an idea of what to weigh more heavily when system tuning.

These reports are sorted by TOTAL KCOREMIN, an arbitrary but useful yardstick for calculating "drain" on a system. The reports contain the following information:

COMMAND NAME
Name of the command. All shell procedures are listed under the name **sh** since only object modules are reported by the process accounting system. The administrator should monitor the frequency of programs called **a.out**, **core**, or some unusual name. **acctcom** is also a good tool to use in determining who executed a suspiciously-named command and whether superuser privileges were used.

NUMBER CMDS
Total number of invocations of this particular command.

TOTAL KCOREMIN
Total cumulative measurement of the kilobyte segments of memory used by a process per minute of run time.

TOTAL CPU-MIN
Total processing time this program has accumulated.

TOTAL REAL-MIN
Total real time (wall clock) minutes the program has accumulated. This total is the actual "waited for" time as opposed to time accumulated by a background process.

MEAN SIZE-K
Mean of the TOTAL KCOREMIN over the number of invocations reflected by NUMBER CMDS.

MEAN CPU-MIN
Mean derived from the NUMBER CMDS and TOTAL CPU-MIN.

HOG FACTOR
Relative measurement of the ratio of system availability to system utilization. It is computed as:

(total CPU time) / (elapsed time)

This gives a relative measure of the total available CPU time consumed by the process during its execution.

CHARS TRNSFD
Total count of the number of characters moved around by the **read**(2) and **write**(2) system calls; it may be a negative.

BLOCKS READ
Total count of the physical block reads and writes performed by that process.

Last Login

This report gives the date when a particular login was last used. It is a good source for finding unused logins and login directories that could be removed.

Account System Files

The following section lists the files found in the accounting directories.

Files in the /usr/adm directory

dtmp
output from the **acctdusg** program

fee
output from the **chargefee** program, ASCII **tacct** records

pacct
active process accounting file

pacct*?*
process accounting files switched via **turnacct**

Spacct*?.MMDD*
process accounting files for *MMDD* during execution of **runacct**

Files in the /usr/adm/acct/nite directory

active
used by **runacct** to record progress and print warning and error messages; **active** *MMDD* same as **active** after **runacct** detects an error

cms
ASCII total command summary used by **prdaily**

ctacct.*MMDD*
connect accounting records in **tacct.h** format

ctmp
output of **acctcon1** program, connect session records in **ctmp.h** format

daycms
ASCII daily command summary used by **prdaily**

daytacct
total accounting records for one day in **tacct.h** format

disktacct
disk accounting records in **tacct.h** format, created by **dodisk** procedure

fd2log
diagnostic output during execution of **runacct**

lastdate
last day **runacct** executed in **date** +%*m*%*d* format

lock lock1
used to control serial use of **runacct**

lineuse
TTY line usage report used by **prdaily**

log
diagnostic output from **acctcon1**

log*MMDD*
same as **log** after **runacct** detects an error

owtmp
stores an existing **tmpwtmp** before creating a new one

ptacct.*n*.*MMDD*
processes **tacct** records *n* files for *MMDD*

reboots
contains beginning and ending dates from **wtmp** and a listing of reboots

statefile
used to record current state during execution of **runacct**

tmpwtmp
wtmp file corrected by **wtmpfix**

wtmperror
place for **wtmpfix** error messages

wtmperror*MMDD*
same as **wtmperror** after **runacct** detects an error

wtmp.*MMDD*
previous day's **wtmp** file

Files in the /usr/adm/acct/sum directory

cms
total command summary file for current fiscal in internal summary format

cmsprev
command summary file without latest update

daycms
command summary file for yesterday in internal summary format

loginlog
created by **lastlogin**

pacct.*MMDD*
concatenated version of all **pacct** files for *MMDD*, removed after reboot by **remove** procedure

rprt.*MMDD*
saved output of **prdaily** program

tacct
cumulative total accounting file for current fiscal

tacctprev
same as **tacct** without latest update

tacct.*MMDD*
total accounting file for *MMDD*

wtmp.*MMDD*
saved copy of **wtmp** file for *MMDD*, removed after reboot by **remove** procedure

Files in the /usr/adm/acct/fiscal directory

cms*?*
total command summary file for fiscal *?* in internal summary format

fiscrpt*?*
report similar to **prdaily** for fiscal *?*

tacct*?*
total accounting file for fiscal *?*

G Glossary

When using the glossary, keep the following in mind:

- Words in *italics* are defined in this glossary.
- A number in square brackets ([ ]) following a word in italics refers to an entry number in that word's definition. The following example refers to the first definition of the word **file**, i.e., 1. in general, a potential source of input or destination for output.

  file [1]

address
A number, label, or name that indicates the location of information in the computer's *memory*.

advertise
To make a local *resource* available to other computers using *Remote File Sharing* (RFS). The **adv**(1M) (or **sysadm advnow**) command is used by administrators to advertise a resource.

advertise table
An internal list of available resources. An advertise table on each computer running RFS has the name of each resource the computer has made available.

a.out
The default name of a freshly compiled *object file*, pronounced 'A-dot-out'; historically **a.out** signified assembler output.

archive
1. A collection of data gathered from several *files* into one file. 2. Especially, such a collection gathered by **ar**(1) for use as a library.

automatic advertise list
A list of local resources that are automatically offered to other computers when RFS is started. The list consists of full **adv** command lines placed inside the **/etc/rstab** file. The command lines are added to **/etc/rstab** using the **sysadm advauto** command or by any standard file editor.

automatic calling unit
A hardware *device* used to dial stored telephone numbers; allows the system to contact another system over phone lines without manual intervention.

automatic mount list
A list of remote resources that are mounted on the local system when RFS

automatic calling unit
A hardware *device* used to dial stored telephone numbers; allows the system to contact another system over phone lines without manual intervention.

automatic mount list
A list of remote resources that are mounted on the local system when RFS is started. The list is contained in the **/etc/fstab** file. (See the **fstab**(4) manual page in the *Programmer's Reference Manual* for the format of the file.) Automatic mount information is added to **/etc/fstab** using the **sysadm mntauto** command or by any standard file editor.

bad block
A section of a storage medium that cannot store data reliably.

block
The basic unit of *buffering* in the *kernel*, 1024 bytes; see *indirect*, *logical*, and *physical blocks*.

block device
A *device* upon which a *file system* [1] can be *mounted*, typically a permanent storage device, e.g., a tape or disk drive, so called because data transfers to the device occur by *blocks*; cf. *character device*.

boot
To start the *operating system*, so called because the *kernel* must bootstrap itself from secondary storage into an empty machine. No *login* [3] or *process* persists across a boot. **boot block** the first block of a *file system* [1], that is reserved for a booting program.

boot program
Loads the *operating system* into *core*, i.e., physical memory.

buffer
1. A staging area for input-output where arbitrary-length transactions are collected into convenient units for system operations; the *file system* [3] uses buffers, as does *stdio*. 2. To use buffers.

buffer pool
A region of store available to the *file system* [3] for holding *blocks*; all but *raw* [2] input-output for *block devices* goes through the buffer pool so read and write operations may be independent of device blocks.

cartridge tape
A storage medium that consists of a magnetic tape wound on spools housed in a plastic container.

character device
A *device* upon which a *file system* [1] cannot be *mounted*, e.g., a terminal or the *null device*.

child process
See *fork*.

client
An RFS computer that is using a remote resource.

client list
When an RFS administrator advertises a resource, the administrator can restrict the resource so only certain remote machines can use it. This list of machines is added to the **adv**(1M) command line when a resource is advertised.

client permissions
When an RFS administrator advertises a resource, the administrator can set permissions for the resource. The permissions are assigned on the **adv**(1M) command line. If the permissions are read-only, the client computers can only mount the resource with read permissions. If they are read/write, a client can mount the resource read/write or read only.

command
1. An instruction to the *shell*, usually to run a *program* [1] as a *child process*.
2. By extension, any *executable file*, especially a *utility program*.

command file
Same as *shell script*.

configuration
The arrangement of the software or hardware of a system, peripheral, or network as defined by the nature, number, and chief characteristics of its functional units.

controller
A *device* that directs the transmission of data over the data links of a *network*.

core file
A *core image* of a terminated *process* saved for debugging; a core file is created under the name 'core' in the *current directory* of the process.

core image
A copy of all the *segments* of a running or terminated program; the copy may exist in main store, in the *swap area*, or in a *core file*.

crash
If a hardware or software *error* condition develops that the system cannot handle, it takes itself out of service, or crashes. Such conditions occur when the system cannot allocate resources, manage *processes*, respond to requests for system functions, or when the electrical power is unstable.

cron
A command that creates a daemon that invokes commands at specified dates and times.

current name server
When a domain is set up, a primary and zero or more secondary domain name servers are assigned. Only one of those machines is actually handling domain name server responsibilities at a time. That machine is referred to as the current name server. Normally, the primary is the current name server. However, if the secondary has taken over temporarily, it is the responsibility of the secondary's administrator to pass the responsibility back to the primary whenever the primary resumes running RFS. (See also **domain**, **primary name server**, and **secondary name server**.)

cylinder
The set of all *tracks* on a *disk* that are the same distance from the axis about which the disk rotates.

daemon
A *background* process, often perpetual, that performs a system-wide public function, e.g., **calendar** (1) and **cron** (1M); the affected spelling is an ancient legacy.

destination
The remote system that will ultimately receive a *file* transferred over a *network*.

device
1. A *file* [2] that is not a *plain file* or a *directory*, e.g., a tape drive, or the *null device*; a *special file*. 2. A physical input-output unit.

diagnostic
A message printed at your terminal that identifies and isolates *program* errors.

directory
A *file* that comprises a catalog of *filenames* [2]; the organizing principle of the *file system* [2], a directory consists of *entries* that specify further *files* (sense 2, including directories), and constitutes a node of the *directory tree*.

directory entry; entry
1. An association of a name with an *inode number* appearing as an element of a *directory*. 2. The name part of such an association.

directory hierarchy
The tree of all *directories*, in which each is reachable from the *root* via a chain of *subdirectories*.

directory pathname
RFS asks you for a full pathname to a directory in two instances. When you advertise a local resource, you need the full pathname of the directory you are advertising. When you mount a remote resource, you need the full pathname of the directory where the remote resource should be attached.

directory tree
Same as *directory hierarchy*.

disk
A platter coated with magnetic material on which data can be stored.

diskette
A magnetic storage medium that is smaller and more flexible than a hard *disk*.

domain
A logical grouping of computers in an RFS environment. A domain name is like a telephone area code, acting as an addressing prefix to attach to a computer name or a resource. Assigning a domain name server for a domain provides a central location where lists of resources and network addresses for the group of computers can be stored. Domains also provide a level of security.

domain information (rfmaster)
The primary and secondary name server assignments for a domain are stored in the **/usr/nserve/rfmaster** file. The primary keeps the definitive copy of this file and distributes it automatically to each computer in the domain when each starts RFS. This file also contains the network address of each name server.

domain member list
The list of the computers that make up an RFS domain. This list is stored in the **/usr/nserve/auth.info/***domain*/**passwd** file on the primary name server, where *domain* is replaced by the name of the domain. Members are added on the primary using the **rfadmin –a** or **sysadm addmember** commands.

domain name server
A computer that creates and maintains the following information for *hosts* in an RFS domain: *advertised resources*, *host* names and passwords, names and addresses for name servers of other domains (optional), *host* user and group information used for *ID mapping* (optional).

drive
The hardware device that holds magnetic disks, diskettes, and tapes while they are in use.

dump
A copy of the *core image* of the *operating system*.

environment
1. A set of strings, distinct from the *arguments*, made available to a *process* when it *executes* [2] a *file*; the environment is usually inherited across *exec*(2) operations. 2. A specific environment [2] maintained by the *shell*. 3. A nebulously identified way of doing things, as in 'interactive environment': a deprecated usage, not always removed from these manuals.

error
Occurs when a hardware or software condition prevents the successful *execution* of a system or a user *process*.

error message
A message sent from the system to the *system console* when an *error* occurs.

exec
A system call that allows the user to request the execution of another program.

executable file
1. An *object file* that is ready to be copied into the *address* space of a *process* to run as the code of that process. 2. A file that has execute *permission*, either an *executable file* [1] or a *shell script*.

execute
1. Informally, to run a *program*. 2. To replace the *text segment* and *data segments* of a *process* with a given *program* [1].

FIFO
A named permanent *pipe* that allows two unrelated *processes* to exchange information using a pipe connection.

file
1. In general, a potential source of input or destination for output. 2. Most specifically, an *inode* and/or associated contents, i.e., a *plain file*, a *special file*, or a *directory*. 3. A *directory entry*; several directory entries may name the same file [2]. 4. Most loosely, a *plain file*.

file descriptor
A conventional integer quantity that designates an *open file*.

filename
1. A *pathname*. 2. The last component name in a pathname.

file system
1. A collection of *files* that can be *mounted* on a block *special file*; each file of a file system appears exactly once in the *i-list* of the file system and is accessible via some *path* from the *root* directory of the file system. 2. The collection of all *files* on a computer. 3. The part of the kernel that deals with file systems [1].

filter
A *program* [1] that reads from the *standard input* and writes on the *standard output*, so called because it can be used as a data-transformer in a *pipeline.*

firmware
The Non-Volatile Random Access Memory (NVRAM), that permanently holds a few, special programs.

flush
To empty a *buffer*, for example to throw away unwanted input-output upon *interrupt* or to release output from the clutches of *stdio.*

forced unmount
To unmount one of your local resources from all remote machines that have mounted it. This has the effect of killing all processes that are currently using the resource on all client machines.

fork
To split one *process* into two, the **parent process** and **child process**, with separate, but initially identical, *text*, *data*, and *stack segments.*

formatting
The process of imposing an addressing scheme on a *disk.* This includes the establishment of bad sector information and the mapping of both sides of the disk into *tracks* and *sectors.*

free list
In a *file system* [1], the list of *blocks* that are not occupied by data.

getty
One of a series of *processes* that connect the user to SYSTEM V/88. **getty** is invoked by **init**, and in turn invokes **login**.

group
1. A set of *permissions* alternative to *owner* permissions for access to a *file.* 2. A set of *userids* that may assume the privileges of a group [1]. 3. The *groupid* of a file.

groupid
An integer value, usually associated with one or more *login names;* as the *userid* of a process becomes the *owner* of files *created* by the process, so the groupid of a process becomes the *group* [3] of such files.

hole
A gap in a *plain file* caused by *seeking* while writing; **read**(2) takes data in holes to be zero; a *block* in a hole occupies no space in its *file system*.

host
A computer that is configured to share *resources* in a RFS environment.

ID mapping
To define the permissions remote users and groups have to your advertised resources. The tools available for mapping let you set permissions on a per-computer basis and a global basis. You can then map individual users or groups by ID name or number. When you map IDs for RFS, it is easiest to do so with ID numbers since mapping by name requires that you have copies of the remote machines' **/etc/passwd** and **/etc/group** files.

i-list
The index to a *file system* [1] listing all the *inodes* of the file system; cf. *inode number*.

indirect blocks
Data blocks that are not directly referenced by an *inode* (because the file is larger than 10 1024-byte blocks); the inode has three *addresses* that indirectly reference (by a cascade of pointers) some 2,114,114 data blocks (an extremely large potential *file* size). The inode has one address that points to 256 more data blocks; a second address that points to 256 blocks that each point to 256 data blocks; and finally a third address that points to 256 blocks each of which point to another 256 blocks, each of which point to 256 data blocks.

init
A general *process* spawner that is invoked as the last step in the *boot* procedure; it regularly checks a table that defines what processes should run at what *run level*.

inode
An element of a *file system* [1]; an inode specifies all properties of a particular *file* [2] and locates the file's contents, if any.

inode number; i-number
The position of an *inode* in the *i-list* of a *file system* [1].

instruction
See *address*.

integrity
In a *file system*, the quality of being without errors due to *bad blocks*.

interface programs
shell scripts furnished with the LP *spooling* software that interface between the user and the printer.

interrupt
1. A *signal* that normally terminates a *process*, caused by a BREAK or an interrupt character. 2. A signal generated by a hardware condition or a peripheral *device*. 3. Loosely, any *signal*.

IPC
An acronym for interprocess communication (e.g., signals, shared memory messages, semaphores).

kernel
SYSTEM V/88 proper; resident code that implements the *system calls*.

kernel address space
A portion of memory used for data and code addressable only by the *kernel*.

line discipline
A module to handle protocol or data conversion for a *stream* [2]. A line discipline, unlike a *filter*, is part of the *kernel*.

link
1. To add an entry for an existing *file* to a directory; converse of *unlink*. 2. By extension, a *directory entry*. 3. Loosely, any but one putatively primary directory entry for a given *inode*; either linked [1] or a *symbolic link*.

link count
The number of *directory entries* that pertain to an *inode*; a *file* ceases to exist when its link count becomes zero and it is not *open*.

load device
Designates the physical *device* from which a program is loaded into main *memory*.

local resource
A directory that resides on your machine that you have made available for other computers running RFS to use. You must advertise the directory (**adv**(1M)) to offer it to other computers. If a remote machine mounts your resource, it could have access to all subdirectories, files, named pipes, and devices within your directory (depending on file permissions you set up).

log files
Files that contain records of transactions that occur on the system; software that *spools*, for example, generates various log files.

logical block
A unit of data as it is handled by the software; SYSTEM V/88 handles data in 1024-byte logical blocks.

login
1. The *program* that controls logging in. 2. The act of *logging in.* 3. By extension, the computing session that follows a login [2].

memory
1. Same as *memory image.* 2. Physical memory represents the available space in main memory; *programs* are either *swapped* or *paged* into physical memory for *execution.* 3. Virtual memory management techniques permit *programs* to treat *disk* storage as an extension of main memory.

memory image
Same as *core image.*

mode; file mode
The *permissions* of a *file;* colloquially referred to by a 3-digit octal number, e.g., 'a 755 file'; see *chmod*(1).

mount
To extend the *directory hierarchy* by associating the *root* of a *file system* [1] with a *directory entry* in an already mounted file system; converse is unmount, spelled 'umount'. The special use of **mount**(1M) in RFS is to attach a remote resource to a directory on your system so local users can access the remote resource. Once mounted, the remote resource appears to be just another part of the local file system tree. See the **mount –d**(1M) command and the **sysadm mountmgmt** procedures.

namelist
Same as *symbol table.*

network
The hardware and software that constitute the interconnections between computer systems, permitting electronic communication between the systems and associated peripherals.

network address
The address by which a computer is known to a particular network. An RFS administrator may need to know the network address of the primary to start RFS for the first time. Address information of other machines is handled internally by RFS. Normally, however, the Transport Provider (TP) maintains the mapping of host name to address. Thus, once the TP is set up, the RFS administrator must only know the host name of the primary name server. RFS does store this name internally, though, so changes in the network address of primaries and secondaries requires reconfiguring RFS.

network listener
The process used by a TP to wait for any type of incoming requests from the network. Once a request comes in, the listener directs it to one of the processes registered with the listener. The process represents a service, e.g., **uucp** or RFS.

network specification (net_spec)
The name that identifies a networking product that is compatible with the Transport Interface (TI). This is also referred to as the TP. RFS requires a TP to communicate with other machines. The network specification is used to tell RFS the exact device to use for communications. For example, you enter **tcpip** to tell RFS to use the **/dev/tcpip** device if the TCP/IP TP package is being used.

networking support utilities
A software package that contains the network listener. This package must be installed to use RFS.

networking
For computer systems, means sending data from one system to another over some communications medium (e.g., coaxial cable, phone lines). Common networking services include *file* transfer, remote *login*, remote *execution*.

node name
The name you assign to your computer to use for communications needs (use **sysadm nodename** to change it). Networking software, e.g., Basic Networking Utilities and RFS, use this name to identify your machine. A full RFS computer name is *domain.nodename*, where *domain* is the name of the computer's RFS domain.

null device
A *device* [1] that always yields *end of file* on reading and discards all data on writing.

object file
A *file* of machine language code and data; object files are produced from source programs by compilers and from other object files and libraries by the link editor; an object file that is ready to run is an *executable file* [1].

operating system
The *program* for managing the resources of the computer. It takes care of such things as input/output procedures, process scheduling, and the file system, thereby removing this burden from user programs.

open file
1. The destination for input or output obtained by *opening* a *file* or creating a *pipe*; a *file descriptor*; open files are shared across *forks* and persist across *executes* [2]. 2. Loosely, a file that has been opened, however an *open file* [1] need not exist in a *file system* [1], and a *file* [2] may be the destination of several *open files* simultaneously.

other
1. A set of *permissions* regulating access to a *file* by processes with *userid* different from the *owner* and *groupid* different from the *group* of the file. 2. The customary name of the default *group* [2] assigned upon *login*.

owner
The *userid* of the *process* that created a *file*; the owner has distinctive *permissions* for a file.

page
A fixed length, 1024-byte block that has a virtual *address*, and that can be transferred between main and secondary storage.

paging
The process by which *programs* are truncated into *pages* and transferred between main and secondary storage by the virtual handler (or paging *daemon*).

parent process
See *fork*.

partitions
Units of storage space on disk.

path; pathname
A chain of names designating a *file*; a **relative pathname** leads from the current directory, for example, a path to *directory* A, thence to directory B, thence to *file* C is denoted A/B/C; a **full pathname** begins at the *root*, indicated by an initial '/', as in /A/B/C.

permission
A right to access a *file* in a particular way; read, write, execute (or look up in, if a directory). Permissions are granted separately to *owner*, *group*, and *others*.

permission A permission, so called because each permission is encoded into one bit in an *inode*.

physical block
A unit of data as it is actually stored and manipulated; SYSTEM V/88 handles data in 1024-byte physical blocks.

physical memory
See *memory*.

pipe
A direct stream connection between *processes*, whereby data written on an *open file* in one process becomes available for reading in another.

pipeline
A sequence of *programs* [1] connected by *pipes*.

polling
The interrogation of *devices* by the *operating system* to avoid contention, determine operation status, or ascertain readiness to send or receive data.

ports
The point of physical connection between a peripheral *device* (e.g., a terminal or a printer) and the device *controller* (ports board), that is part of the computer hardware.

primary name server
The computer that is assigned to provide a central location for addressing and information collection for an RFS domain. Information includes a list of domain members, resources offered by domain members, and optional user ID mapping information. Secondary name servers can be assigned to continue limited name service when the primary is down. For example, a secondary cannot add or delete domain members.

process
A connected sequence of computation; a process is characterized by a *core image* with instruction location counter, *current directory*, a set of *open files*, *control terminal*, *userid*, and *groupid*.

process id
An integer that identifies a *process*.

process number
Same as *process id*.

profile
1. An optional *shell script*, '.profile', conventionally used by the *shell* upon *logging in* to establish the *environment* [3] and other working conditions customary to a particular user. 2. To collect a histogram of values of the instruction location counter of a *process*.

program
1. An *executable file*. 2. A *process*. 3. All the usual meanings.

queue
A line or list formed by items in a system waiting for service.

raw device
A *block device*, read and write operations to which are not *buffered*, and are synchronized to natural records of the physical *device*.

reboot
Same as *boot*.

region
A group of machine *addresses* that refer to a base address.

release
A distribution of fixes or new functions for an existing software product.

Remote File Sharing (RFS)
A software utilities package that enables computers to share *resources* across a network.

RFS automatic startup
RFS can be set to start automatically when your machine is booted. This is done by changing the **initdefault** line in the **/etc/inittab** file from **2** to **3**. The **sysadm setauto** command does this for you automatically. (**init** state **3** is the RFS/Multi-user state.)

RFS daemon (rfudaemon)
A daemon process that runs when RFS is running. When network connections to remote resources are broken, **rfudaemon** sends a message to **rfuadmin**, which then continuously tries to remount the resource. (See **rfudaemon**(1M) and **rfuadmin**(1M) for further information.)

RFS password
A password assigned by the primary name server for every computer in its domain. Each computer must enter its password the first time it starts RFS. After that the password is stored locally in **/usr/nserve/loc.passwd**. By copying the domain password file from the primary (**/usr/nserve/auth.info/***domain*/**passwd**), a computer can verify that a remote machine trying to mount its resource is the machine it claims to be.

RFS State (init 3)
The special initialization state used to start RFS. When you type **init 3** or set the **initdefault** line in **/etc/inittab** to **3**, your system starts RFS, advertises all resources in your automatic advertise list, and mounts all resources in your automatic mount list.

remote resource
A directory that resides on a remote machine that is available for you to connect to using RFS. You must mount the resource (using **mount**(1M) or **sysadm mntnow**) to make it available to users on your system. Once you mount the remote resource, your users could have access to all subdirectories, files, named pipes, and devices related to your directory (depending on file permissions the remote machine set up).

resource
A directory that is *advertised* in a RFS environment. When a *resource* is *mount*ed on a *client*, the contents of the directory (files, devices, and named pipes) and any of its subdirectories are potentially available to users on the *client*. See also **remote resource** and **local resource**.

resource identifier
The name assigned to a resource when it is advertised. The name is limited to 14 printable ASCII characters; slash (/), period (.), and white space cannot be used.

re-tension
The process of re-winding the tape in a *cartridge tape device* to make sure it is at the correct tautness for accurate recording of data.

root
1. A distinguished directory that constitutes the origin of the *directory hierarchy* in a *file system* [1]. 2. Specifically, the origin for the *file system* [2], with the conventional *pathname* '/'. 3. The origin of the directory hierarchy in a *file system* [1].

rotational gap
The gap between the actual *disk* locations of blocks of data belonging to the same *file*; the rotational gap compensates for the continuous, high-speed rotation of the disk so that when the controller is ready to reference the next physical block, the read-write head is positioned correctly at the beginning of that block.

run level
A software *configuration* of the system that allows a particular group of *processes* to exist.

schedule
To assign resources —main store and CPU time— to *processes*.

scheduler
A permanent *process*, with *process number* 1, and associated *kernel* facilities that does scheduling.

search path
In the *shell*, a list of *pathnames* of *directories* that determines the meaning of a *command*; the command name is prefixed with members of the search path in turn until a pathname of an *executable file* [2] results; the search path is given by the shell variable PATH.

secondary name server
A computer designated to take over name server responsibilities temporarily should the primary domain name server fail. The secondary cannot change any domain information. It can, and should, only pass name server responsibility back to the primary when RFS is running on the primary again.

section, sector
A 512-byte portion of a *track* that can be accessed by the magnetic disk heads in the course of a predetermined *rotational* displacement of the storage device.

segment
A contiguous range of the address space of a *process* with consistent store access capabilities; the four segments are (i) the **text segment**, occupied by executable code, (ii) the **data segment**, occupied by *static* data that is specifically initialized, (iii) the **bss segment**, occupied by static data that is initialed by default to zero values, and (iv) the **stack segment**, occupied by *automatic* data, see *stack*; sometimes (ii), (iii), and (iv) are collectively called data segments.

semaphore
An *IPC* facility that allows two or more processes to be synchronized.

server
A RFS computer that offers a resource to others.

set userid
A special *permission* for an *executable file* [1] that causes a *process* executing it to have the access rights of the *owner* of the file; the owner's *userid* becomes the **effective userid** of the process, distinguished from the **real userid** under which the process began.

set userid bit
The associated *permission bit*.

shared memory
An *IPC* facility that allows two or more processes to share the same data space.

shell
1. The program **sh**(1) that causes other programs to be executed on *command*; the shell is usually started on a user's behalf when the user *logs in*. 2. By analogy, any program started upon logging in.

shell script
An executable *file* of *commands* taken as input to the *shell*.

signal
An exceptional occurrence that causes a *process* to terminate or divert from the normal flow of control; see *interrupt*, *trap*.

single-user
A state of the *operating system* in which only one user is supported.

source file
1. The uncompiled version of a *program*. 2. Generally, the unprocessed version of a *file*.

special file
An *inode* that designates a *device*, further categorized as either (i) a **block special file** describing a *block device*, or (ii) a **character special file** describing a *character device*.

spool
To collect and serialize output from multiple *processes* competing for a single output service.

spool area
A *directory* in which a spooler collects work.

spooler
A *daemon* that spools.

stack
A *segment* of the *address* space into which *automatic* data and subroutine linkage information is allocated in last-in-first-out fashion; the stack occupies the largest data addresses and grows downward towards *static* data.

standard error
One of three files described under *standard output*.

standard input
The second of three files described under *standard output*.

standard output
open files, customarily available when a *process* begins, with *file descriptors* 0, 1, 2 and *stdio* names 'stdin', 'stdout', 'stderr'; where possible, utilities by default read from the standard input, write on the standard output, and place error comments on the standard error file. Initially, all three of these files default to your terminal.

startup
Same as *boot*

sticky bit
A *permission* flag that identifies a file as a *sticky file*.

sticky file
A special *permission* for a *shared text* file that causes a copy of the *text segment* to be retained in the *swap area* to improve system response.

super block
The second *block* in a *file system* [1] that describes the allocation of space in the file system; cf. *boot block*.

super user
userid 0, that can access any *file* regardless of *permissions* and can perform certain privileged *system calls*, e.g., setting the clock.

swap
To move the *core image* of an executing program between main and secondary storage to make room for other *processes*.

swap area
The part of secondary store to which *core images* are *swapped*; the swap area is disjointed from the *file system.*

symbolic link
An *inode* that contains the *pathname* of another. References to the symbolic link become references to the named inode.

symbol table
Information in an *object file* about the names of data and functions in that file; the symbol table and *address* relocation information are used by debuggers and also by the link editor to compile *object files.*

System Administration
When capitalized, refers to the package of screens and interactive prompts, invoked through the **sysadm**(1) command, that help you accomplish most system administration tasks.

system calls
1. The set of system primitive functions through which all system operations are allocated, initiated, monitored, manipulated, and terminated.
2. The system primitives invoked by user *processes* for system-dependent functions, e.g., I/O, process creation.

system console
The directly connected terminal used for communication between the operator and the computer.

system name
An up-to-six character name for the system; resides in the SYS parameter.

table
An array of data each item of which may be uniquely identified by means of one or more arguments.

text file; ASCII file
A *file*, the bytes of which are understood to be in ASCII code.

track
An addressable ring of *sections* on a *disk* or *diskette*; each disk or diskette has a predefined number of concentric tracks that allows the disk head to properly access *sections* of data.

transport provider (TP)
The software that provides a path through which network applications can communicate. RFS can communicate over any TP that meets the Transport Interface Specification. (The TCP/IP TP is one Transport Interface-compatible provider.)

trap
A method of detecting and interpreting certain hardware and software conditions via software; a trap is set to catch a *signal* (or *interrupt*), and determine what course of action to take.

tunable parameters
Variables used to set the sizes and thresholds of the various control structures of the *operating system*.

tuning
1. Modifying the *tunable parameters* so as to improve system performance.
2. The reconfiguration of the *operating system* to incorporate the modifications into *executable* version of the system.

user/group name
The names associated with each local user and group that is allowed access to your computer. This information can be found in the first field of the **/etc/passwd** or **/etc/group** files, respectively. Remote users and groups can be assigned the same permissions as the local users and groups by using RFS ID mapping.

user/group ID number
Every user and group name has a corresponding number that is used by the *operating system* to handle permissions to files, directories, devices. These numbers are defined in the third field of the **/etc/passwd** or **/etc/group** files, respectively. Remote users and groups can be assigned the same permissions as the local users and groups by using RFS ID mapping.

userid
An integer value, usually associated with a *login name*; the userid of a *process* becomes the *owner* of files *created* by the process and descendent (*forked*) processes.

utility; utility program
A standard, generally useful, permanently available *program*.

version
A separate *program* product, based on an existing one, but containing significant new code or new functions.

virtual memory
See *memory*.

VTOC
Volume Table of Contents is the section of a disk that shows how the *partitions* on the *disk* are allocated.

MOTOROLA INC.

Microcomputer Division
2900 South Diablo Way
Tempe, Arizona 85282
P.O. Box 2953
Phoenix, Arizona 85062

11043 PRINTED IN USA (3/90) WPC 2,500