IRIX™ Advanced Site and Server Administration Guide

Document Number 007-0603-100

CONTRIBUTORS

Written by Jeffrey B. Zurschmeide
Edited by Christina Cary
Cover design and illustration by Rob Aguilar, Rikk Carey, Dean Hodgkinson, Erik Lindholm, and Kay Maitz
Document Production by Lorrie Williams
Engineering contributions by Kam Kashani, Andrew Cherenson, Chris Wagner, Dave Higgen, Jeff Doughty, Paul Mielke, Robert Stephens, Joe Yetter, Jack Weldon, Gretchen Helms, andVernon Schryver.

IRIX™ Advanced Site and Server Administration Guide
Document Number 007-0603-100

Contents

List of Figures

List of Tables

Introduction

This guide explains how to use the system-level IRIX® utilities available with IRIS® workstations and servers. It provides descriptions of a broad range of tasks, from turning on a system, to adding users, to connecting systems in a network.

The standard network communications software that runs on Silicon Graphics® workstations is derived from the networking software in the 4.3BSD UNIX® releases from the University of California at Berkeley and the Sun® Microsystems RPC® (remote procedure call) system. The IRIX operating system implements the Internet Protocol suite and UNIX domain sockets using the 4.3BSD UNIX socket mechanism. The system also supports access to the underlying network media by means of raw sockets.

If you have a graphics workstation, you may find it convenient to use the System Manager, which is described in the *Personal System Administration Guide*. That guide should be your first resource for administering graphics workstations. Regardless of whether you use the System Manager or the IRIX command-line interface, the results are the same. The System Manager does not create any new files on your system, unlike applications such as WorkSpace.

If you have a server, this book (the *IRIX Advanced Site and Server Administration Guide*) is your primary guide to system administration, since without graphics, you cannot use the System Manager. This guide does not describe the System Manager in great detail. Instead, it covers the traditional shell command approach to administering an IRIX operating system.

If you are running the Trusted IRIX/B™ operating system, you should also read the *Trusted IRIX/B Security Administration Guide* for additional instructions and procedures necessary to maintain system security.

Overview of This Guide

Objective

The *IRIX Advanced Site and Server Administration Guide* is written for administrators who are responsible for performing tasks beyond the reasonable scope of "end users." Frequently, people who would consider themselves end users find themselves performing advanced administrative tasks. This book has been prepared to help both the new and experienced administrator successfully perform all operations necessary to maintain a single system or network of systems. It is hoped that people who considered themselves end users in the past will, by using this book, gain experience and confidence in successfully performing advanced system administration tasks.

The title of the book indicates that the material covered is advanced, beyond the scope of the Personal System Administration Guide, and that the topics covered are not only those needed to administer a single system, but also those needed to maintain an entire network (a site) of systems and servers. This guide contains chapters that address the advanced issues a graphics workstation administrator encounters and all the issues that a site and server administrator encounters.

Contents

This guide contains:

Chapter 1 "System Administration Basics" provides an overview of the tasks expected of a system administrator. It describes the various tools available to the administrator and the various pieces of the administration documentation.

Chapter 2 "Operating the System" addresses the standard operations of your workstation or server. It also describes a site administrator's responsibilities and how to keep the system running smoothly.

Chapter 3 "User Services" deals with login administration, the user environment, communication services, and resolving user problems.

Chapter 4	"The Command (PROM) Monitor" tells you how to use boot-level utilities to configure and test your system. It describes the boot environment of the workstation and each of the Command Monitor commands.
Chapter 5	"Tuning System Performance" describes how to analyze system performance and adjust system parameters.
Chapter 6	"Backing Up and Restoring Files" tells you how and when to back up the data on your system.
Chapter 7	"Disks and Tape Drives" lists the steps to add and maintain hard disks and how to use tape drives.
Chapter 8	"File System Administration" discusses how file systems are organized, how they work, and how to maintain them.
Chapter 9	"Administering Printers" provides instruction on the installation and maintenance of local and networked printers.
Chapter 10	"Terminals and Modems" describes how to set up and maintain serial terminals, modems, and other serial devices.
Chapter 11	"Administering the CADMIN Object System" describes the maintenance of the daemons that support the System Manager.
Chapter 12	"System Security" describes how you can keep your system as secure as possible using the standard IRIX system. The optional Trusted IRIX/B secure operating system is covered in separate documentation, not in this chapter.
Chapter 13	"Administering the System Audit Trail" describes the System Audit Trail and demonstrates how to use this subsystem to produce an exact record of all system activity.
Chapter 14	"System Accounting" describes the accounting subsystem and demonstrates how to use it to account for CPU time and disk space on a per-user basis.
Chapter 15	"Understanding Silicon Graphics' Networking Products" discusses Silicon Graphics standard hardware and software networking products and describes the standard software configuration (files, daemon, processes).

Chapter 16	"Planning a Network" provides insight into planning a network. It includes internet addressing, the *hosts* database file, when to use certain applications, how to subnet a network, security issues, and heterogeneous network considerations.
Chapter 17	"Setting Up a Network" describes, through example, the process of configuring a network (homogeneous and heterogeneous), how to set up a router, and basic troubleshooting advice.
Chapter 18	"Managing a Network" describes the various tools available for managing a network, including backup strategies, performance issues, and fault isolation.
Chapter 19	"The BIND Name Server" provides an overview of the Berkeley Internet Name Domain (BIND) server, also known as *named*. It also provides an example setup procedure and general information on managing and troubleshooting BIND.
Chapter 20	"IRIX sendmail" provides an overview of the mail system, the *sendmail* program, and the *alias* database. It contains a planning checklist and a setup example for various *sendmail* configurations.
Chapter 21	"UUCP" compares TCP/IP and UUCP and describes the features and functions of the UUCP networking utilities. It also provides a setup example and information about common UUCP error messages.
Chapter 22	"SLIP and PPP" describes the features and functions of SLIP and details how to connect two stations using SLIP.
Appendix A	"IRIX Kernel Tunable Parameters" describes the kernel parameters you can change to influence system performance.
Appendix B	"IRIX Device Files" lists the device files and directories on IRIS workstations and servers.
Appendix C	"IRIX Kernel Error Messages" lists kernel error messages, their meanings, and what you should do about them.
Appendix D	"IRIX sendmail Reference" provides a concise reference to *sendmail* as it is implemented under IRIX.

Appendix E "BIND Standard Resource Record Format" provides detailed information about all standard resource record formats used in BIND configuration files.

System Administration Resources

For easy reference, here is a list of the guides and resources provided with your system and the specific focus and scope of each:

Personal System Administration Guide
Covers all activities that can be performed by the end user, including those administrative activities performed using the System Manager. For example, adding a printer using the System Manager is covered, as well as how to back up specific files and directories. This guide is available through the InSight online viewing system.

IRIX Advanced Site and Server Administration Guide
Covers all activities that may be necessary to administer a system or group of systems at one site. (A site is any place where all the systems are connected or are all used by the same organization.) This guide is available through the InSight online viewing system.

Trusted IRIX/B Security Administration Guide
Covers the specific administration of the Trusted IRIX/B operating system security features. No other guide mentions the special features of this operating system. Standard IRIX systems do not support these features and this guide is not shipped with standard IRIX systems.

Other Administration Guides
You may have other administration guides for optional products that are not covered in the standard documentation set. Each of these guides is product specific.

*Software Installation Administrator's Guid*e
The Software Installation Administrator's Guide explains how to use Inst, the command line interface to inst(1M), the Silicon Graphics installation utility. This guide explains Silicon Graphics software release conventions and software product structure and provides clear instructions for

planning, installing, and maintaining a software installation. Installation procedures cover miniroot and live installations on all models of personal workstations and servers.

Reference Pages

Provide concise reference information on the use of commands. Generally, each reference page covers one command, although some reference pages cover several closely related commands. Reference pages are available online through the *man* command.

Release Notes

Provide specific information about the current release. Exceptions to the administration guides are found in this document. Release Notes are available online through the *relnotes* command.

When you have an administration question or problem, first consider the nature of your problem and compare it with the books on this list. As you learn more about your IRIS workstation or server, you'll be able to select the correct documentation automatically.

Note to Readers

This guide contains material from five guides that are no longer being published:

- The *Network Communications Guide*
- The *TCP/IP User's Guide*
- The *Network Administration Guide*
- The *System Tuning and Configuration Guide*
- The *IRIX Site Administrator's Guide*

Audience

This guide is intended for administrators who manage one or more servers or a group of workstations. Most simple system administration on an individual graphics workstation can be performed by the user with System Manager. This tool is documented thoroughly in the *Personal System Administration Guide*. The *IRIX Advanced Site and Server Administration Guide* is written for the administrator who:

- has a solid understanding of the UNIX operating system and command line interface
- is moving into the area of network administration
- is experienced with general network administration but needs specific knowledge about Silicon Graphics networking implementations
- is responsible for setting up and managing a new IRIS network
- is responsible for integrating IRIS systems into an existing network

This guide is *not* written for users who simply want to attach their workstation to the network. If this is your goal, see the *Personal System Administration Guide* for easy-to-follow directions.

Style Conventions

This guide follows these conventions:

- In command syntax descriptions and examples, square brackets ([]) surround an optional argument. (Square brackets are also used with shell commands as metacharacters, see "Using Regular Expressions and Metacharacters" on page 13.)
- Variable parameters are in *italics*. You replace these variables with the appropriate string or value.
- In text descriptions, file names, IRIX commands, and Command Monitor commands are in *italics*.
- System messages and displays are shown in `typewriter font`.
- **`Bold typewriter font`** is for user input and non-printing characters. For example: **`<Return>`**.

This guide uses the standard UNIX convention for referring to entries in IRIX documentation. The entry name is followed by a section number in parentheses. For example, *rcp*(1C) refers to the *rcp* online reference page.

Product Support

Silicon Graphics, Inc. provides a comprehensive product support and maintenance program for hardware and software products. For further information, contact your service organization.

Bibliography and Suggested Reading

Internet Request For Comment documents are available from the Internet Network Information Center (INTERNIC) at the following address:

Network Solutions
Attn: InterNIC Registration Services
505 Huntmar Park Drive
Herndon, VA 22070
Phone: 1-800-444-4345 or 1-703-742-4777

Bach, M., *The Design of the UNIX Operating System* (Englewood Cliffs, NJ:Prentice Hall, 1986).

Braden, R. "Requirements for Internet Hosts." *Internet Request For Comment 1112 (*1989).

Costales, B., *sendmail.* (Sebastopol, CA: O'Reilly & Associates, Inc., 1993).

Deering, S. "Host Extensions for IP Multicasting." *Internet Request For Comment 1112 (1989).*

Everhart, C., Mamakos, L., Ullmann, R., Mockapetris, P. "New DNS RR Definitions." *Internet Request For Comment 1183* (1990)

Fiedler, D., Hunter, B., *UNIX System V Release 4 Administration* (Carmel, IN: Hayden Books, 1991).

Frisch, A., *Essential System Administration.* (Sebastopol, CA: O'Reilly & Associates, Inc., 1991).

Gilly, D., *UNIX in a Nutshell.* (Sebastopol, CA: O'Reilly & Associates, Inc., 1992).

Hunt, C., *TCP/IP Network Administration.* (Sebastopol, CA: O'Reilly & Associates, Inc., 1992).

Leffler, S., *The Design and Implementation of the 4.3 BSD UNIX Operating System.* (Menlo Park, CA: Addison Wesley, 1989).

Lottor, M. "Domain Administrator's Guide." *Internet Request For Comment 1033 (*1987).

Lottor, M. "TCP Port Service Multiplexer (TCPMUX)." *Internet Request For Comment 1078 (*1988).

Mockapetris, P. "DNS Encoding of Network Names and Other Types." *Internet Request For Comment 1101 (*1989).

Mockapetris, P. "Domain Names – Concept and Facilities." *Internet Request For Comment 1034 (*1987).

Mockapetris, P. “Domain Names – Implementation and Specification.” *Internet Request For Comment 1035 (*1987).

Mogul, J., Postel, J. “Internet Standard Subnetting Procedure.” *Internet Request for Comment 950* (1985).

Nemeth, E., Snyder, G., Sebass, S., *UNIX System Administration Handbook* (Englewood Cliffs, NJ: Prentice Hall, 1989).

Partridge, C. “Mail Routing and The Domain System.” *Internet Request For Comment 974 (*1986).

Stahl, M. “Domain Administrator’s Guide.” *Internet Request For Comment 1032 (*1987).

Thomas, R., *UNIX System Administration Guide for System V.* (Englewood Cliffs, NJ: Prentice Hall, 1989).

Chapter 1

System Administration Basics

Chapter 1 introduces you to the basics of effective system administration. The basic tools that you will use are described here, a quick reference to each of the following chapters and a thumbnail guide to the IRIX manual pages is also included.

Chapter 1

System Administration Basics

The system administrator is responsible for all tasks that are beyond the scope of end users, whether for system security or other reasons. The system administrator will undoubtedly use the more advanced programs described in this guide.

A system administrator has many varied responsibilities. They can be organized into the following categories:

- Operations—seeing that the machine stays up and running, scheduling preventive maintenance downtime, adding new users, installing new software, and updating the */etc/motd* and */etc/issue* files. See Chapter 2, "Operating the System." Also see Chapter 3, "User Services."
- Failure Analysis—troubleshooting by reading system logs and drawing on past experience. See "Operating Policies" on page 56. Also see "System and Site Policies" on page 76.
- Capacity Planning—knowing the general level of system use and planning for additional resources when necessary. See Chapter 7, "Disks and Tape Drives" and Chapter 5, "Tuning System Performance."
- System Tuning—tuning the kernel and user process priorities for optimum performance. See Chapter 5, "Tuning System Performance" and Appendix A, "IRIX Kernel Tunable Parameters."
- Resource Management—planning process and disk accounting and other resource sharing. See Chapter 14, "System Accounting" and "Disk Use and Quotas" on page 76.
- Networking— interconnecting machines, modems, and printers. See Chapter 15, "Understanding Silicon Graphics' Networking Products."
- Security—maintaining sufficient security against break-ins as well as maintaining internal privacy and system integrity. See Chapter 12, "System Security."
- User Migration—helping users work on all workstations at a site. See Chapter 15, "Understanding Silicon Graphics' Networking Products."

- User Education—helping users develop good habits and instructing them in the use of the system. See Chapter 3, "User Services."
- Backups—creating and maintaining system backups. See Chapter 6, "Backing Up and Restoring Files."

Superuser Account

Most system administration is performed while the system administrator is logged in as *root* (the superuser). This account is different from an ordinary user account because *root* has access to all system files and is not constrained by the usual system of permissions that controls access to files, directories, and programs. The *root* account exists so that the administrator can perform all necessary tasks on the system while maintaining the privacy of user files and the sanctity of system files. Other operating systems that do not differentiate between users have little or no means of providing for the privacy of users' files or for keeping system files uncorrupted. UNIX systems place the power to override system permissions and to change system files only with the *root* account.

All administrators at your site should have regular user accounts for their ordinary user tasks. The *root* account should be used only for necessary system administration tasks.

Administration Tools

Depending on the exact configuration of your system, you may have the following tools available for performing system administration:

System Manager
: This tool, available on graphics workstations, provides easy access to system administration functions. It features a quick and easy method of performing most system administration tasks. The System Manager is available only on those systems that have graphics capability.

Command-line tools
: The IRIX system provides a rich set of system administration tools that have command-line interfaces.

These are especially useful for automatically configuring systems with shell scripts and for repairing the system in unusual circumstances, such as when you must log in remotely from another system.

For example, using command-line tools, a site administrator can alter the system automatically at designated times in the future (for instance, to distribute configuration files at regular intervals). These commands are available on all IRIX systems.

The IRIX Reference Pages

IRIX reference pages, also known as *man pages*, describe commands, subroutines, and other elements that make up the IRIX operating system. This collection of entries is one of the most important references for a site administrator.

The reference pages are available online. To view a reference page, use the *man* command at the shell prompt. For example, to see the reference page for *diff*, enter:

```
man diff
```

It is a good practice to print those reference pages you consistently use for reference and those you are likely to need before major administrative operations and keep them in a notebook of some kind.

Each command, system file, or other system object is described on a separate page. The reference pages are divided into seven sections, as shown in Table 1-1. When referring to reference pages, this document follows a standard UNIX convention: the name of the command is followed by its section number in parentheses. For example, *cc*(1) refers to the *cc* reference page in Section 1.

Table 1-1 shows the reference page sections and the types of reference pages that they contain.

Table 1-1 Outline of Reference Page Organization

Type of Reference Page	Section Number
General Commands	(1)
System Calls and Error Numbers	(2)
Library Subroutines	(3)
File Formats	(4)
Miscellaneous	(5)
Demos and Games	(6)
Special Files	(7)

Chapter 2

Operating the System

Chapter 2 describes the basic procedures used to operate the system on a day-to-day basis. Look to this chapter when you need information on the most basic administration tasks, such as:

- *Booting and shutting down an IRIX system.*
- *Checking the system configuration.*
- *Changing system defaults.*
- *Managing processes and system resources.*
- *Using command shortcuts.*
- *Using system utilities to automate tasks.*

Chapter 2

Operating the System

This chapter deals with the day-to-day operations of an IRIS workstation or server running the IRIX Operating System. It covers:

- Starting an IRIX system. See "Starting the System" on page 10.
- Shutting down an IRIX system. See "Shutting Down the System" on page 11.
- Using regular expressions and metacharacters. See "Using Regular Expressions and Metacharacters" on page 13.
- Using common shortcuts that are built in to the IRIX system to make shell command system administration more convenient. See "Using Shell Shortcuts in the IRIX System" on page 15.
- Creating a new shell window on a graphics workstation with customized colors and fonts. See "Creating a Custom Shell Window" on page 23
- Performing large scale operations on many files at once. See "Finding and Manipulating Files Automatically" on page 25 and "Automating Tasks with at(1), batch(1), and cron(1M)" on page 29.
- Using system utilities to automate tasks. See "Automating Tasks with at(1), batch(1), and cron(1M)" on page 29.
- Checking the hardware and software configuration. See "Checking System Configuration" on page 31.
- General operations, such as changing the system date or name, managing processes and system resources, reconfiguring various system defaults. and maintaining the File Alteration Monitor. See "General Operations" on page 33.
- General procedures, including 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. See "Operating Policies" on page 56.

- Definition of the operating system run levels; how they are controlled and how to change them. See “Operating Levels” on page 62.
- Recovering from a system crash and analyzing what has happened. See “Maintaining a System Log Book” on page 57.

Starting the System

To start up a system, follow these steps:

1. Make sure all cables (such as power, display monitor cable, and keyboard) are properly connected. See your owner’s guide and hardware guide for complete information about cabling your particular workstation or server.
2. Turn on the power switches on the display monitor (or console terminal) and the computer.

   The computer runs power-on diagnostics and displays some copyright messages and some system startup information. These messages appear on the *console* screen or on the screen of a *diagnostics terminal* (an ASCII terminal connected to the first serial port) of a server. A copy of these messages is also written to the */var/adm/SYSLOG* file in case you miss them.

   If the operating system determines that the file systems need checking, it checks them with the *fsck* program. *fsck* fixes any problems it finds before the operating system mounts the file systems. *fsck* will run if the system is not shut down properly, such as in the event of a power failure. For information about using *fsck*, see “Maintaining File Systems” on page 289 in this guide, and the *fsck*(1M) reference page. Note that it is not necessarily a problem if *fsck* runs, it is merely a precaution.

   The system now comes up in multiuser mode and you can log in. You should leave your system running at all times. The IRIX operating system works best when it is allowed to run continuously, so that scheduled operations and “housekeeping” processes can be performed on schedule.

Shutting Down the System

This section describes how to turn off a workstation or server from multiuser or single-user mode.

Shutting Down from Multiuser Mode

To shut down the system from multiuser mode:

1. Use the *who*(1) command to determine which users are logged in to the operating system, if any:

   ```
   who
   ```

   Notify any users that the system is shutting down. Issue the */etc/wall*(1M) command:

   ```
   wall
   ```

   Enter your message. For example, you might enter:

   ```
   There is a problem with the building's power system.
   I will be shutting down the system in 10 minutes.
   Please clean up and log off.
   Sorry for the inconvenience,
   norton
   ```

2. When you finish entering your message, type **<Ctrl-D>**. The message is sent to all users on the system. They see something like this:

   ```
   Broadcast Message from root Tue Oct 17 17:02:27...
   There is a problem with the building's power system.
   I will be shutting down the system in 10 minutes.
   Please clean up and log off.
   Sorry for the inconvenience,
   norton
   ```

3. Issue the */etc/shutdown* command:

   ```
   /etc/shutdown -y -i0 -g600
   ```

   The above command specifies a 10 minute (600 second) grace period to allow users to clean up and log off. The other flags indicate that the system is to be completely shut down (**-i0**) and that the system can assume that all answers to any prompts regarding the shutdown are "yes." You see the following message:

```
Shutdown started. Fri Aug 28 17:10:57...
Broadcast Message from root (console) Fri Aug 28 17:10:59
The system will be shut down in 600 seconds.
Please log off now.
```

After ten minutes, you see this message:

```
INIT: New run level: 0
The system is coming down. Please wait.
```

The Command Monitor prompt or System Maintenance menu appears. Wait for a Command Monitor prompt or System Maintenance menu to appear before turning off power to the workstation or you may damage your hard disk.

4. You can now turn off the power.

For more information on shutting down the system, see the *halt*(1M) and *shutdown*(1M) reference pages. Remember that you should shut down the system only when something is wrong or if modifications to the software or hardware are necessary. IRIX is designed to run continuously, even when no users are logged in and the system is not in use.

Turning Off from Single-user Mode

If the system is in single-user mode, follow these steps:

1. Use the *shutdown* command to turn off the system and guarantee file system integrity. As *root*, enter the command:

   shutdown -y -i0 -g0

   where:

   -y assumes yes answers to all questions, **-i0** goes to state 0 (System Maintenance Menu), and **-g0** allows a grace period of 0 seconds.

   You see a display similar to this one:

   ```
   Shutdown started. Fri Aug 28 17:11:50 EDT 1987
   INIT: New run level: 0
   The system is coming down. Please wait.
   ```

   The system stops all services and the Command Monitor prompt or System Maintenance Menu appears.

Wait for the Command Monitor prompt or System Maintenance menu to appear or for a message that the system can be powered off before turning off power to the computer. Doing so prematurely may damage your hard disk.

2. Turn off power to the computer.

Using Regular Expressions and Metacharacters

There are shortcuts available to you when you wish to define large numbers of files or directories in your IRIX shell commands. These shortcuts are known as "regular expressions." Regular expressions are made up of a combination of alpha-numeric characters and a series of punctuation characters that have special meaning to the IRIX shells. These punctuation characters are called metacharacters when they are used for their special meanings with shell commands. The following is a list of the IRIX metacharacters:

Table 2-1 IRIX Metacharacters

Metacharacter	Meaning
*	wildcard
?	single character wildcard
[]	set definition marks

The asterisk (*) metacharacter is a universal wildcard. This means that the shell interprets the character to mean any and all files. For example, the command:

```
cat *
```

tells the shell to concatenate all the files in a directory, in alphabetical order by filename. The command:

```
rm *
```

tells the shell to remove everything in the directory. Only files will be removed, though, since a different command, *rmdir*(1) is used to remove directories. However, the asterisk character does not always have to refer to

whole files. It can be used to denote parts of files as well. For example, the command:

```
rm *.old
```

will remove all files with the suffix.*old* on their names.

The single character wildcard is a question mark (?). This metacharacter is used to denote a wildcard character in one position. For example, if your directory contains the following files:

```
file1
file2
file3
file.different
```

and you wish to remove *file1*, *file2*, and *file3*, but not *file.different*, you would use the command:

```
rm file?
```

If you used an asterisk in place of the question mark, all your files would be removed, but since the question mark is a wildcard for a single space, your desired file is not chosen.

Square brackets denote members of a set. For example, consider the list of files used in the example of the single character wildcard. If you wanted to remove *file1* and *file2*, but not *file3* or *file.different*, you would use the following command:

```
rm file[12]
```

This command tells the shell to remove any files with names starting with *file* and with the character *1* or *2* following, and no other characters in the name. Each character in the brackets is taken separately. Thus, if our example directory had included a file named *file12*, it would not have been removed by the above command. You can also use a dash (-) to indicate a span of characters. For example, to remove *file1*, *file2*, and *file3*, use the following command:

```
rm file[1-3]
```

Alphabet characters can be spanned as well, in alphabetical order. The shell does pay attention to upper case and lower case letter, though, so to select all alphabet characters within square brackets, use the following syntax:

```
[a-z,A-Z]
```

You can use the square brackets in combination with other metacharacters as well. For example, the command:

```
rm *[2,3]
```

removes any files with names ending with a *2* or *3*, but not *file1* or *file.different*.

Using Shell Shortcuts in the IRIX System

IRIX provides a number of facilities that can be used for your convenience as you administer and use the system. Some of these shortcuts are available through the command shells (*/usr/bin/csh*, */usr/bin/tcsh*, */usr/bin/sh* and */usr/bin/ksh*) and others are made available through the computer hardware itself. These shortcuts are useful because they minimize keystrokes. While minimizing keystrokes may seem to be a minor concern at first glance, an administrator who issues lengthy and complex command lines repeatedly may find these shortcuts a handy and necessary time-saving feature.

C Shell

The IRIX C Shell (*/bin/csh*) provides several features that can be used to minimize keystrokes for routine tasks. Complete information about these and many other features of the C Shell is available in the *csh*(1) reference page. Among the features provided are:

filename completion

This feature is activated with the command:

```
set filec
```

Filename completion allows you to enter the first character or two of a command or file name and then press the **Escape** key to have the shell complete the name. This is useful when you have long filenames with many suffixes.

If more than one file or directory or command matches the characters you have given, the shell completes as much as possible of the name, and then prompts you with a beep for more information. You can also use the **<Ctrl-D>** character to select all files or directories that match your given characters.

shell scripts
: This feature allows you to create a program that will be executed by the shell. This feature is similar to a programming language in that it has a set syntax and set of instructions, yet it requires no compiler and produces no object file; it is directly executed by the shell. Many administrators use this feature for frequently performed procedures that require some planning and complex execution, such as finding large files and notifying the owners that such files cannot be kept on the system for long periods of time. The shell script programming rules are clearly presented on the *csh*(1) reference page.

input/output redirection
: This feature allows you to direct the output of a command into a file or into another command as input. You can also direct a command to take its input from a file. It is often used as part of a shell script, but is generally used on the command line to string together a series of commands. For example, consider the command line:

  ps -ef | grep *commandname*

  The pipe character directs the shell to use the output of the *ps* command as the input to the *grep* command. The result is that all instances of the command *commandname* in the process list are printed on the screen, saving the administrator the effort of searching through the process listing.

job control
: This feature allows you to use a single screen (or shell window) to manage several programs running simultaneously. It is most useful for the server administrator who manages the system from a single character-based terminal.

command aliasing

This feature allows you to create aliases for commonly used command strings, saving keystrokes. For example, suppose you frequently give the command:

```
ls -CF | more
```

This command line executes the *ls* command with certain options and ensures that if the output is greater than a screenful it will be stopped until you have read it. However, it would be tedious to type the whole command each time you wanted to see a directory listing in your preferred format. Therefore, you should create an alias. You can alias the above command line to any series of keystrokes you like. You can even alias it to "ls," thus bypassing the standard meaning of the *ls* command.

When you create the alias, however, be aware that any command that requires one or more arguments, or one such as *ls* that may or may not receive arguments, must have a provision made in the alias for those arguments. The standard provision made in aliases for possible arguments is the following regular expression:

```
\!*
```

The leading backslash escapes the initial meaning of the exclamation point to the shell and passes the exclamation point through to the command line, where it is interpreted by the shell to refer to arguments given on the aliased command line. The asterisk in the expression means that all characters typed in as arguments are to be passed through to the shell. As an example, the line you place in your *.cshrc* file to create the example alias is:

```
alias ls "ls -CF \!* | more"
```

Then, when you type the command:

ls *filename*

at your shell prompt, the command is executed as:

ls -CF *filename* **| more**

Aliases can be used freely within shell scripts, with filename completion and full use of regular expressions and output redirection.

command history
: The shell maintains a log of your past commands given during this login session. You can repeat or edit a previously given command to save keystrokes. The history command shows the numbered log of commands in order. The first command given in your login session is number 1, the second is number 2, and so on. You can set the number of commands the shell remembers in your *.cshrc* file. To execute the most recent command again, type:

  !!

  To execute the most recent command beginning with the letter "q," use the command line:

  !q

  And to execute a command by its number in the history, give the command line:

  !*n*

  where *n* is the number of the previous command you wish to re-execute.

Tcsh Shell

The */usr/bin/tcsh* program is an improved version of the C shell. In addition to the C shell features listed above, this shell offers many new features. A few of the most useful to system administrators are:

- Better command line editing using *emacs* and *vi* key commands.
- Improved history mechanisms, including time stamps for each command.
- Built-in mechanisms for listing directory contents and for executing commands at specific times or intervals.

There are many more features implemented in *Tcsh*, and all of them are covered in the *tcsh*(1) reference page.

Bourne Shell

The Bourne shell (*/bin/sh*) provides fewer features than the C shell, but in its place offers a level of access to the shell that contains far fewer restrictions and intervening layers of interface. For example, you can write shell script programs directly from the shell prompt with Bourne shell. Input and output redirection and command aliasing are supported with the Bourne shell, but no command history, job control, or filename completions are available. For a complete discussion of the Bourne shell and its features, see the *sh*(1) reference page.

Korn Shell

The Korn shell was developed to provide the best features of both the C shell and the Bourne shell. The */bin/ksh* program provides the ease of shell programming found in the Bourne shell, along with the job control, history mechanism, filename completion, and other features found in the C shell. This shell has changed many of the ways these features are implemented, and also provides improved command line editing facilities. See the *ksh*(1) reference page for complete information on this shell. Useful features include:

Emacs Editing	This mode is entered by enabling either the **emacs** or **gmacs** option. To edit, the user moves the cursor to the point needing correction and then inserts or deletes characters or words as needed as if the command line were a text file being edited using *Emacs*. All edit commands operate from any place on the line (not just at the beginning).
vi Editing	To enter this mode, enable the **vi** option. There are two typing modes in this option. Initially, when you enter a command you are in the input mode. To edit, the user enters control mode by typing *ESC,* moves the cursor to the point needing correction, then inserts or deletes characters or words as needed as if the command line were a text file being edited using *vi*.
job control	Lists information about each given process (**job**) or all active processes if the **job** argument is omitted. The **-l** flag lists process ID numbers in addition to the normal information. The **-n** flag only displays jobs that have

stopped or exited since last notified. The **-p** flag causes only the process group to be listed. See the *ksh*(1) reference page for a description of the format of the **job** argument.

The *bg* command puts each specified process into the background. The current process is put in the background if **job** is not specified.

The *fg* command brings each process specified to the foreground. Otherwise, the current process is brought into the foreground.

Mouse Shortcuts

The system hardware for graphical workstations (and some X-terminals) can provide you with shortcuts. These may not be available to server administrators who rely solely on character-based terminals for their administration. Using the graphics console of your system, you can cut and paste between windows without using pull-down or pop-up menus of any sort. Using the pop-up menu, you can manipulate your windows completely.

Note that you can customize the action of your mouse buttons. All examples in this section assume the default mouse button meanings are being used. If you have modified your mouse action, you must allow for that modification before you use these techniques.

For complete information on using the pop-up windows, see your *IRIS Essentials* book, either in hard copy or on screen through the IRIS InSight software package.

Using the Mouse to Copy and Paste Text

The most common mouse shortcut is to cut, copy, and paste between windows on your screen. Here is how you do it:

1. Find the cursor controlled by your mouse on your screen. It should appear as a small arrow when it is positioned in the working area of one of your windows, or as an "X" when it is positioned on your background screen, or as some other figure when it is positioned on the

frame of a window or in the working area of an application's window. If you can't locate the cursor immediately, move the mouse around a bit and look for motion on your screen. You should find the cursor easily.

2. Place the cursor at the beginning of the text you wish to paste between windows and press the leftmost key on the top of the mouse. Now, keeping the mouse button depressed, move the cursor to the end of the text you wish to paste. The intervening area of the window changes color to show the selected text. If you are selecting a large section of text, it is not necessary to move the cursor over every space. You may move the cursor directly to the end point and all intervening text will be selected. It is not possible to select "columns" of text or several disconnected pieces of text at once. When you have moved the cursor to the desired end point, release the mouse button. The text remains highlighted.
3. Now move the cursor to the window you want to paste the text into and make certain the window is ready to receive the pasted text. For example, if you are pasting a long command line, make certain that there is a shell prompt waiting with no other command already typed in. If the pasted matter is text into a file, make certain that the receiving file has been opened with an editor and that the editor is in a mode where text can be inserted.
4. To paste the text, place the cursor in the receiving window and press the middle mouse button once quickly. Each time you press the middle button, the selected text will be pasted into the receiving window. Sometimes it takes a moment for the text to appear, so be patient. If you press the button several times before the text appears, you will paste several copies of your text.
5. You can also paste your selected text to the bottom of a window (including the window from which you selected the text). Press the rightmost mouse button while the cursor is in that window and select the *send* option from the pop-up menu that appears.

The text you originally selected remains selected until you select new text somewhere else or until you place the cursor back in the original window and click once on the leftmost mouse button.

Using the Mouse to Create a New Shell Window

If you need a new shell window, you can use the mouse to create one. Follow these steps:

1. With the cursor in a shell window, press the rightmost button on your mouse. A pop-up menu appears:

Figure 2-1 Shell Pop-Up Menu

2. The last item on the pop-up menu is the **clone** option. There is a small triangle to the right of this option. This triangle indicates that there are more sub-choices available in another pop-up menu. While keeping the button on the mouse depressed, move the mouse down until the **clone** option is highlighted and the sub-menu pops up, showing various shell window cloning options. These options create another shell window functionally identical to the one in use. This is why the option is called cloning. The text and background colors of the current window are carried forward to the cloned window, and the selections in the sub-menu specify the number of lines in the new window. You can choose to have the same number of lines in the cloned window as in the current window, or to have 24, 40, or 60 lines.

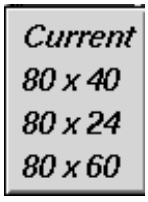


Figure 2-2 Shell Window Cloning Submenu

3. Select the size you want by moving the mouse down to highlight each option and releasing the mouse button when the option you desire is highlighted. The new window will appear on your screen presently. You may repeat this process as often as you like on any shell window.

Creating a Custom Shell Window

IRIX allows you to create a shell window using any colors you like from the palette on your graphics workstation. You may also select any font you prefer from the font set on your system. The *xwsh*(1) command creates the shell window, and the options to this command control the various fonts, colors, and other features available to you.

For a complete list of the features available with *xwsh*(1), see the *xwsh* reference page. The most commonly used features are described here in the following examples.

To create a simple shell window with a dark gray background and yellow text, issue the following command:

```
xwsh -fg yellow -bg gray40 &
```

The above command generates a new window and a new shell using the colors specified. The window will use the default font selection and window size, since these attributes were not specified. The command that created the shell was placed in the background, so the shell does not tie up the window where you gave the command. You can always place a command in the background by adding the ampersand character (&) to the end of the command line. For more information on placing processes in the background, see the *csh*(1) reference page.

There are 100 shades of gray available. Gray0 is the darkest, and is virtually black. Gray100 is the lightest and is virtually white. The effect of selecting foreground (text) in yellow and background in gray40 is similar to yellow chalk on a gray chalkboard. For a complete list of the available colors in your palette, use the *colorview*(1) command. This brings up a window with the list of colors in a scrollable list, and a display window to show a patch of the currently selected color.

In the next example, we change the colors to black on a sky blue background (high contrast between the foreground and background makes reading the screen easier), and we add a specification for the size of the window.

```
xwsh -fg black -bg skyblue -geometry 80x40 &
```

The first number in the *geometry* option is 80, indicating that the new shell window should be 80 characters wide (this is the default). The second number indicates the desired number of lines on the screen, in this case 40. Once again, the *xwsh* command has been placed in the background by adding the ampersand character to the end of the command line.

You can make your new shell come up on your desktop as an icon by adding the **-iconic** flag to any *xwsh* command.

To select a font other than the default, you can use the on-screen font selection utility, or you can specify the font on the command line. It is a great deal easier to use the on-screen font selection utility, as you must specify a great number of attributes for the font on the command line. Also, it frequently takes a great number of selections before you settle on a font, a weight (regular or bold, condensed or normal) and a font size that appeals to you. Using the on-screen font utility, you can preview what each selection will look like on your windows.

Once you have made your selections, you can copy and paste the font selection information and the rest of your *xwsh* command into a shell script file for convenient future use. For example, here is an *xwsh* command line that specifies the IRIS-specific font *haebfix* in a medium weight with normal spacing, 15 pixels tall. The remaining information is generated by the font selection utility for the shell.

```
xwsh -iconic -fg yellow -bg grey40 -geometry 80x40 -fn \ -
sgi-haebfix-medium-r-normal--15-150-72-72-m-90-iso8859-1 &
```

Note that in your shell script, the above command appears all on one line. Due to formatting constraints, the command is broken across two lines in this example.

For complete information on using the font selection utility in *xwsh* and the *xfontsel*(1) command, see Chapter 2 of the *IRIS Utilities Guide*.

Finding and Manipulating Files Automatically

The IRIX system provides several tools for manipulating large numbers of files quickly. Some of the most common are described below:

- The *find*(1) utility locates files and can invoke other commands to manipulate them.
- The *sed*(1) program edits files using pre-determined commands.
- Many other programs have recursive options, with which you can quickly operate on files that are in many levels of subdirectories.

Using find to Locate Files

The *find*(1) command is used to find files and possibly execute commands on the files found. It starts at a given directory and searches all directories below the starting directory for the specified files. A basic *find* command line looks like this:

```
find . -name <filename> -print
```

This command searches from the current directory until it finds the exact **<filename>** and then displays its location on your screen. You can also use regular expressions (see "Using Regular Expressions and Metacharacters" on page 13) in your searches.

The following command line searches for files that have been changed after the time */tmp/file* was last modified. If you use *touch*(1) to create */tmp/file* with an old date, this command can help you find all files changed after that date.

```
find / -local -newer /tmp/file -print
```

You can use *find* to locate files and then to run another command on the found files. This example shows how to locate a file in a user's directory:

```
cd /usr/people/trixie
find . -name 'missingfile' -print
```

In this example, the period (.) indicates the current directory, the *-name* option indicates that the next argument in quotes is the name of the file you are looking for, and the *-print* option tells *find* to display the pathname of the file when the file is located.

The next example shows how to change the permissions on all the files in the current directory and in all subdirectories:

```
find . -name '*' -local -exec chmod 644 {} \;
```

The option immediately following the find command is a period (.). This indicates to *find* that the search is to begin in the current directory and include all directories below the current one. The next flag, **-name**, indicates the name of the files that are being found. In this case, all files in the directory structure are selected through the use of the asterisk metacharacter (*). See "Using Regular Expressions and Metacharacters" on page 13 for more information on metacharacters and regular expressions.

The **-local** option indicates to *find* that the search is limited to files that physically reside in the directory structure. This eliminates files and directories that are mounted via the Network File System (NFS). The *-exec* option causes *find* to execute the next argument as a command, in this case *chmod 644*. The braces, { }, refer to the current file that *find* is examining.

The last two characters in the command line are part of the *chmod* command that will be executed (with the **-exec** option) on all files that match the search parameters. The backslash (\) is necessary to keep the shell from interpreting the semicolon (;). The semicolon must be passed along to the *chmod* process. The semicolon indicates a carriage return in the *chmod* command.

find has several other useful options:

-inum *n* Locate files by their inode number (*n*) instead of their name.

-mtime *n* Identify files that haven't been modified within a certain amount of time (*n*).

-perm [**-**]|**onum**
Identify files with permissions matching *onum*, an octal number that specifies file permissions. See the *chmod*(1) reference page. Without the minus sign (-), only file permissions that match exactly are identified.

If you place a minus sign in front of *onum*, only the bits that are actually set in *onum* are compared with the file permission flags.

-type *x*	Identifies files by type, where *x* specifies the type. Types can be *b, c, d, l, p, f,* or *s* for block special file, character special file, directory, symbolic link, FIFO (a named pipe), plain file, or socket respectively.
-links *n*	Matches files that have *n* number of links.
-**user** *uname*	Identifies files that belong to the user *uname*. If *uname* is a number and does not appear as a login name in the file */etc/passwd*, it is interpreted as a user ID.
-**group** *gname*	Identifies files that belong to the group *gname*. If *gname* is a number and does not appear in the file */etc/group*, it is interpreted as a group ID.
-**size** *n* [*c*]	Identifies files that are *n* blocks long (512 bytes per block). If you place a *c* after the *n*, the size is in characters.
-**ok** *cmd*	Works like *-exec*, except a question mark (?) prompts you to indicate whether you want the command (*cmd*) to operate on the file that is found. This is useful for such operations as selectively removing files.

Copying Directories or Directory Hierarchies

The *find* and *cpio* commands can be used to easily and safely copy a directory or a directory hierarchy as long as the user has permissions to access the directory. To copy a directory with all its files, or an entire hierarchy of directories and their files, use commands like the following:

```
mkdir new_directory_name
cd the_directory_you_want_to_copy
find . -print | cpio -pdlmv new_directory_name
```

This command sequence preserves the symbolic links in the new directory as well as transparently cross file system boundaries.

Automated Editing with sed

You can use *sed*(1), the Stream Editor, to automate file editing. *sed* follows an editing script that defines changes to be made to text in a file. *sed* takes a file (or files), performs the changes as defined in the editing script, and sends the modified file to the standard output. *sed* is fully described in the IRIX Development Option documentation and in the *sed*(1) reference page, which is included in your IRIX distribution. The IRIX Development Option is available for separate purchase from Silicon Graphics.

Some Recursive Commands

Recursive commands can save you a lot of time. For example, to change the ownership of all the files and directories in a directory recursively, and all of the files and directories in all of the subdirectories below that, you can use the recursive option with *chown*(1):

`chown -R` *username directory*

Some of the other commands in the IRIX system that have recursive options are:

`ls -R`

`rm -r`

`chgrp -R`

If you want to use a particular command recursively, but it does not have a recursive option, you can run the command using *find*. See “Using find to Locate Files” on page 25.

Note that using recursive options to commands can be very dangerous in that the command automatically makes changes to your files and file system without prompting you in each case. The *chgrp* command can also recursively operate up the file system tree as well as down. Unless you are sure that each and every case where the recursive command will perform an action is desired, it is better to perform the actions individually. Similarly, it is good practice to avoid the use of metacharacters (described in “Using Regular Expressions and Metacharacters” on page 13) in combination with recursive commands.

Automating Tasks with at(1), batch(1), and cron(1M)

You can use the *at*(1), *batch*(1), and *cron*(1M) utilities to automate many of your usual tasks, both as an administrator and as a user. These utilities perform similar functions. All execute commands at a later point in time. The difference between the commands is that *at* executes the given command at one specific time; *cron* sets up a schedule and executes the command or commands as often as directed, according to the schedule; and *batch* executes the commands when system load levels permit the execution.

at(1) Command

If you have a task to process once at a later point in time, use *at*. For example, if you wish to close down permissions on a public directory at midnight of the current day, but you do not need to be present when this occurs, you could use the command string:

```
at 2400 July 14
chmod 000 /usr/public
<Ctrl-D>
```

It is required that the *at* command itself and the date and time of the command be placed alone on a line. When you press **<Return>**, you do not see a prompt; *at* is waiting for input. Enter the command to be executed just as you would type it at a shell prompt. After entering the command, press **<Return>** again and enter **<Ctrl-D>** to tell *at* that no more commands are forthcoming. You can use a single *at* command to execute several commands at the appointed time. For example, if you want to close the public directory and change the message of the day to reflect this closure, you can create the new message of the day in the file */tmp/newmesg*, and then issue the following command string:

```
at 2400 July 14
chmod 000 /usr/public
mv /etc/motd /etc/oldmotd
mv /tmp/newmesg /etc/motd
<Ctrl-D>
```

By default, any output of commands executed using *at* is mailed to the executing user through the system electronic mail. You can specify a different location for the disposition of output by using the standard output redirects, such as pipes (|) and file redirects (>). See your command shell documentation for a complete description of redirecting the standard output.

For complete information on the *at* command, see the *at*(1) reference page.

batch(1) Command

The *batch* command works just like the *at* command, except that you do not specify a time for the command or commands to be executed. The system determines when the overall load is low enough to execute the commands, and then does so. As with all other *cron* subsystem commands, the output of the commands is mailed to you unless you specify otherwise. *batch* is useful for large CPU-intensive jobs that slow down the system or cripple it during peak periods. If the job can wait until a non-peak time, you can place it on the *batch* queue until the system executes it. For complete information on the *batch* command, see the *batch*(1) reference page.

cron(1M) Command

If you desire to have a command executed regularly on schedule, the *cron* command and subsystem provide a precise mechanism for scheduled jobs. The *at* and *batch* commands are technically part of the *cron* subsystem and use *cron* to accomplish their tasks. The *cron* command itself, though, is the most configurable command of the subsystem.

You use *cron* by setting up a *crontab* file, where you list the commands you would like to have executed and the schedule for their execution. Complete information on setting up your *crontab* file is available in the *cron*(1M) and *crontab*(1) reference pages.

cron is useful for scheduling network backups, checking the integrity of the password file, and any other scheduled tasks that do not require interaction between you and the system. By default, *cron* mails the results or output of the command to the user who submitted the *crontabs* file, so if you use *cron*

to schedule something like a *pwck*(1M), the results of the test are mailed to you and you can interpret them at your convenience.

Note that you must restart *cron* after each change to a *crontabs* file, whether made through the *cron* utility or the *at* command, for the changes to take effect.

Checking System Configuration

IRIX provides two commands that allow you to check your system hardware and software configurations. The *hinv*(1M) and *versions*(1M) commands display the hardware and software inventories, respectively.

hinv Command

The *hinv* command displays the machine's hardware inventory. *hinv* can be run from the Command Monitor or from your system shell prompt. Pertinent information such as the processor type, amount of main memory, and all disks, tape drives, or other devices is included. A sample *hinv* output for a typical workstation is:

```
1 100 MHZ IP22 Processor
FPU: MIPS R4010 Floating Point Chip Revision: 0.0
CPU: MIPS R4000 Processor Chip Revision: 3.0
On-board serial ports: 2
On-board bi-directional parallel port
Data cache size: 8 Kbytes
Instruction cache size: 8 Kbytes
Secondary unified instruction/data cache size: 1 Mbyte
Main memory size: 64 Mbytes
Vino video: unit 1, revision 1
Iris Audio Processor: version A2 revision 4.1.0
Integral Ethernet: ec0, version 1
CDROM: unit 4 on SCSI controller 0
Disk drive: unit 1 on SCSI controller 0
Integral SCSI controller 0: Version WD33C93B, revision D
Graphics board: Indy 24-bit
```

If a piece of peripheral hardware installed on your system does not appear in the *hinv* output, it may or may not be an indication of trouble with your

hardware. Some peripherals connected to the system by a board on a VME bus will not be identified when running *hinv* from the Command Monitor. First, you should invoke *hinv* from a system shell prompt; If your peripheral is still not recognized, attempt to reseat the board in its socket and check that it is using the correct SCSI address. If this does not relieve the problem, the hardware itself may be defective. Note also that most devices are not recognized by *hinv* until the *MAKEDEV*(1M) command has been run after their installation.

versions Command

The *versions* command gives you an inventory of software packages that have been installed using *inst*(1M). This command can only be run at the system shell prompt, not from the Command Monitor. Software installed by other means is not included in the *versions* output. Along with the names of the software products, the release revision level numbers are displayed. By default, the output of *versions* includes all the products and their subsystems and is typically several hundred lines long, so it is often convenient to redirect the output to a file that you can view at your convenience. For a more general look at the products you have installed, without the list of specific subsystems, use the **-b** (brief) flag.

A sample *versions -b* output reads as follows (an actual listing will be much longer):

```
I = Installed, R = Removed
   Name        Date      Description
I  4Dwm        04/29/93  4Dwm -- Default Window Manager, 5.3
I  demos       04/29/93  Graphics Demonstration Program, 5.3
I  desktop_eoe 04/29/93  Desktop Environment, 5.3
I  dps_eoe     04/29/93  Display PostScript, 2.0
I  eoe1        04/29/93  Execution Only Environment 1, 5.3
I  eoe2        04/29/93  Execution Only Environment 2, 5.3
I  insight     04/29/93  IRIS InSight Viewer, 2.1
I  motif_eoe   04/29/93  Motif Execution Only Environment
I  nfs         04/29/93  Network File System, 5.2
```

gfxinfo command

The *gfxinfo* command is useful for determining the graphics hardware installed in the system. It is in the */usr/gfx* directory, which is not on any of

the standard search paths. Thus *gfxinfo* typically needs full path name for execution. It requires no arguments to run.

An sample output for an Indy:

```
% /usr/gfx/gfxinfo

Graphics board 0 is "NG1" graphics.
        Managed (":0.0") 1280x1024
        24 bitplanes, NG1 revision 3, REX3 revision B,
        VC2 revision A
        MC revision C, xmap9 revision A, cmap revision C,
        bt445 revision A
        Display 1280x1024 @ 60Hz, monitor id 12
```

This command provides much more information about the graphics system than the *hinv* command (*hinv* would simply return Indy 24-bit). From the output of *gfxinfo* you can determine the number of screens and their pixel resolutions, bitplane configurations, component revision levels, and monitor types. There is no reference page for *gfxinfo*.

uname command

uname returns information such as the OS version and hostname. The *-a* options gives a complete list of the uname output. See the reference page for a description of all the *uname* options and fields.

lpstat command

lpstat with the *-a* option will show all the printers configured for the *lp* spooling system and also give their status. For more information on the *lpstat* command, see “lpstat: Report lp Status” on page 345 or the *lpstat* reference page.

General Operations

This section covers general system operations. Topics are:

- “Checking the System Configuration” on page 34.
- “Altering the System Configuration” on page 39.

Information on specific parts of the system is presented in other chapters. For example, information on system security is covered in Chapter 12, "System Security."

Also, defaults for tape and other backup operations (such as the default tape drive) are covered in Chapter 6, "Backing Up and Restoring Files." Defaults for shell types and user groups are covered in Chapter 3, "User Services."

Checking the System Configuration

You can quickly check the configuration of a workstation or server with the *chkconfig*(1) command. The */sbin/chkconfig* utility reports the state of various process daemons (that is, whether or not they are supposed to be active).

For example, enter the *chkconfig* command:

```
chkconfig
```

You see a display similar to this:

```
Flag                State
====                =====
acct                off
audit               off
automount           on
fmlicserv           off
gated               off
lockd               on
mrouted             off
named               off
network             on
nfs                 on
noiconlogin         off
nsr                 on
quotacheck          off
quotas              off
routed              on
rtnetd              off
rwhod               off
sar                 on
snmpd               on
timed               on
timeslave           off
verbose             off
visuallogin         on
windowsystem        off
yp                  on
ypmaster            off
ypserv              off
```

This example is typical for a networked workstation with the Network File System (NFS) option installed. The left column of the output describes a system feature, and the right column indicates whether it is on or off. The following list provides more specific information about each system feature:

acct	Detailed system accounting is turned on or off.
audit	The System Audit Trail is turned on or off.
automount	The NFS *automount*(1M) daemon is turned on or off. This configuration option is available only if you have NFS installed on the workstation.

gated	The *gated*(1M) daemon, which manages multiple routing protocols is turned on or off.
glb	This option is used by the NetLS license server for the global location broker daemon.
llb	This option is used by the NetLS license server for the local location broker daemon.
lockd	The Network File System (NFS) lock daemon is turned on or off. This configuration option is available only if you have NFS installed on the workstation.
mrouted	The Stanford IP multicast routing daemon is turned on or off.
named	*named*(1M), the Internet domain name server, is turned on or off.
network	The network is turned on or off.
nfs	NFS is turned on or off. This configuration option is available only if you have NFS installed on the workstation.
noiconlogin	The visual login program, *pandora*(1), displays icons that represent users on the system. This feature does not enable or disable *pandora*; it only affects whether or not *pandora* displays icons. It is turned on or off. To enable or disable *pandora*, use the *visuallogin* feature.
nsr	IRIS Networker backup utility. This configuration option is available only if you have Networker installed on the workstation.
quotacheck	The disk space quota checker is enabled or disabled.
quotas	Disk quotas are enabled or disabled.
routed	*routed*(1M), which manages the network routing tables, is turned on or off.
rtnetd	*rtnetd*(1M), which allows higher priority real-time processes to preempt processing of incoming network packets, is turned on or off.
rwhod	*rwhod*(1M) is turned on or off.
sar	*sar*(1), the system activity reporter, is turned on or off.

snmpd	The Simple Network Management Protocol Daemon is turned on or off.
timed	*timed*(1M), the 4.3 BSD time server daemon, is turned on or off.
timeslave	The Silicon Graphics time server daemon is turned on or off. Like *timed*, this attaches a workstation's clock to a different clock, usually some kind of master time server for a group of workstations or for the entire site.
verbose	If this feature is enabled, as the system boots or is shut down, daemons print information about their functions. If this feature is disabled, less information is printed when the system is started and shut down.
visuallogin	The visual login program, *pandora*(1), is turned on or off.
windowsystem	The window manager is turned on or off.
yp	The network information service (NIS) is enabled on or off. This is called "yp" for historical reasons. NIS is available with the NFS software. This configuration option is available only if you have NFS installed on the workstation.
ypmaster	NIS master services are turned on or off. This configuration option is available only if you have NFS installed on the workstation.
ypserv	NIS server and bind processes are turned on or off. This configuration option is available only if you have NFS installed on the workstation.

Note that if a daemon is enabled using *chkconfig*, it does not necessarily mean that the daemon starts up immediately, or that it is running successfully. To verify that a daemon is running, use the *ps*(1) command to identify what processes are running on the system. For example, the command:

```
ps -ef
```

produces output similar to this:

Table 2-2 *ps -ef* Output

UID	PID	PPID	C	STIME	TTY	TIME	COMMAND
root	0	0	0	Aug 3	?	0:00	sched
root	1	0	0	Aug 3	?	0:45	/etc/init
root	2	0	0	Aug 3	?	0:08	vhand
root	3	0	0	Aug 3	?	0:09	bdflush

This example is edited for simplicity. An actual, full *ps* listing shows many more active processes.

To view information about specific processes, and avoid searching through a large *ps* listing, you can filter the listing with the *grep*(1) or *egrep*(1) commands. For example, to look at process information for only the NFS daemons, use this command:

```
ps -ef | egrep 'nfsd|biod'
```

The output of this command is similar to this (assuming you have NFS installed and running):

```
root  120 1  0    09:40:05 ?     0:02 /usr/etc/nfsd 4
root  122 12 0 0  09:40:05 ?     0:02 /usr/etc/nfsd 4
root  123 12 0 0  09:40:05 ?     0:02 /usr/etc/nfsd 4
root  124 12 0 0  09:40:05 ?     0:02 /usr/etc/nfsd 4
root  126 1  0    09:40:05 ?     0:00 /usr/etc/biod 4
root  127 1  0    09:40:05 ?     0:00 /usr/etc/biod 4
root  128 1  0    09:40:05 ?     0:00 /usr/etc/biod 4
root  129 1  0    09:40:05 ?     0:00 /usr/etc/biod 4
root  131 1  0    09:40:11 ?     0:00 /etc/mount -at nfs
ralph 589 55 0 11 15:25:30 ttyq1 0:00 egrep nfsd|biod
```

Note that the final entry in the *ps* listing is the process that produced the listing and that it is the only non-*root* process to have *nfsd* or *biod* in its name.

Altering the System Configuration

You can use the *chkconfig* command to change some aspects of system configuration. To determine which aspects of a system you can alter with

chkconfig, enter the *chkconfig* command:

```
chkconfig
```

You see a list of configuration options, which are described in "Checking the System Configuration" on page 34 If you use the **-s** option, you see a list that is sorted by whether the configuration item is on or off.

To change a configuration option, use the *chkconfig* command with two arguments: the name of the option you wish to change and the new status of the configuration (on or off). You must have *root* privilege to change a system configuration.

For example, to turn on detailed process accounting, log in either as *root* or as the system administrator, and enter:

```
chkconfig acct on
```

To turn off process accounting, enter:

```
chkconfig acct off
```

Some aspects of system configuration do not take effect until the system is shut down and rebooted because startup scripts, which are in the directory */etc/init.d*, are run when the system is booted up and brought to multiuser mode. These scripts read the files that *chkconfig* sets to determine which daemons to start.

Some configuration items that can be controlled by *chkconfig* may not be displayed by *chkconfig*. These include:

nostickytmp Sets "sticky" behavior for the directory */tmp*. When the directory is sticky, (with *nostickytmp* set to **off**), users may not remove files from the directory unless they own the files, have explicit permission to remove the files (write permission), or have superuser privileges.

The opposite behavior allows users to remove or replace files in */tmp*, which is a publicly writable directory, even if they do not own the files. This is handy behavior if you have users who need to create large temporary files and you are short on disk space. But it is better to increase disk space to avoid important files being removed.

nocleantmp Controls whether or not the directory */tmp* is cleaned out each time the system is booted. If *nocleantmp* is on, */tmp* is not cleaned. If *nocleantmp* is off, all files in */tmp* are removed each time the system is started.

If you want to see these flags in the *chkconfig* menu, you can use the **-f** option to force *chkconfig* to create a configuration file for the options:

```
chkconfig -f nocleantmp on
```

In this example, *chkconfig* creates a configuration file called *nocleantmp* in the directory */etc/config*.

Checking the Password File

At least once a week, you should run the *pwck*(1M) and *grpck*(1M) programs to check your */etc/passwd* and */etc/group* files for errors. You can automate this process using the *cron*(1) command, and you can direct *cron* to mail the results of the checks to your user account. For more information on using *cron* to automate your routine tasks, see "Automating Tasks with at(1), batch(1), and cron(1M)" on page 29.

The *pwck* and *grpck* commands read the password and group files and report any incorrect or inconsistent entries. Any inconsistency with normal IRIX operation is reported. For example, if you have */etc/passwd* entries for two user names with the same User Identification (UID) number, *pwck* will report this as an error. *grpck* performs a similar function on the */etc/group* file. Note that the standard *passwd* file shipped with the system generates several errors.

Changing System Defaults

These system-wide defaults affect programs and system functions:

- the system display
- the time zone
- the name of the system
- the network address
- the default system printer

Some of these defaults are described more thoroughly in specific sections of this guide, but they are all presented here to provide a more thorough overview of the IRIX system.

Setting the System Display

You can make the output of programs and utilities running on one system appear on the screen of another system on the same network by changing the DISPLAY environment variable. This is useful if your network includes graphical systems and non-graphical servers. In order to view information from the server graphically, you reset the display to a graphics workstation.

For example, if your server has only a character-based terminal as its console and you wish to run *gr_osview*(1M) to visually inspect your CPU usage, you would issue commands similar to these on the server:

setenv DISPLAY *graphics_machine*:0

```
gr_osview
```

When you invoke *gr_osview*, the window with the output will appear on the machine name you specify. In this example, *graphics_machine* was used in place of the system name. The **:0** used after the machine name indicates that display monitor 0 (the graphics console) should be used to display the output. When you have finished using the graphics console, be sure to reset the display by issuing this command on the server:

setenv DISPLAY *local_server*:0

where *local_server* is the name of your server.

Setting the Time Zone

To set the time zone of the system, edit the file */etc/TIMEZONE*. For a site on the east coast of the United States, the file might look something like this:

```
# Time Zone
TZ=EST5EDT
```

The line `TZ=EST5EDT` means:

- The current time zone is Eastern Standard Time.
- It is 5 hours to the east of Greenwich mean time.
- Daylight saving time applies here (EDT).

The *TZ* environment variable is read by *init*(1) when the system boots, and the value of *TZ* is passed to all subsequent processes. For complete information about setting your time zone, see the *timezone*(4) reference page.

Changing Processors on Multi-Processor Systems

If you have a multi-processor system, the *mpadmin*(1M) and *pset*(1M) commands allow you to change the way programs are assigned to the various processors on your system. To determine if your system is multi-processor, use the *hinv*(1M) command. A multi-processor system returns information similar to the following in its *hinv* output:

```
Processor 0: 40 MHZ IP7
Processor 1: 36 MHZ IP7
Processor 2: 40 MHZ IP7
Processor 3: 40 MHZ IP7
Processor 4: 40 MHZ IP7
Processor 5: 40 MHZ IP7
Processor 6: 40 MHZ IP7
Processor 7: 40 MHZ IP7
```

Or, alternately, output similar to the following:

```
8 40 MHZ IP7 Processors
```

A single-processor system returns information similar to the following for the *hinv* command:

```
1 100 MHZ IP22 Processor
```

If you have only one processor on your system (and the vast majority of systems have only one processor) these commands still operate, though they have no useful purpose.

The *mpadmin* command allows you to ''turn off'' processors, report various states of the processors, and move system functions such as the system clock to specific processors. The *pset* command is used both to display and modify information concerning the use of processor sets and programs running in the current system. The *pset* command provides a much more detailed level of control of processes and processors.

For complete information on *mpadmin*(1M) and *pset*(1M), see the respective reference pages.

Changing the Name of a System

The name of the system is stored in several places. If you wish to change the name of your system, you must change all these files together or your system will not function correctly:

- in the file */etc/sys_id*
- in the file */etc/hosts* (for networking purposes)
- in a kernel data structure, which you read and set with either *hostname*(1) or *uname*(1)
- in an NIS map on the NIS master server, if you are running NIS

Note that you should not arbitrarily change the name of a running workstation. Many programs that are started at boot time depend on the name of the workstation.

To display the name of the system, use the *hostname* command with no arguments:

```
hostname
```

This displays the name of the system. The *uname* command also displays the name of the system, along with other information.

To change the name of the workstation, follow these steps:

1. Log in as **root**.
2. Edit the file */etc/sys_id*. Change the name of the host to *newname*. Write and exit the editor.
3. You must also change the name of the host in any network files, such as */etc/hosts*, and possibly in the NIS map on the master NIS server.
4. Reboot your system.

The name of the workstation is now changed. When the workstation is booted, all programs that are started at boot time, and read the host name when they start, now use the correct host name.

For information about the Internet address of a workstation, see Chapter 15, "Understanding Silicon Graphics' Networking Products." For more information about the name of the system, see the *hostname*(1) and *uname*(1) reference pages.

Setting the Network Address

The system's network address (IP address) is covered more thoroughly in Chapter 15, "Understanding Silicon Graphics' Networking Products."

To set the network address, follow these steps:

1. Place the network address in */etc/hosts* on the same line as the system name.
2. If you use the network information service (NIS), place the name of your domain in the file */var/yp/ypdomain*, if it is installed.
3. Use the *nvram*(1M) command to set the variable *netaddr* to the IP number of the machine. For example:

   ```
   nvram netaddr 192.13.52.4
   ```

Setting the Default Printer

The *lpadmin*(1M) command sets the default printer. This command sets the default printer to *laser*:

```
lpadmin -dlaser
```

Note that the printer *laser* must already exist and be configured. For complete information on setting up printers, see Chapter 9, "Administering Printers."

Managing User Processes

Just as files can use up your available disk space, too many processes going at once can use up your available CPU time. When this happens, your system response time gets slower and slower until finally the system cannot execute any processes effectively. If you have not tuned your kernel to allow for more processes, the system will refuse new processes long before it reaches a saturation point. However, due to normal variations in system usage, you may experience fluctuations in your system performance without reaching the maximum number of processes allowed by your system.

Monitoring User Processes

Not all processes require the same amount of system resources. Some processes, such as database applications working with large files, tend to be *disk intensive*, requiring a great deal of reading from and writing to the disk as well as a large amount of space on the disk. These activities take up CPU time. Time is also spent waiting for the hardware to perform the requested operations. Other jobs, such as compiling programs or processing large amounts of data, are *CPU intensive*, since they require a great number of CPU instructions to be performed. Some jobs are *memory intensive*, such as a process that reads a great deal of data and manipulates it in memory. Since the disk, CPU, and memory resources are limited, if you have more than a few intensive processes running at once on your system, you may see a performance degradation.

As the administrator, you should be on the lookout for general trends in system usage, so you can respond to them and keep the systems running as efficiently as possible. If a system shows signs of being overloaded, and yet the total number of processes is low, your system may still be at or above reasonable capacity. The following sections show four ways to monitor your system processes.

Monitoring Processes with top

top and *gr_top* are the most convenient utilities provided with IRIX to monitor the top CPU-using processes on your system. These utilities display the top such processes dynamically, that is, if a listed process exits, it is removed from the table and the next-highest CPU-using process takes its place. *gr_top* graphically displays the same information as *top*. If you are using a non-graphics server, you cannot use *gr_top* locally, but you can use it if you set the display to another system on the network that does have graphics capability. For complete information on configuring and using *top* and *gr_top*, consult the *top*(1) and *gr_top*(1) reference pages. For information on resetting the display, see "Setting the System Display" on page 41.

Monitoring Processes with osview

osview and *gr_osview* are utilities that display kernel execution statistics dynamically. If you have a graphics workstation, you can use the *gr_osview*(1) tool, which provides a real-time graphical display of system memory and CPU usage. *osview* provides the same information in ASCII format. You can configure *gr_osview* to display several different types of information about your system's current status. In its default configuration, *gr_osview* provides information on the amount of CPU time spent on user process execution, system overhead tasks, interrupts, and idle time. For complete information on *osview* and *gr_osview*, see the *osview*(1) and *gr_osview*(1) reference pages.

Monitoring Processes with sar

The System Activity Reporter, *sar*, provides essentially the same information as *osview*, but it represents a "snapshot" of the system status, not a dynamic reflection. Because *sar* generates a single snapshot, it is easily saved and can be compared with a similar snapshot taken at another time. You can use *sar* automatically with *cron* to get a series of system snapshots over time to help you locate chronic system bottlenecks. For complete information on *sar*, see the *sar*(1) reference page.

Monitoring Processes with ps

The *ps* -**ef** command allows you to look at all the processes currently running on your system. *ps* -**ef** output looks very similar to Table 2-3:

Table 2-3 Output of the *ps -ef* Command

Name	PID	PPID	C	Time	TTY	CPU Time	Process
joe	23328	316	1	May 5	ttyq1	1:01	csh

In this table, the process shown is for the user "joe." In a real situation, each user with processes running on the system is represented. Each field in the output contains some useful information.

Name	The login name of the user who "owns" the process.
PID	The process identification number.
PPID	The process identification number of the parent process that spawned or forked the listed process.
C	Current execution priority. The higher this number, the lower the scheduling priority. This number is based on the recent scheduling of the process and is not a definitive indicator of its overall priority.
Time	The time when the process began executing. If it began more than 24 hours before the *ps* command was given, the date on which it began is displayed.
TTY	The TTY (Terminal or window) with which the process is associated.
CPU	The total amount of CPU time expended to date on this process. This field is useful in determining which processes are using the most CPU time. If a process uses a great deal in a brief period, it can cause a general system slowdown.

For even more information, including the general system priority of each process, use the *-l* flag to *ps*. For complete information on interpreting *ps* output, see the *ps*(1) reference page.

Prioritizing Processes with *nice*

IRIX provides methods for users to force their CPU-intensive processes to execute at a lower priority than general user processes. The */bin/nice*(1) and *npri*(1M) commands allow the user to control the priority of their processes on the system. The *nice* command functions as follows:

```
nice [ -increment ] command
```

When you form your command line using */bin/nice*, you fill in the **increment** field with a number between 1 and 19. If you do not fill in a number, a default of 10 is assumed. The higher the number you use for the **increment**, the lower your process' priority will be (19 is the lowest possible priority; all numbers greater than 19 are interpreted as 19). The *csh*(1) shell has its own internal *nice* functions, which operate differently from the *nice* command, and are documented in the *csh*(1) reference page.

After entering the *nice* command and the increment on your command line, give the **command** as you would ordinarily enter it. For example, if the user "joe" wants to make his costly compile command described in the *ps -ef* listing above happen at the lowest possible priority, he forms the command line as follows:

```
nice -19 cc -o prog prog.c
```

If a process is invoked using *nice*, the total amount of CPU time required to execute the program does not change, but the time is spread out, since the process executes less often.

The superuser (*root*) is the only user who can give *nice* a negative value and thereby *increase* the priority of a process. To give *nice* a negative value, use two minus signs before the increment. For example:

```
nice --19 cc -o prog prog.c
```

The above command endows that process with the highest priority a user process may possess. The superuser should not use this feature frequently, as even a single process that has been upgraded in priority causes a significant system slowdown for all other users. Note that */bin/csh* has a built-in *nice* program that uses slightly different syntax than that described here. For complete information on *csh*, see the *csh*(1) reference page.

The *npri* command allows users to make their process' priority *nondegrading*. In the normal flow of operations, a process loses priority as it executes, so large jobs typically use fewer CPU cycles per minute as they grow older. (There is a minimum priority, too. This priority degradation simply serves to maintain performance for simple tasks.) By using *npri*, the user can set the *nice* value of a process, make that process non-degrading, and also set the default time slice that the CPU allocates to that process. *npri* also allows you to change the priority of a currently running process. The following example usage of *npri* sets all the possible variables for a command:

```
npri -h 10 -n 10 -t 3 cc -o prog prog.c
```

In this example, the *-h* flag sets the nondegrading priority of the process, while the *-n* flag sets the absolute *nice* priority. The *-t* flag sets the time slice allocated to the process. IRIX uses a 10-millisecond time slice as the default, so the example above sets the time slice to 30 milliseconds. For complete information about *npri* and its flags and options, see the *npri*(1) reference page.

Changing the Priority of a Running Process

The superuser can change the priority of a running process with the *renice*(1M) or *npri* commands. Only the superuser can use these commands. *renice* is used as follows:

```
renice increment pid [-u user] [-g pgrp]
```

In the most commonly used form, *renice* is invoked on a specific process that is using system time at an overwhelming rate. However, you can also invoke it with the *-u* flag to lower the priority of all processes associated with a certain user, or with the *-g* flag to lower the priorities of all processes associated with a process group. More options exist and are documented in the *renice*(1M) reference page.

The *npri* command can also be used to change the parameters of a running process. This example changes the parameters of a running process with *npri*:

```
npri -h 10 -n 10 -t 3 -p 11962
```

The superuser can use *renice* or *npri* to increase the priority of a process or user, but this can severely impact system performance.

Terminating Processes

From time to time a process may use so much memory, disk, or CPU time that your only alternative is to terminate it before it causes a system crash. Before you kill a process, make sure that the user who invoked the process will not try to invoke it again. You should, if at all possible, speak to the user before killing the process, and at a minimum you should notify the user that the process was prematurely terminated and give a reason for the termination. If you do this, the user can reinvoke the process at a lower priority or possibly use the system's job processing facilities (*at*, *batch*, and *cron*) to execute the process at another time.

To terminate a process, you use the *kill* command. Typically, for most terminations, you should use the *kill -15* variation. The *-15* flag indicates that the process is to be allowed time to exit gracefully, closing any open files and descriptors. The *-9* flag to *kill* terminates the process immediately, with no provision for cleanup. If the process you are going to kill has any child processes executing, using the *kill -9* command may cause those child processes to continue to exist on the process table, though they will not be responsive to input. The *wait*(1) command, given with the process number of the child process, removes them. For complete information about the syntax and usage of the *kill* command, see the *kill*(1) reference page. You must always know the PID of the process you intend to kill with the *kill* command.

Killing Processes by Name with the killall(1M) Command

The *killall*(1M) command allows you to kill processes by their command name. For example, if you wish to kill the program *a.out* that you invoked, use the syntax:

```
killall a.out
```

This command allows you to kill processes without the time-consuming task of looking up the process ID number with the *ps*(1M) command.

Note: This command kills all instances of the named program running under your shell and if invoked with no arguments, kills all processes on the system that are killable by the user who invoked the command. For ordinary

users, these are simply the processes invoked and forked by that user, but if invoked by *root*, all processes on the system will be killed. For this reason, this command should be used carefully.

Changing the Date and Time

Use the *date*(1) command to set the date and time. For example, to set the date to April 1st, 1999, and the time to 09:00, log in as *root* and enter:

```
date 0401090099
```

Changing the date and time on a running system can have unexpected consequences. Users and administrators use system scheduling utilities (*at*(1), *cron*(1), and *batch*(1)) to perform commands at specified times. If you change the effective date or time on the system, these commands may not execute at the desired times. Similarly, if your users use the *make*(1) utility provided with the system, the commands specified in Makefiles can incorrectly compile or process your users' work. Always try to keep your system date and time accurate within reason. Random changes of the date and time can be extremely inconvenient and possibly destructive to users' work.

If *timed*(1M) is running on the system, and it is a slave system, the time is reset by *timed* and not the above command. For more information, see the *timed*(1M) reference page.

Creating a Message of the Day

The file */etc/motd* contains the ''message of the day.'' This message is displayed on a user's screen, either by */etc/profile* if the user runs Bourne shell, or by */etc/cshrc* if the user runs C shell, when the user first logs in to the system.

You can place announcements of system activity in the *motd* file. For example, you should warn users of scheduled maintenance, changes in billing rates, new hardware and software, and any changes in the system or site configuration that affect them.

Since users see this message every time they log in, you should change it frequently to keep it from becoming stale. If users see the same message repeatedly, they lose interest in reading the message of the day and can miss an important announcement.

Make sure you remove outdated announcements. If nothing new is happening on the system, trim the file to a short "welcome to the system" message.

A typical *motd* file looks something like this:

```
Upcoming Events: ---------------

26 November -- The system will be down from 8PM until
Midnight for a software upgrade. We are installing
FareSaver+, Release 3.2.2d.

Watch this space for further details.

28 November through 31 November -- We will be operating with
a minimal staff during the holiday. Please be patient if you
need computer services. Use the admin beeper (555-3465) if
there is a serious problem.
```

The *motd* file is used more frequently on servers than on workstations, but can be handy for networked workstations with guest accounts. You can also send electronic mail to all people who use the system, but this method consumes more disk space, and users may accidentally skip over the mail in their mailboxes.

Creating a Remote Login Message

The file */etc/issue* is displayed before the login prompt is given to someone attempting to log in to a system over a serial line or the network. If */etc/issue* does not exist, or is empty, no message is displayed. This is essentially different from */etc/motd* because it is displayed *before* the login prompt. This message is often used to notify potential users of rules about using the equipment; for example, a disclaimer that the workstation or server is reserved for employees of a particular corporation and that intruders will be prosecuted for unauthorized use.

Maintaining the File Alteration Monitor

The File Alteration Monitor (*famd*) is a daemon that monitors files and directories. Application writers can include certain function calls in their applications to let *famd* know that they want to be informed of changes to whichever files and/or directories they specify. WorkSpace and Mailbox are two applications that use *famd*; WorkSpace uses it to keep the directory views up to date, and Mailbox uses it to know when to indicate the arrival of new mail.

The *famd* daemon runs only when applications using it are running; it exits after all programs using it have stopped.

Sometimes, when attempting to start up an application which uses *famd*, an error message is displayed:

```
Cannot connect with File Alteration Monitor (fam)
```

There are several reasons why this message appears. Below are some of the common ways to troubleshoot the problem.

Using a Foreign NIS Master

If you have the optional NIS (YP) software installed at your site, and you are using another manufacturer's system as your NIS master, with no *rpc* entries for *sgi_toolkitbus* and *sgi_fam*, this section provides the information to correct the error message.

Depending on the operating system (the Sun 3.*x* or the Sun 4.0) on the Sun NIS (YP) server, one of the two following solutions applies.

- Sun 3.*x*

  If the Sun Workstation is running version 3.*x* of Sun/OS, then two entries need to be added to the */etc/rpc* database on the Sun NIS server machine. They are `sgi_toolkitbus 391001` and `sgi_fam 391002`

  On the NIS server, enter the command:

  ```
  cd /usr/etc/yp make rpc
  ```

  It may take as much as an hour before the NIS server pushes this information to its clients.

- Sun 4.0

  If the Sun Workstation is running version 4.0 of Sun/OS or later, then two entries need to be added to the */etc/rpc* database on the Sun NIS server machine. They are `sgi_toolkitbus 391001` and `sgi_fam 391002`.

  On the NIS server, type:

  ```
  cd /var/yp make rpc
  ```

  It may take as much as an hour before the NIS server pushes this information to its clients.

  Note: If the NIS server machine is neither an SGI or Sun, the same *rpc* entries must be added, but the syntax may be different.

Verify fam Installation

From a shell window, enter the command:

```
ls -l /usr/etc/fam
```

If the file is not found, you must reinstall the software package:

eoe2.sw.envm.

Check Network Activity

The *famd* daemon is enabled via the */usr/etc/inetd.conf* file, which is read when the system starts up the network. Even if the system is not connected via the network to other systems, the network software must be started.

If the message:

```
portmapper failure
```

is displayed, it is also a sign that the network is not active.

If the command:

```
/etc/chkconfig | grep network
```

returns the message:

```
network off
```

Turn the network on by entering the command (as **root**):

```
/etc/chkconfig network on
```

and then either reboot your system by entering the command:

```
reboot
```

or just start up the network by issuing the command:

```
/etc/init.d/network start
```

"localhost" Entry in /etc/hosts

Applications connect to *famd* via the *localhost*. Therefore, there must be a *localhost* entry in the */etc/hosts* file.

This entry is in the *hosts* file by default, but sometimes gets removed. The correct entry would be:

```
127.0.0.1 localhost
```

sgi_famd/sgi_toolkitbus

Make sure that both *sgi_famd* and *sgi_toolkitbus* are running by entering the command:

```
/usr/etc/rpcinfo -p
```

In addition to other daemons running, you should see the following entries:

```
391002 1 tcp 1086 sgi_fam
391001 1 tcp 1087 sgi_toolkitbus
```

If you do not see these lines, check your */usr/etc/inetd.conf* file. The following entries must be in that file:

```
sgi_fam/1 stream rpc/tcp wait root /usr/etc/fam famd
sgi_toolkitbus/1 stream rpc/tcp wait root /usr/etc/
rpc.toolkitbus toolkitbusd
```

If those lines are not in the file, you must add them, and then restart your network by entering the following command, as **root**:

```
/etc/init.d/network stop
/etc/init.d/network start
```

vadmin Permissions

The permissions on the */usr/lib/vadmin* file should be:

```
drwxr-xr-x 2 root sys
```

eoe2.sw.fonts

If none of the listed problems exists, reinstall *eoe2.sw.fonts*. This will install a file called */usr/sysgen/boot/imon.o*. Confirm that the file is there, and then use the *autoconfig -vf* command to build a new kernel. Then reboot your system to install the new kernel.

Operating Policies

This section describes general operating policies for you, the site administrator. For a discussion of site policies, especially as they affect your users, see "System and Site Policies" on page 76 in this guide.

Shutting Down the System Carefully

Many administrative tasks require the system to be shut down to a run level other than the multiuser state. This means that conventional users cannot access the system. Just before the system is taken out of the multiuser state, users on the system 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 situations arise that require the system to be taken down with little or no notice provided to the users. This is often unavoidable, but try to give at least five to fifteen minutes of notice, if possible.

In general, try to provide the user community as much notice as possible about events affecting the use of the system. When the system must be taken out of service, also tell the users when to expect the system to be available. Use the "message of the day" file */etc/motd* to keep users informed about changes in hardware, software, policies, and procedures.

At your discretion, the following actions should be prerequisites for any task that requires the system to leave the multiuser state:

- When possible, perform service-affecting tasks during periods of low system use. For scheduled actions, use */etc/motd* to inform users of future actions.
- Check to see who is logged in before taking any actions that would affect a logged-in user. You can use the */etc/whodo*, */bin/who* and */usr/bsd/w* commands to see who is on the system. You may also wish to check for large background tasks, such as background compilations, by executing *ps -ef*.
- 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 (five to fifteen minutes) to terminate their activities and log off before taking the system down.

Maintaining a System Log Book

It is important to keep a complete set of records about each system you administer. A system log book is a useful tool when troubleshooting transient problems or when trying to establish system operating characteristics over a period of time. Keeping a hardcopy book is important, since you won't be able to refer to an online log if you have trouble starting the system.

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), including serial numbers of various parts (if applicable)
- copies of important configuration files
- the output of *prtvtoc*(1M) for each disk on the system
- the */etc/passwd* file

- the */etc/group* file
- the */etc/fstab* file
- the */etc/exports* file

The format of the system log and the types of items noted in the log should follow a logical structure. Think of the log as a diary that you update periodically. To a large measure, 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 a system administrator uses frequently. For additional information on the formats of the system files, refer to the IRIX reference pages.

Appendix B, "IRIX Device Files,"contains further information on the device files and directories that reside in the */dev* directory.

Root Directories

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

/	Contains hardware-specific files and files required to start the system.
bin	Contains publicly executable commands. (Some are *root*-only.)
debug	Provides a link to */proc*.
dev	Contains special files that define all of the devices on the system.
etc	Contains administrative programs and tables.
lib	Contains public libraries.
lost+found	Used by *fsck*(1M) to save disconnected files and directories.
proc	Provides an interface to running processes that may be used by debuggers such as *dbx*(1).

tmp	Used for temporary files.
usr	Used to mount the */usr* file system and for files that are the same from system to system. These files are not writable.
var	Used for files that are specific to each system. There is typically a symbolic link to */usr* for each file in */var*.

Important System Directories

The following directories are important in the administration of your system:

/etc/init.d	Contains shell scripts used in upward and downward transitions to all system run levels. These files are linked to files beginning with *S* (start) or *K* (kill) in */etc/rcn.d*, where *n* is replaced by the appropriate run level number.
/etc/config	Contains start-up and run-time configuration information.
/etc/rc0.d	Contains files executed by */etc/rc0* to bring the system to run-level 0. 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 killed, and *S* indicates processes that are started when entering run-level 0.
/etc/rc2.d	Contains files executed by */etc/rc2* for transitions to system run-level 2. 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 killed, and *S* indicates processes that should be started, when entering run-level 2.
/etc/rc3.d	Contains files executed by */etc/rc3* for transitions to system run-level 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, and *S* indicates processes that should be started when entering run-level 3.
/var/adm/acct	Contains information collected by the accounting subsystem.

/var/adm/crash
: Contains crash dumps of the system. After analysis, and if appropriate, these dumps can safely be removed unless your support provider has requested otherwise. See the *savecore*(1) reference page for more information.

/var/adm/sa
: Contains information collected by *sar*(1).

/usr/people
: Contains the home directories of users of the system or network. This directory can be a link to */var/people* or a mount point for a totally separate file system.

/usr/share
: This directory contains files that are the same on all systems.

/var/spool
: Contains spooling directories. The directories in this directory hold outbound mail, print requests, and other data.

/var/spool/cron/crontabs
: Contains *crontab* files for the *adm*, *root*, and *sys* logins and ordinary users listed in *cron.allow*.

/var/sysgen/master.d
: Contains files that define the configuration of hardware devices, software services and utilities, and aliases.

/var/sysgen/stune
: Contains files that define the default settings of all kernel tunable parameters.

/var/sysgen/mtune
: Contains files that define the current settings of all kernel tunable parameters.

Important System Files

The following files are important in the administration of your system:

/etc/cshrc
: Contains the standard (default) environment for */bin/csh* users.

/etc/exports
: Contains the list of NFS file systems exported at boot time to NFS clients if the optional NFS software is installed.

/etc/fstab	Specifies the file system(s) to be mounted by */etc/mountall* if the optional NFS software is installed.
/etc/gettydefs	Contains information used by *getty* to set the speed and terminal settings for a line.
/etc/group	Describes each group to the system.
/etc/hosts	Contains information about the known hosts on the network.
/etc/hosts.equiv	Contains a list of hosts trusted for non-superuser *rlogin* and *rsh* execution.
/etc/inittab	Contains the instructions to define the processes created or terminated by *init* for each initialization state.
/etc/issue	Displays a message to users before logging in to the system over the network or on serial lines.
/etc/lvtab	Contains information describing the logical volumes used by the workstation. This file is read by the logical volumes utilities.
/etc/motd	Contains a brief ''message of the day.''
/etc/passwd	Identifies each user to the system.
/etc/profile	Contains the standard (default) environment for */bin/sh* users.
/etc/rc0	Contains a script that executes shell scripts in */etc/rc0.d* to bring the system to run-level 0.
/etc/rc2	Contains a script that executes shell scripts in */etc/rc2.d* and */etc/rc.d* on transitions to system run-level 2.
/etc/shutdown	Contains a shell script that gracefully shuts down the system in preparation for system backup or for scheduled downtime.
/etc/sys_id	Contains the system name.
/etc/ttytype	Contains a list, ordered by terminal port, of what kind of terminal is likely to log in to that port.

/etc/TIMEZONE
: Used to set the default time zone shell variable *TZ*.

/etc/utmp
: Contains the information on the current runstate of the system.

/etc/wtmp
: Contains a history of system logins.

/etc/xwtmp
: Contains an extended history of system logins.

/var/adm/sulog
: Contains a history of *su* command usage. This file should be checked periodically for excessive size and archived.

/var/adm/SYSLOG
: Contains system and daemon error messages.

/var/yp/ypdomain
: Contains the domain name if the workstation is using NIS.

/var/cron/log
: Contains a history of all the actions taken by *cron*. This file should be checked periodically for excessive size and reduced if necessary.

/usr/lib/cron/cron.allow
: Contains a list of users allowed to use *crontab*(1). This file cannot exist on the system at the same time as *cron.deny*.

/usr/lib/cron/cron.deny
: Contains a list of users who are denied access to *crontab*(1). It is checked if */usr/lib/cron/cron.allow* does not exist.

Operating Levels

The IRIX system can run in either single-user or multiuser mode. In single-user mode, only a few processes are active on the system, no graphics are available, and only a single login is allowed. In multiuser mode, there can be multiple login sessions, many files open at once, and many active processes, including numerous background daemons.

The *init* program controls whether the system is in the multiuser or single-user state. Each possible state that the system can be in is assigned a label,

either a number or a letter. The shutdown state is state *0*. Single-user mode is state *s*.

Multiuse state labeling is more complex, because there can be great variations in multiuser states. For example, in one multiuser state, there can be unlimited logins, but in another state there can be a restricted number of logins. The different states can all be assigned different numbers.

The state of the system is controlled by the file */etc/inittab*. This file lists the possible states, and the label associated with each.

When you bring the system to standard multiuser mode, *init* state 2, the following happen:

- The file systems are mounted.
- The *cron* daemon is started for scheduled tasks.
- Network services are started, if turned on.
- The serial-line networking functions of *uucp* are available for use.
- The spooling and scheduling functions of the *lp* package (if added to the system) are available for use.
- Users can log in.

Not all activities can or should be performed in the multiuser state. Some tasks, such as installing software that requires the miniroot and checking file systems must be done with the system in single-user mode.

There are many synonyms for the system state. These include:

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

Likewise, each system state may be referred to in a number of ways; for example:

- single user

- single-user mode
- run level 1

Table 2-4 shows the various possible states of the operating system as it is shipped. You can, of course, create your own custom states.

Table 2-4 System States

Run Level	Description
0	Power-down state.
1, s, or S	Single-user mode is used to install/remove software utilities, run file system backups/restores, and check file systems. This state unmounts everything except root, and kills all user processes except those that relate to the console.
2	Multiuser 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 multiuser mode.
6	Reboot mode is used to bring down the system and then bring it back up again. This mode is useful when you are changing certain system configuration parameters.

How init Controls the System State

The *init* process is the first general process created by the system at startup. It reads the file */etc/inittab*, which defines exactly which processes exist for each run level.

In the multiuser state (run level 2), *init* scans the file for entries that have a tag (also 2) for the run level and executes everything after the third colon on the line containing the tag. For complete information, see the *inittab*(4) reference page.

The system */etc/inittab* looks something like this:

```
is:2:initdefault:
fs::sysinit:/etc/bcheckrc </dev/console >/dev/console 2>&1
mt::sysinit:/etc/brc </dev/console >/dev/console 2>&1
s0:06s: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
or:06:wait:/etc/umount -ak -b /proc,/debug > /dev/console
2>&1
of:0:wait:/etc/uadmin 2 0 >/dev/console 2>&1 </dev/console
RB:6:wait:echo "The system is being restarted." >/dev/console
rb:6:wait:/etc/uadmin 2 1 >/dev/console 2>&1 </dev/console
#
```

This display has been edited for brevity; the actual */etc/inittab* file is lengthy. If */etc/inittab* is removed by mistake and is missing during shutdown, *init* enters the single-user state (*init s*). While entering single-user state, */usr* remains mounted, and processes not spawned by *init* continue to run. Immediately replace */etc/inittab* before changing states again. The format of each line in *inittab* is:

id:*level*:*action*:*process*

where:

- *id* is one or two characters that uniquely identify an entry.
- *level* is zero or more numbers and letters (0 through 6, s, a, b, and c) that determine in what *level*(s) *action* is to take place. If *level* is null, the *action* is valid in all levels.
- *action* can be 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 multiuser state after the system is started. (If *initdefault* is set to 2, the process runs right after the startup.) *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 has finished.

  - initdefault

    When *init* starts, it enters *level*; the *process* field for this *action* has no meaning.

  - once

    Run *process* once. If it finishes, don't start it again.

  - powerfail

    Run *process* whenever a direct powerdown of the computer is requested.

  - 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.
- # can be used to add a comment to the end of a line. *init* ignores all lines beginning with the # character.

When changing levels, *init* kills all processes not specified for that level.

Entering the Multiuser State from System Shutdown

When your system is up and running, it is usually in multiuser mode. It is only in this mode that the full power of IRIX is available to your users.

Powering Up the System

When you power up your system, it enters multiuser mode by default. (You can change the default by modifying the *initdefault* line in your *inittab* file.) In effect, going to the multiuser state follows these stages (see Table 2-4, "System States," on page 64):

1. The operating system loads and the early system initializations are started by *init*.
2. The run level change is prepared by the */etc/rc2* procedure.
3. The system is made public through the spawning of *getty* processes along the enabled terminal lines.

Early Initialization

Just after the operating system is first loaded into physical memory through the specialized boot programs that are resident in the PROM hardware, 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
mt::sysinit:/etc/brc </dev/console >/dev/console 2>&1
```

These entries 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 establishes communication with the system console before the system is brought to the multiuser state.

Preparing the Run Level Change

Now the system is 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 to select the next entries to be executed. In our sample */etc/inittab*, the *initdefault* entry is run level 2 (the multiuser state):

```
is:2:initdefault:
s2:23:wait:/etc/rc2 >/dev/console 2>&1 </dev/console
co:23:respawn:/etc/gl/conslog
t1:23:respawn:/etc/getty -s console ttyd1 co_9600 #altconsole
t2:23:off:/etc/getty ttyd2 co_9600 # port 2
t3:23:off:/etc/getty ttyd3 co_9600 # port 3
t4:23:off:/etc/getty ttyd4 co_9600 # port 4
```

The other entries shown above specify the actions necessary to prepare the system to change to the multiuser 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:

- Sets up and mounts the file systems
- Starts the *cron* daemon
- Makes *uucp* available for use
- Makes the line printer (*lp*) system available for use, if installed
- Starts accounting, if installed and configured on
- Starts networking, if installed and configured on
- Starts *sar*, if installed and configured on
- Starts the mail daemon
- Starts the system monitor

init then starts a *getty* for the console and starts *getty* on the lines connected to the ports indicated.

At this point, the full multiuser environment is established, and your system is available for users to log in.

Changing Run Levels

To change run levels, the system administrator enters a command that directs *init* to execute entries in */etc/inittab* for a new run level, such as *multi*, *single*, or *reboot*. Then key procedures, such as *shutdown*, */etc/rc0*, and */etc/rc2*, are run to initialize the new state.

The new state is reached. If it is state 1 (single-user mode), the system administrator can continue.

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 file names look like this:

S00name

or

K00name

The filenames can be split into three parts:

S or K	The first letter defines whether the process should be started (S) or killed (K) upon 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, etc).
name	The rest of the filename is the */etc/init.d* filename to which this file is linked.

For example, the *init.d* file *cron* is linked to the *rc2.d* file and *rc0.d* file *S75cron*. 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 */sbin/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 these files, though it is preferable to add your own since the delivered scripts may change in future releases. To create your own scripts, follow these rules:

- Place the file in */etc/init.d*.
- Symbolically 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.

Note that it may prove easier to simply copy an existing script from the directory and make appropriate changes.

Going to Single-user Mode From Multiuser Mode

Sometimes you must perform administrative functions, such as backing up the *root* file system, in single-user mode.

Use the *shutdown* command. This procedure executes all the files in */etc/rc0.d* by calling the */etc/rc0* procedure. The *shutdown* command does the following things, among others:

- Closes all open files and stops all user processes
- Stops all daemons and services
- Writes all system buffers out to the disk
- Unmounts 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
```

There are three recommended ways to start the shutdown to single-user mode:

1. You can enter the *shutdown -i1* command (recommended). The option **-g** specifies a grace period between the first warning message and the final message.
2. You can enter the *single* command, which runs a shell script that switches to single-user mode and turns the *getty* processes off.
3. You can enter the *init 1* command, which forces the *init* process to scan the table. The first entry it finds is the **s1** entry, and *init* starts the shutdown processing according to that entry.

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

/etc/inittab and Power Off

The following entries in */etc/inittab* power off the system:

```
s0:06s:wait:/etc/rc0 >/dev/console 2>&1 </dev/console
of:0:wait:/etc/uadmin 2 0 >/dev/console 2>&1 </dev/console
```

Always attempt to shut the system down gracefully. You can either enter the *powerdown* command, the *init 0* 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 */sbin/uadmin* command is called, which indicates

that the last step (physically removing power from the system) is under firmware control.

Encapsulated PostScript File v.3.0 vs. PostScript File Format

In the course of maintaining your system, you are likely to receive files in various versions of the PostScript format. Following are some of the main differences between the Encapsulated PostScript® File (EPSF) version 3.0 format and PostScript file format:

- EPSF is used to describe the appearance of a single page, while the PostScript format is used to describe the appearance of one or more pages.
- EPSF requires the following two DSC (document structuring conventions) Header comments:

  ```
  %!PS-Adobe-3.0 EPSF-3.0 %%BoundingBox: llx lly urx ury
  ```

  If a PS 3.0 file conforms to the document structuring conventions, it should start with the comment:

  ```
  %!PS-Adobe-3.0
  ```

  A PS file does not have to contain any DSC comment statements if it does not conform to the DS conventions.
- Some PostScript language operators, such as *copypage*, *erasepage*, or *exitserver* must not be used in an EPS file.

  Certain rules must be followed when some other PostScript operators, such as *nulldevice*, *setscreen*, or *undefinefont* are used in an EPS file.

  All PostScript operators can be used in a PS file.
- An EPS file can be (and usually is) inserted into a PS file, but a PS file must not be inserted into an EPS file if that will violate the rules for creating valid EPS files.

- An EPS file may contain a device-specific screen preview for the image it describes. A PS file usually contains screen previews only when EPS files are included in that PS file.
- The recommended file name extension for EPS files is .**EPS** or .**eps**, while the recommended file name extension for PS files is .**PS** or .**ps**.

The EPSF format was designed for importing the PostScript description of a single page or part of a page, such as a figure, into a document, without affecting the rest of the description of that document. EPS code should be encapsulated, i.e., it should not interfere with the PS code that may surround it, and it should not depend on that code.

The EPSF format is usually used for the output from a program that was designed for the preparation of illustrations or figures, (such as the Adobe Illustrator) and as input into desktop publishing programs (such as the Frame Technology's FrameMaker). Most desktop publishing programs can produce the description of a document in the PostScript format that may include the imported EPS code.

For more information about these formats, see the book *PostScript Language Reference Manual*, Second Edition, Addison-Wesley Publishing Company, Inc., Reading, MA, 1990. You can get several documents on the EPS and PS formats from the Adobe Systems PostScript file server by entering the following at a UNIX prompt, and then following the instructions you get from the file server:

mail ps-file-server@adobe.com
Subject: help
Ctrl-D

You can get a description of the EPSF format in the PS format by sending the following message to that file server:

```
send Documents EPSF2.0.ps
```

Chapter 3

User Services

Chapter 3 provides information on the services an administrator provides to the user community. This chapter covers the following general topics:

- *Adding and deleting users*
- *Adding and deleting user groups*
- *Establishing system policies that affect users.*
- *Helping users configure the system for their needs*
- *Communicating with users.*

Chapter 3

User Services

This chapter describes how to provide a variety of services to the users of your IRIX system. These services include:

- Establishing and maintaining system and site policies. See "System and Site Policies" on page 76.
- Administering user accounts: creating and removing accounts, creating and removing groups of users, and maintaining the */etc/passwd* and */etc/group* files. See "Login Administration" on page 82.
- Helping users configure their login environments. See "The User's Environment" on page 97.
- Establishing and maintaining communications tools such as the "message of the day" and news. See "Communicating with Users" on page 107. Setting up electronic mail and setting up networks are described in Chapter 20, "IRIX sendmail," and Chapter 17, "Setting Up a Network."
- Developing an organized plan for responding to user problems. See "Anticipating User Requests" on page 111.
- Creating reference pages to help users understand and use your custom developed applications and utilities. See "Creating Reference Pages" on page 112.

System and Site Policies

Before you can administer a system or site efficiently, you must make sure that there are consistent policies regarding system resources and that they are enforced. Everyone who uses the computers at the site should be aware of these policies.

Site policies should cover such topics as:

- disk use
- accounts and passwords
- root access
- privacy
- malicious activities

You may need other policies, depending upon the specific requirements of your site. For a more complete discussion of system security, see Chapter 12, "System Security."

Also see Chapter 2, "Operating the System," for a discussion of what policies you, as site administrator, should follow when you administer systems.

Disk Use and Quotas

For a variety of reasons, users often accumulate files without realizing how much disk space they are using. This is not so much a problem for workstation users, but it is a real problem for servers, where multiple users must share system resources. IRIX provides a number of utilities to help you manage disk usage.

This section provides you with some background information and tips on the tools and options available to you. For specifics on setting and maintaining disk quotas, see "Imposing Disk Quotas" on page 296.

The du(1) Command

The *du*(1) command shows specific disk usage by file or directory anywhere on the system. The size of the files and directories can be shown in 512-byte blocks, or in 1K blocks if you use the **-k** flag. For more complete information, consult the *du*(1) reference page.

The df(1) Command

The *df*(1) command shows the free space remaining on each file system on your workstation. Free space is shown for all file systems, whether they are local or NFS-mounted. The amount of free space is displayed in 512-byte blocks, or in 1K blocks if you use the **-k** option. For more complete information, consult the *df*(1) reference page.

The quot(1M) Command

The *quot*(1M) command displays the number of kilobytes of disk usage for each user in a specified file system. You can use the output of this command to mail your users, notifying them of their disk usage. For complete information, see the *quot*(1M) reference page.

The diskusg(1) Command

Part of the system accounting package is the *diskusg*(1) command. Like *quot*, this utility reports the disk usage of each user on your system. The difference is that *diskusg* is typically used as part of the system accounting package with *dodisk* rather than as a standalone command. For complete information on using *diskusg*(1), see the *diskusg* reference page.

File Compression and Archiving

One way to minimize disk usage is to encourage your users to archive or compress their files. Compression works best for files that are rarely used but that should not be discarded. Users can achieve some disk savings by using the *compress*(1) utility. Compressing a file allows you to leave it on the system, but can reduce the size of the file by as much as 50%. Compressed files have the suffix **.Z** and should not be renamed. Compression and archiving can be used together for maximum disk space savings.

If the files in question are used infrequently, consider archiving them to tape, floppy, or some other archive media. This maintains protection if you need the files later, but regaining them is slightly more difficult than if they were compressed.

The *quotas*(4) Subsystem

The *quotas*(4) subsystem exists to allow you to deal with severe disk usage problems. Using this subsystem, you can place a maximum disk usage quota on each user on your system. For complete information about this subsystem, see the *quotas*(4) reference page.

In general, it is your job as the site administrator to set disk use policies, establishing and enforcing quotas if necessary. You should publish clear guidelines for disk use, and notify users who regularly exceed their quotas. It is also a good idea to impose quotas on the use of temporary directories, such as */tmp* and on all anonymous "guest" accounts, such as *guest* and *uucp*. If your root file system reaches 100% capacity, your system may shut down and inconvenience your users.

You should be as flexible as possible with disk quotas. Often, legitimate work forces users to temporarily exceed disk quotas. This is not a problem as long as it is not chronic.

Do not, under any circumstances, remove user files arbitrarily and without proper warning.

A typical scenario is when all the users on the system know they should try to limit disk use in their home accounts to about 20 MB (about 40,000 512-byte blocks). User *norton* consistently uses more than twice this limit. These are the steps you take to alleviate the problem:

1. Try to meet with the user and find out why he or she is using this much disk space. There may be legitimate reasons that require you to reconsider the disk use policy and perhaps increase the amount of available disk space.
2. If the user is merely saving an inordinate number of outdated files, suggest that he or she back up the files onto tape and remove them from the system. For example, many users save electronic mail messages in enormous mailboxes for long after the messages are useful. Saving the files to tape keeps them handy, while saving disk space.

3. If you cannot meet with the person, or cannot discuss the matter with them, try sending them electronic mail.

   If you use a script that automatically checks disk use (with *diskusg*) and sends mail to users who exceed their quotas, note that people get used to these messages after some time and start to ignore them. Send the particular user a personal message, stating that you need to discuss the situation with him or her.

4. Sometimes a user is not aware that data is available elsewhere on the system, or on other accessible workstations at the site. A user may have personal copies of site-specific tools and data files. Work with the user to reduce this kind of redundancy.

5. Make sure the user is still active on the system. Sometimes people leave an organization, and the site administrators are not immediately informed.

   Also, the user may not need the account on a particular workstation any longer and may not have cleaned up the files in that account. To see if this is the case, check the last time the user logged in to the system with the *finger*(1) command:

   ```
   finger norton
   ```

   Among other information, *finger* displays the date and time the user last logged in to the system. This information is read from */etc/wtmp*, if it exists.

6. If in an extreme case you must remove a user's files, back them up to tape before removing them. Do not take this step lightly. Removing user files unnecessarily can disrupt work and engender ill-will from your coworkers. Make sure you give the user plenty of advance notice that you are going to copy the files to tape and remove them from the system.

   As an added precaution, you may wish to make two copies of the tape and send one copy to the user whose files you remove. Make sure you verify that the tapes are readable before you remove the files from the system.

Managing Disk Space with NFS

If your system is running the optional Network File System (NFS) software, you can use this product to reduce disk consumption on your workstations

by exporting commonly used directories. When a system exports a directory, it makes that directory available to all systems running the NFS software. For example, if you have 10 workstations on your network and each workstation is storing 5 MB of online reference pages and release notes, you can eliminate 45 MB of wasted disk space by designating one workstation to be the reference page server and exporting the */usr/man* and */usr/catman* directories. All other workstations can remove those files and mount them remotely from the server. Since NFS mounts take up no disk space on the client workstation, that disk space is available for other uses.

Another option is to mount free disk space from another system. This option works best when there is an uneven load across several systems. For example, if one workstation is using only 25% of its available space and other workstations are above 90%, it may be useful to mount a file system to even out the load.

Used wisely, your network can save a tremendous amount of disk space through this strategy. Be aware, though, that the drawback to this strategy is that if your server must be rebooted or serviced, no one will be able to access the files and programs that are mounted from that server while it is off-line.

Managing Disk Space with Disk Partitions

An extreme method of enforcing disk use quotas is to create a disk partition and file system for each user account and to mount the file system on an empty directory as the user's home directory. Then, when users run out of disk space, they will not be able to create or enlarge any files. They will, however, still be able to write their files to */tmp*, */usr/tmp.O*, and */var/tmp*, unless those directories are also full. When users attempt to write or create a file that takes them over their limit, they receive an error message indicating that no disk space is left on the device.

This method of disk control is not generally recommended. It requires a great deal of preparation and maintenance on the part of the Administrator, and is not easily modified once in place. Additionally, fragmenting your disk into many small partitions reduces the total amount of available disk space and produces a performance overhead on your system. In this case, the disk controller must write a user's files within only one small partition, instead of in the next convenient place on the disk.

You should consider this method of disk space control only if your system is so overloaded and your users so obstinate that all other measures have failed. If your */usr* partition is chronically 100% full, the system is halting operations daily due to lack of disk space, and there is no way to increase your disk space, then you may want to consider this method.

Accounts and Passwords

To make your site as secure as possible, each user should have an account, with a unique user ID number, and each account should have a password. Users should never give out their passwords to anyone else under any circumstances. For more information on passwords and system security, see "Logins and Passwords" on page 431.

Root Access

For best security on a multiuser server, access to the *root* account should be restricted. On workstations, whoever is the primary user of the workstation can certainly use the *root* account, though people should not have access to the *root* account on each other's workstations.

Make it a policy to give root passwords to as few people as is practical. Some sites maintain locked file cabinets of root passwords so that the passwords are not widely distributed, but are available in an emergency.

Privacy

On a multiuser system, users may have access to personal files that belong to others. Such access can be controlled by setting file permissions with the *chmod*(1) command. Default permissions are controlled by the *umask* shell parameter. (See "umask" on page 105 for information on setting *umask*.)

However, to make it easier to exchange data, many users do not set their *umasks*, and they forget to change the access permissions of personal files. Make sure users are aware of file permissions and of your policy on examining other users' personal files.

You can make this policy as lenient or stringent as you deem necessary.

Malicious Activities

You should set a policy regarding malicious activities. These include:

- deliberately crashing the system
- breaking into other accounts; for example, using password-guessing and password-stealing programs
- forging electronic mail from other users
- creating and unleashing malicious programs, such as worm and virus processes

Make sure that all users at the site are aware that these sorts of activities are potentially very harmful to the community of users on the system. Penalties for malicious behavior should be severe and the enforcement should be consistent.

The most important thing you can do to prevent malicious damage to the system is to restrict access to the root password.

Login Administration

Creating and deleting user accounts are two of the most common system administration tasks you will perform. You should read and understand this section before you undertake to manipulate user accounts. The graphical System Manager tool (available on graphics workstations only) is the most convenient tool for these tasks. The System Manager is described in the *Personal System Administration Guide*.

User IDs

Each user account has a user ID number. These numbers are unique on each workstation and should be unique throughout your entire site. A user ID for an account is kept in the third field of the */etc/passwd* file.

After you close a user account, do not reuse that account's user ID number. It is possible that files somewhere on a system could still be owned by the ID number, or files may be placed on the system from an old backup tape. These

sorts of file would be compromised by associating a new user with an old ID number. In general, the rule is that the ID number is the permanent property of the user to whom it is assigned. For more information on user ID numbers, see the *passwd*(4) reference page.

Group IDs

Each user account belongs to a group of users on the system. Users with similar interests or jobs can belong to the same group. For example, members of the publications department might belong to group *pub*. The benefit to this arrangement is that it allows groups of related users to share files and resources without sharing those files or resources with the entire system community.

Each group has a group ID number. These numbers are unique on each system and should be unique throughout the entire site. Like user IDs, you should not reuse group IDs.

When you create a file, it is assigned your group ID. You can change the group ID of a file with the *chgrp*(1) command. 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. A sample entry from this file is shown and explained below:

```
raccoons::101:norton,ralph
```

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

group name — The group name can be any length, though some commands will truncate the name to 8 characters. The first character must be alphabetic.

password — The password field may contain an encrypted password. An empty field, as in the above example, indicates that no password is required. The *passwd*(1M) command cannot be used to create or modify a group password. To place a password on a group, you must use the *passwd* command to encrypt a password. (Use a test user account created

	specifically for this purpose and then delete the test account.) Then, copy that encrypted password verbatim from the */etc/passwd* file into the */etc/group* entry you wish to protect with the password. Users specifically listed as group members in the */etc/group* file entry will not be required to give the password, but other users will be so required when they attempt to change groups to the protected group with the *newgrp*(1) command. Password protection, though, is rarely used on user groups.
group ID	The group ID is a number from 0 to 60,000. The number must not include a comma. Numbers below 100 are reserved for system accounts.
login names	The login names of group members are in a comma-separated list.

For complete information on user groups, see the *group*(4) reference page.

Adding Users Using Shell Commands

Occasionally, you may have to add a user account manually; in other words, without using the automated tools such as the System Manager. This method is the default for server administrators, but all administrators should understand the process in case a problem develops with some part of the automated tools or if you wish to design your own scripts and programs for administering user accounts at your site. Be sure to check your work with the *pwck*(1M) command.

Follow these steps to add a user manually:

1. Log in as **root**.
2. Edit the file */etc/passwd* with your preferred text editor.

   The file */etc/passwd* has one line for each account on the system. Each line contains seven fields, and each field is separated by a colon. The lines look similar to this:

   ```
   ralph:+:103:101:Ralph Cramden:/usr/people/ralph:/bin/csh
   ```

3. Copy one of the lines (for example, the last line in the file) and add it to the end of the file.

4. Change the first field (*ralph* in this example) to the name of the new account; for example, *alice*.
5. Remove any characters between the first colon after the account name and the second colon. Deleting these characters removes the password from an account. Either you or the new user can add a password later.
6. The next field (in this case "103") is the user ID of the new account. Change it to a number 1 greater than the current highest user ID on your system. You should not use user ID numbers between 0 and 100, as these are reserved for system use.
7. The next field (in this case "101") is the group ID number of the new account. Check the file */etc/group* and pick a suitable group ID for the new user account. The */etc/group* file lists all the groups on the system by group ID, followed by a list of the current users who belong to that group.
8. Change the next field (in this case *Ralph Cramden*) to the name of the new user, in this case *Alice Cramden*. If you wish, you can add an "office" and "phone number" to this field. After the user's name, add a comma, then the office location, another comma, and the phone number. For example:

   ```
   :Alice Cramden, Brooklyn, (212) 555-1212:
   ```

   Actually, you can put any information you wish in these fields. The fields are interpreted by the *finger*(1) program as "user name, office, phone number."
9. The next field (in this case */usr/people/ralph*) is the location of the user's home directory. Change this field to reflect the name of the new user's account. In this example, you would change */usr/people/ralph* to */usr/people/alice*.
10. The last field (in this example */bin/csh*) is the user's login shell. For most users, the C shell (*/bin/csh*), Korn Shell (*/bin/ksh*), or Bourne shell (*/bin/sh*) is appropriate. You should leave this field unchanged, unless you wish to use a different or special shell. Special shells are discussed in "Special Login Shells" on page 106. Once you have selected a shell, you are finished editing */etc/passwd*.
11. Write the changes you made and exit the file.

The next step, which is optional, is to add the new user to the file */etc/group*. A user can be a member of a group without being listed in the */etc/group* file.

12. If you want to maintain a list of the groups to which users belong, edit the file */etc/group*. You should see some lines similar to this:

```
sys::0:root,bin,sys,adm
root::0:root
daemon::1:root,daemon
bin::2:root,bin,daemon
adm::3:root,adm,daemon
mail::4:root
uucp::5:uucp
rje::8:rje,shqer
lp:*:9:
nuucp::10:nuucp
bowling:*:101:ralph
other:*:102:
```

Place the name of the new account (in this example *alice*) after any of the groups. Separate the account name from any other account names with a comma, but not with blank spaces. For example:

```
bowling:*:101:ralph,alice
```

Adding account names to the */etc/group* file is optional, but it is a good way to keep track of who belongs to the various system groups.

Also, you can assign an account to more than one group by placing the account name after the names of the various groups in */etc/group*. The user can change group affiliations with the *newgrp*(1) and *multgrps*(1) commands.

13. When you finish editing */etc/group*, write your changes and exit the file.

The next step is to create the new user's home directory and copy shell startup files over to that directory.

14. Use the *mkdir*(1) command to create the user's home directory. For example, to create a home directory for the user "alice":

```
mkdir /usr/people/alice
```

Make the directory owned by user *alice*, who is in group *bowling*:

```
chown alice /usr/people/alice
chgrp bowling /usr/people/alice
```

Make sure the new home directory has the appropriate access permissions for your site. For a site with relaxed security:

```
chmod 755 /usr/people/alice
```

For more information on the *chown*(1), *chgrp*(1), and *chmod*(1) commands, see the respective reference pages.

15. Copy the shell startup files to the new user's home directory.

    If the new account uses the C shell:

    ```
    cp /etc/stdcshrc /usr/people/alice/.cshrc
    cp /etc/stdlogin /usr/people/alice/.login
    ```

    If the new account uses the Korn or Bourne shell:

    ```
    cp /etc/stdprofile /usr/people/alice/.profile
    ```

16. You can make these shell startup files owned by the user, or leave them owned by *root*. Neither approach affects how the user logs in to the system, although if the files are owned by *root*, the user is less likely to alter them accidentally and be unable to log in.

    To give a user complete access to his or her shell startup files, use the *chmod* command. For C shell:

    ```
    chmod 755 /usr/people/alice/.cshrc /usr/people/alice/
    .login
    ```

    For Korn or Bourne shell:

    ```
    chmod 755 /usr/people/alice/.profile
    ```

    Remember to check for any other user files that may be owned by *root* in the user's directory, and change those, too.

17. Issue the *pwck*(1M) command to check your work. This command performs a simple check of the */etc/passwd* file and makes sure that no user ID numbers have been reused and that all fields have reasonable entries. If your work has been done correctly, you should see output similar to the following:

    ```
    sysadm:*:0:0:System V Administration:/usr/admin:/bin/sh
            Login directory not found
    auditor::11:0:Audit Activity Owner:/auditor:/bin/sh
            Login directory not found
    dbadmin::12:0:Security Database Owner:/dbadmin:/bin/sh
            Login directory not found
    tour::995:997:IRIS Space Tour:/usr/people/tour:/bin/csh
    ```

```
        Login directory not found
4Dgifts::999:998:4Dgifts Acct:/usr/people/4Dgifts:/bin/csh
        First char in logname not lower case alpha
        1 Bad character(s) in logname
        Login directory not found
nobody:*:-2:-2::/dev/null:/dev/null
        Invalid UID
        Invalid GID
```

These messages are normal and expected from *pwck*. All errors generated by *pwck* are described in detail in the *pwck*(1M) reference page.

Adding a Group Using Shell Commands

Follow these steps to add a group to the system manually:

1. Log in as **root**.
2. Edit the file */etc/group*. The file contains a list of groups on the system, one group per line. Each line contains the name of the group, an optional password, the group ID number, and the user accounts who belong to that group.

   For example, to create a group called *raccoons*, with a group ID of 103, place this line at the end of the file:

   ```
   raccoons:*:103:
   ```

3. If there are users who should belong to the group, add their names in the last field. Each name should be separated by a comma, for example:

   ```
   raccoons:*:103:ralph,norton
   ```

4. Write and exit the file. Make sure the group IDs in the file */etc/passwd* file match those in the */etc/group* file.

For more information on user groups, see the *group*(4) reference page.

Changing a User's Group

To change a user's group affiliation, perform these steps:

1. Login as **root**.
2. Edit the file */etc/group*. Place the user's account name on the line corresponding to the desired group. If the account name appears as a member of another group, remove that reference unless you want the account to be a member of both groups.
3. Write and exit the file */etc/group*.
4. Edit the file */etc/passwd*.
5. Find the user's entry in the file.
6. Change the old group ID on that line to the new group ID. The group ID is the fourth field (after the account name, password, and user ID).
7. Write and exit the file.

The user's group affiliation is now changed. Remind the user to change the group ownership on his or her files. If you prefer, you can perform this task yourself as **root** using the *find*(1) and *chgrp*(1) commands. See "Using find to Locate Files" on page 25 for more information.

Deleting a User from the System

This procedure deletes the user's home directory and all the files in and below that directory. If you only wish to close or disable a user account but preserve the user's files and other information, see "Closing a User Account" on page 90.

To delete a user's account completely, follow these steps:

1. Log in as **root**.
2. If you think you might need a copy of the user's files later on, make a backup copy of the directory (for example, on cartridge tape using *tar*(1) or *cpio*(1)).

3. Edit the */etc/passwd* file and replace the encrypted password (or "+" sign if you are using shadow passwords) with the following string:

   ```
   *ACCOUNT CLOSED*
   ```

   It is imperative that the asterisks shown in this example be used as shown. An asterisk in the encrypted password field disallows all logins on that account. Alternately, you can lock an account by using fewer than thirteen characters in the password field, but it is better to use the asterisks and an identifiable lock message.

4. Use *find*(1) to locate all files on the system that are owned by the user and remove them or change their ownership. Information on the use of *find*(1) is provided in "Using find to Locate Files" on page 25 and in the *find*(1) reference page.

Deleting a Group from the System

Deleting a group from the system is done in the following steps:

1. Edit the */etc/group* file and remove the group entry.
2. Edit the */etc/passwd* file and remove the group from the user entries wherever it exists.
3. Use *find*(1) to find all files and directories with the group affiliation and change it to a surviving group with the *chgrp*(1) command.

Closing a User Account

If you wish, you can close a user's account so that nobody can log in to it or *su* to that user's ID number.

If you expect that the user will again require access to the system in the near future, close an account in the manner described below rather than removing it from the system completely (as described in "Deleting a User from the System" on page 89).

To close an account, follow these steps:

1. Log in as **root**.
2. Edit the file */etc/passwd*. Find the user's account entry.

3. Make the entry a comment by placing a number sign at the beginning of the line. For example:

   ```
   # ralph:+:103:101:Ralph Cramden:/usr/people/ralph:/bin/csh
   ```

4. As an added measure of security, you can replace the encrypted password (the second field in the entry) with a string that cannot be interpreted as a valid password. For example:

   ```
   # ralph:*:103:101:Ralph Cramden:/usr/people/ralph:/bin/csh
   ```

   This has the added benefit of reminding you that you deliberately closed the account.

5. If necessary, you can also close off the user's home directory with the following commands:

   ```
   chown root /usr/people/ralph
   chgrp bin /usr/people/ralph
   chmod 700 /usr/people/ralph
   ```

The user's account is now locked, and only *root* has access to the user's home account.

```
cramden:x:103:101:Ralph Cramden:/usr/people/cramden:/bin/csh
```

Temporarily Changing Groups

Normally you are a member of only one group at a time. However, you can change groups using the *newgrp*(1) command. Changing groups is sometimes useful for performing administrative tasks.

The superuser can belong to any group listed in the */etc/groups* file. Other users must be listed as members of a group in order to temporarily change groups with *newgrp*.

You can belong to multiple groups simultaneously by invoking the *multgrps*(1) command. In this case, files you create have their group IDs set to the group you were in before you issued the *multgrps* command. You have group access permissions to any file whose group ID matches any of the groups you are in.

You can only change groups to a group you are affiliated with in the */etc/groups* file. To determine which groups you belong to, use the *id*(1) command.

Changing User Information

This section covers the following procedures:

This section tells you how to change the values for an individual user's login information. You cannot use these commands for a login you installed as an NIS-type entry. If the login account is an NIS type, you must change the master */etc/passwd* file on the network server. See the chapter "The Network Information Service (NIS)" in the *NFS User's Guide* for more information about NIS.

Changing a User's Login Name

To change a user's login name, perform the following steps:

1. Edit the */etc/passwd* file and change the entry for the user's login to reflect the new login name. For example, to change the login name *cramden* to *ralph*, find the line:

   ```
   cramden:x:103:101:Ralph Cramden:/usr/people/cramden:/bin/csh
   ```

   Change the name field and the default directory field as follows:

   ```
   ralph:x:103:101:Ralph Cramden:/usr/people/ralph:/bin/csh
   ```

For consistency's sake, the home directory should always have the same name as the user account. If your system has shadow passwords enabled, you will see the character "x" in place of the encoded password in the user's entry. For more information on shadow passwords, see "Creating a Shadow Password File" on page 439.

When you save and exit the file, you may see an error message that the file is "read-only" or that write permission is denied. This is a protection that IRIX puts on the */etc/passwd* file. Use the command:

```
:w!
```

in the *vi*(1) editor to override this protection. If you are using *jot*(1), the file can be saved with no difficulty. For complete information on the commands available in *vi*(1), see the *vi*(1) reference page.

2. If your system has shadow passwords enabled, edit the */etc/shadow* file next. Look for the line that begins with the user name you wish to change. In this example, the line would look like this:

   ```
   cramden:XmlGDVKQYet5c:::::::
   ```

   Change the name in the entry to "ralph" as follows:

   ```
   ralph:XmlGDVKQYet5c:::::::
   ```

   When you have made the change, save and exit the file. As with */etc/passwd*, if you are using *vi*(1), you may encounter an error message.

3. Go to the directory that contains the user's home directory and change the name of the home directory to match the user's new login name. Use the command:

   ```
   mv cramden ralph
   ```

4. Since IRIX identifies the files owned by the user "cramden" by the user ID number, rather than by the login name, there should be no need to change the ownership.

Changing a User's Password

Occasionally, a user forgets his or her password. To solve the problem, you must assign that user a temporary password, then have the user change the temporary password to something else. This is because there is no easy way to guess a forgotten password.

To assign a new password, perform these steps:

1. Log in as **root**.
2. Use the *passwd* command to change the password for the user's account.

   For example, if the user *ralph* forgets his password, enter:

   ```
   passwd ralph
   ```

3. Follow the screen prompts: (*The password entered is not echoed.*)

   ```
   New password: 2themoon
   Re-enter new password: 2themoon
   ```

   Because you are logged in as the superuser (*root*), you are not prompted for an old password.

The user's password is now changed to "2themoon." The user should immediately change the password to something else.

Changing a User's Login ID Number

It is not recommended to change user login ID numbers. These numbers are crucial in maintaining ownership information and responsibility for files and processes. However, if for some reason you must change a user's login ID number, perform the following steps:

1. Make a complete backup tape of the user's home directory and any working directories he or she may have on the system.
2. Lock the user's account by placing a number sign (#) at the beginning of the line, and an asterisk (*) in the password field of the */etc/passwd* file. Do not delete the entry, as you will want to keep it as a record that the old user ID number was used and should not be reused. When you are finished, the entry should look like this:

   ```
   # ralph:*:103:101:Ralph Cramden:/usr/people/ralph:/bin/csh
   ```

3. Completely remove the user's home directory, all subdirectories, and any working directories the user may have.
4. Use the following command from your system's root directory to find any other files the user may own on the system and display the filenames on the console:

find . -user *name* **-print**

Archive and remove any files that are found.

5. Create a new user account using the instructions provided in this chapter. It may have the same name as the old account, but it is better to change names at the same time, to avoid confusion. Use the new user ID number.
6. Restore the home directory, working directories, and any other files you located from the backup you made. If necessary, use the *chown*(1) and *chgrp*(1) commands to set the new user ID correctly.

Changing a User's Default Group

A user can be a member of many different groups on your system, but only one group is the default group. This is the group that the user begins with at login time. To change the default group at login time, simply edit the */etc/passwd* file and find the appropriate line. For example:

```
cramden:+:103:101:Ralph Cramden:/usr/people/cramden:/bin/csh
```

To change the default group from 101 to 105, simply change the field in the passwd file entry as follows:

```
cramden:+:103:105:Ralph Cramden:/usr/people/cramden:/bin/csh
```

Be certain before you make any change, however, that group 105 is a valid group in your */etc/groups* file and that the user is a member of the group. For more information on creating groups, see "Adding a Group Using Shell Commands" on page 88.

Changing a User's Comments Field

There is a field in each entry in */etc/passwd* for comments about the user account. This field typically contains the user's name and possibly his or her telephone number or desk location. To change this information, simply edit the */etc/passwd* file and change the information. Note only that you cannot use colon (:) within the comments, since IRIX will interpret these as ending the comments field. For example, consider this entry:

```
cramden:x:103:101:Ralph Crumdin:/usr/people/cramden:/bin/csh
```

It would not be long before Ralph came to the administrator and requested to have the misspelling of his name corrected. In this case, you would change the line to read:

```
cramden:x:103:101:Ralph Cramden:/usr/people/cramden:/bin/csh
```

Changing a User's Default Home Directory

The next field in an */etc/passwd* entry specifies the user's home directory. This is the directory that the user is placed in at login time, and the directory that is entered in the shell variable *$HOME*. For complete information on shell variables, see the reference pages for the shell you are using (typically *csh*(1), *sh*(1), or *ksh*(1)). Home directories are typically placed in */usr/people*, but there is no reason why you cannot select another directory. Some administrators select a different directory to preserve file system space on the */usr* file system, or simply because the users have a strong preference for another directory name. In any case, the procedure for changing the home directory of a user is quite simple. Follow these steps:

1. Log in as **root**.
2. Edit the */etc/passwd* file and look for the user entry you wish to change. For example:

   ```
   cramden:x:103:101:Ralph Cramden:/usr/people/cramden:/bin/
   csh
   ```

   In this example, the home directory is */usr/people/cramden*. If you wish to change the home directory to a recently added file system called *disk2*, change the entry to read:

   ```
   cramden:x:103:101:Ralph Cramden:/disk2/cramden:/bin/csh
   ```

   When you have made your changes, write and exit the file.
3. Create the directory in the new file system and move all of the files and subdirectories to their new locations. When this is done, remove the old home directory and the process is finished.
4. Be sure that you notify your users well in advance if you plan to change their home directories. Most users have personal aliases and programs that may depend on the location of their home directory. As with all major changes to your system's layout, changing home directories should not be done for trivial reasons, as it can cause serious inconvenience to your users.

Changing a User's Default Shell

To change the default shell, follow these steps:

1. Log in as **root**.
2. Edit the */etc/passwd* file and change the field that names the user's default shell. For example, in the *passwd* entry:

   ```
   ralph:x:103:101:Ralph Cramden:/usr/people/ralph:/bin/csh
   ```

   The default shell is */bin/csh*. To change the user's default shell, change the field in the passwd entry to the desired program. For example, to change Ralph's shell to */bin/sh*, edit the line to look like this:

   ```
   ralph:x:103:101:Ralph Cramden:/usr/bin/ralph:/bin/sh
   ```

3. Save and exit the */etc/passwd* file. When the user next logs in, the new default shell will be used.
4. Note that you can use any executable program as the default shell. IRIX simply executes the given program when a user logs in. Note also that using a program other than a command shell such as *csh* or *sh* can cause problems for the user. When the program identified as the default shell in the *passwd* file exits, the user is logged out. Therefore, if you use */bin/mail* as the default shell, when the user exits *mail*, he or she will be logged out and will not be able to perform any other work.

The User's Environment

A user's environment is determined by certain shell startup files. For C shell users, these are the files */etc/cshrc* and, in their home directories, *.cshrc* and *.login*. For Korn Shell users, these are the */etc/profile* file and the *.profile* file in their home directories. For Bourne shell users, these are the files */etc/profile* and, in their home directories, *.profile*.

Shell startup files configure a user's login environment and control aspects of sub-shells created during a login session.

Available Login Shells

The following login shells are provided with IRIX:

/bin/csh This shell provides a history mechanism (that is, it remembers commands you enter); job control (you can stop processes, place them in background, and return them to foreground); and the ability to use wildcard characters to specify file names. Users can construct sophisticated commands and scripts using a C programming language-like syntax. For a complete description of this shell, see the *csh*(1) reference page.

/bin/ksh This shell is an expansion of the best features of the Bourne shell and the C shell, and allows for command line editing, job control, programming from the shell prompt in the shell language, and other features. For a complete description of this shell, see the *ksh*(1) reference page.

/bin/sh This is a simpler shell than *csh* and is also called the *Bourne* shell after its principle designer. Bourne shell does not contain any kind of history mechanism and uses a somewhat different syntax than *csh*. It does make use of wildcard characters and is smaller and faster to invoke than *csh*. For a complete description of this shell, see the *sh*(1) reference page.

/bin/rsh This is a restricted shell, which limits the commands a user can type. The *rsh* command syntax is identical to *sh*, except that users cannot perform the following:

- change directories
- use the *ls*(1) command
- set the shell search path ($PATH)
- specify path or command names containing /
- redirect output (> and >>)

The restrictions against */bin/rsh* are enforced after *profile* has been executed.

For complete information about these shells, see the *ksh*(1), *csh*(1), and *sh*(1) reference pages. The *rsh* restricted shell is described on the *sh*(1) reference page.

Note: Two shells called *rsh* are shipped with IRIX. */bin/rsh* is the restricted shell. The other shell, in */usr/bsd/rsh*, is the Berkeley remote shell. Be careful not to confuse the two.

The various startup files that configure these shells are described in the next sections.

C Shell Files

When a C shell user logs in to the system, three startup files are executed in the following order:

1. The */etc/cshrc* file.

   This is an ASCII text file that contains commands and shell procedures, and sets environment variables that are appropriate for all users on the system. This file is executed by the *login* process.

   A sample */etc/cshrc* is shown below:

   ```
   # default settings for all users
   # loaded <<before>> $HOME/.cshrc
   umask 022
   if ($?prompt) cat /etc/motd
   set mail=$MAIL
   if ( { /bin/mail -e } ) then
    echo 'You have mail.'
   endif
   ```

   In the example, several commands are executed:

   - the message of the day is displayed if a prompt exists
   - the location of the user's mail file is set
   - a message informs the user if he or she has mail

2. The individual user's *.cshrc*.

   This file is similar to */etc/cshrc*, but is kept in the user's home directory. The *.cshrc* file can contain additional commands and variables that further customize a user's environment. For example, use this file to set the shell prompt for the user. The *.cshrc* file is executed whenever a user spawns a subshell. A sample *.cshrc* is shown below:

   ```
   set prompt  = "Get back to work: "
   set filec
   set history = 20
   ```

   In this example, the user's prompt is set, automatic filename completion is turned on, and the length of the history of recently issued commands is set to 20.

3. The *.login* file.

   This is an executable command file that resides in the user's home directory. The *.login* also customizes the user's environment, but is only executed once, at login time. For this reason, you should use this file to set environment variables and to run shell script programs that need be done only once per login session. A sample *.login* is shown below:

   ```
   eval ‘tset -s -Q‘
   umask 022
   stty line 1 erase ’^H’ kill ’^U’ intr ’^C’ echoe
   setenv DISPLAY wheeler:0
   setenv SHELL csh
   setenv VISUAL /usr/bin/vi
   setenv EDITOR /usr/bin/emacs
   setenv ROOT /
   set path = (. ~/bin /usr/bsd /bin /usr/bin /usr/sbin \ /
   usr/bin/X11 /usr/demos /usr/local/bin)
   ```

   In this example, the user's terminal is further initialized with *tset*(1), then the file creation mask is set to 022. Some useful key bindings are set, using the command *stty*(1). The user's default display is set to be on the console screen of the machine called "wheeler." Several important environment variables are set for commonly used utilities and the file system point of reference. Finally, the default path is expanded to include the user's own binary directory and other system directories.

For information on the shell programming commands used in these examples, see the *csh*(1) reference page.

Bourne and Korn Shell Files

When a Bourne or Korn shell user logs in to the system, two startup files are executed in the following order:

1. The */etc/profile* file.

   This is an ASCII text file that contains commands and shell procedures and sets environment variables that are appropriate for all users on the system. This file is executed by the *login* process.

   A sample */etc/profile* is shown below:

```
# Ignore keyboard interrupts.
trap "" 2 3
# Set the umask so that newly created files and
directories will be readable
# by others, but writable only by the user.
umask 022
case "$0" in
*su )
# Special processing for ‘‘su -’’ could go here.
;;
-* )
# This is a first time login.
#
# Allow the user to break the Message-Of-The-Day only.
trap "trap ’’ 2" 2
cat -s /etc/motd
trap "" 2
# Check for mail.
if /bin/mail -e
then
echo "you have mail"
fi
;;
esac
trap 2 3
```

   In the example, several commands are executed:

   - keyboard interrupts are trapped
   - the user's *umask* is set to 022—full permission for the user, read and execute permission for members of the user's group and others on the system

- if the user is logging in for the first time, the message of the day (*/etc/motd*) is displayed, and the user is notified if he or she has mail

2. The individual user's *.profile*.

   This file is similar to */etc/profile*, but is kept in the user's home directory. The *.profile* file can contain additional commands and variables that further customize a user's environment. It is executed whenever a user spawns a subshell.

   A sample *.profile* is shown below:

```
# Set the interrupt character to <Ctrl-C> and do clean
backspacing.
stty intr '' echoe
# Set the TERM environment variable
eval 'tset -s -Q'
# List files in columns if standard out is a terminal.
ls() { if [ -t ]; then /bin/ls -C $*; else /bin/ls $*; fi
}
PATH=:/bin:/usr/bin:/usr/sbin:/usr/bsd:$HOME/bin:.
EDITOR=/usr/bin/vi
PS1="IRIX> "
export EDITOR PATH PS1
```

   In this example:

   - the interrupt character is set to <**Ctrl-C**>
   - the TERM environment variable is set with *tset*(1)
   - a function called *ls* is defined so that when the user enters *ls* to list files in a directory, and the command is issued from a terminal or window, the *ls* command is invoked with the **-C** option
   - the environment variables PATH and EDITOR are set
   - the user's prompt (PS1) is set to *IRIX*>

For information on the shell programming commands used in these examples, see the *ksh*(1) and *sh*(1)reference pages.

Environment Variables

Every shell uses a series of variables that hold information about the shell and about the login account from which it originated. These variables provide information to other processes as well as to the shell itself.

Collectively, these environment variables make up what is called the shell's *environment*. The basic concepts of environment variables and an environment are the same for all types of IRIX shells, although the exact method of creating and manipulating the environment variables differs.

A basic set of environment variables includes where in the IRIX file system to search for commands (PATH), the location of the home directory of the user's account (HOME), the present working directory (PWD), the name of the *terminfo* description used to communicate with the user's display screen or terminal (TERM), and some other variables.

When a process (shell) begins, the *exec*(2) system call passes it an array of strings, called the *environment*.

Since *login* is a process, the array of environment strings is made available to it. To look at your current shell environment, use the *printenv* command. A typical C shell environment might look something like this:

```
LOGNAME=trixie
PWD=/usr/people/trixie
HOME=/usr/people/trixie
PATH=.:/usr/people/trixie/bin:/usr/bsd:/bin:/etc:/usr/sbin:
/usr/bin: /usr/local/bin:
SHELL=/bin/csh
MAIL=/var/mail/trixie
TERM=iris-ansi
PAGER=more
TZ=EST5EDT
EDITOR=emacs
DISPLAY=myhost:0
VISUAL=vi
```

For C shell users, these variables are set in either the */etc/cshrc*, *.cshrc*, or *.login* startup files. For Korn and Bourne shell users, these variables are set in either the */etc/profile* or *.profile* startup files.

The default environment variables that are assigned for C shell users, if no others are set in any of the startup files, are:

- HOME
- PATH
- LOGNAME
- SHELL
- MAIL
- TZ
- USER
- TERM

Other processes use this information. For example, user *trixie*'s terminal is defined as an *iris-ansi* (TERM=iris-ansi). When the user invokes the default visual editor *vi*(1), *vi* checks this environment variable, then looks up the characteristics of an *iris-ansi* terminal.

New variables can be defined and the values of existing variables can be changed at any time with the *setenv* command (C shell only). For example, to change the PAGER variable under C shell, enter:

```
setenv PAGER pg
```

This sets the value of the PAGER environment variable to the command *pg*(1). The PAGER variable is used by *mail*(1).

Bourne and Korn shell users set environment variables like this:

```
$ PAGER=pg ; export PAGER
```

Environment variables can be set on the command line, or in either of the shell startup files */etc/profile* or *$HOME/.profile*.

umask

A system default called *umask* controls the access permissions of any files or directories that you create. The system default for IRIX, without *umask* set, is 022, which sets the following permissions respectively:

user	Full access: read, write, and, if applicable, execute permission. Directories can be executed; that is, you can ''change directories'' into any of your own directories and copy files from them.
group	Anyone in the same group can read and, if applicable, execute other group members' files. Execute permission is turned on for directories. Write permission is turned off.
other	All other people on the system have the same access permissions as group access.

The system default *umask* of 022 is the same as running *chmod 644* on files that you create and *chmod 755* on directories and executable files that you create. Setting your *umask* does not affect existing files and directories. To change the default permission, use the *umask* shell command. Like *chmod*, *umask* uses a three-digit argument to set file permissions. However, the argument to *umask* works the opposite as the argument to *chmod*. The argument to *umask* lowers the access permissions from a maximum of 666 (full access for files) and 777 (full access for directories).

The following command leaves permission unchanged for user, group, and other when you create files and directories:

```
umask 000
```

This command reduces access for other users by 1 (it removes execute permission):

```
umask 001
```

This command reduces access for group by 1 (no execute permission) and for others by 2 (no write permission, but execute is allowed):

```
umask 012
```

This command removes write and execute permission for group and removes all permissions for others:

```
umask 037
```

For more information, see the *umask*(1) reference page.

Special Login Shells

You may want to assign an account a login shell other than one of the system defaults. Reasons for doing this include:

- The need for special-use accounts that require restricted or very specific access to the system
- a user requesting a special shell

You can specify any program as the login shell for an account. For example, you can use a third-party application program as the login shell. Users with this application as a shell log in to the system and are immediately placed in the application. All interaction with the system is through the application, and when the users quit from the application, they are automatically logged out. To restrict access to the system, you can also use a custom shell that you create.

Another example is the *nuucp* account, which uses */usr/lib/uucp/uucico* as a login shell.

Many users have favorite shells, for example the *bash* shell, that they might want you to install. As with any other software, make sure it comes from a reputable source. (*bash* shell is public domain software .) You may wish to back up the system completely before installing the shell, then monitor the system closely for a while to be sure there are no problems with the shell.

For security reasons, you should not blindly accept compiled binaries and install them as login shells on the system (or anywhere else on the system, for that matter). Start with the source code for the shell, make sure there are no security holes in the code, then compile it for your site.

Note that special shells should be located in a file system that is always mounted before users log in to the system. If the file system that contains a

login shell is not mounted, people who use that shell cannot log in to their accounts.

Communicating with Users

There are several ways to communicate with users in the IRIX system, including electronic mail, the message of the day, the remote login message, *news*, *write*, and *wall*.

Electronic Mail

Users can send messages to one another using one of the electronic mail programs provided with IRIX. Media Mail is a graphical mail reader with an extensive on-line help system. Media Mail also has a command line interface for those systems without graphics capability. For a complete discussion of configuring electronic mail, see Chapter 20, "IRIX sendmail," in this guide.

Message of the Day

You can communicate items of broad interest to all users with the */etc/motd* file. The contents of */etc/motd* are displayed on the user's terminal as part of the login process. The login process executes */etc/cshrc* (for the C shell), which commonly contains the command:

```
cat /etc/motd
```

Any text contained in */etc/motd* is displayed for each user every time the user logs in. For this information to have any impact on users, you must take pains to use it sparingly and to remove outdated announcements.

A typical use for the message of the day facility might be

```
5/30: The system will be unavailable from 6-11pm Thursday, 5/
30 while we install additional hardware
```

Be sure to remove the message when it is no longer important.

Remote Login Message

In addition to */etc/motd*, the file */etc/issue* is displayed to whomever logs in to a system over a serial line or the network. If */etc/issue* does not exist, or is empty, no message is displayed.

News

You can set up a simple electronic bulletin board facility with the */usr/news* directory and the *news*(1) command. With *news*, you can post messages of interest about the system. This is not the same system as the publicly distributed Usenet system. Place announcements of interest about the system in the directory */usr/news*. Use one file per announcement, and name each file something descriptive, like "downtime" and "new-network." Use the *news* command to display the items.

You can automatically invoke *news* from a shell startup file, for example from the */etc/cshrc* file. It is a good idea to check for new news items only from a shell startup file, since users may not be ready to read news immediately upon logging in. For example:

```
news -s
```

With the **-s** argument, *news* indicates how many articles there are since you last read news.

When you read news with the *news* command, you can do the following:

read everything
: To read all news posted since the last time you read articles, enter the *news* command with no arguments: **news**

select some items
: To read selected articles, enter the *news* command with the names of one or more items as arguments:

```
news downtime new-network
```

read and delete
: After you run the *news* command, you can stop any item from printing by pressing **<Ctrl-C>** or **<Break>**. Pressing **<Ctrl-C>** or **<Break>** twice stops the program.

ignore everything

If you are too busy to read announcements at the moment, you can read them later. Items remain in */usr/news* until the administrator (*root*) removes them. The list of news items are still displayed each time you log in.

flush all items

There are two ways to catch up with all current news items:

```
touch .news_time
```

This updates the time-accessed and time-modified fields of the *.news_time* file and thus the *news* program no longer considers there are articles for you to read.

This command prints all current articles, but sends the output to */dev/null* so you do not see the articles:

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

This brings you up to date without reading any outstanding articles.

Write to a User

Use the *write* command to write messages to a user on the system. For example:

```
write ralph
```

User *ralph* sees this on his screen:

```
Message from root on brooklyn (console) [ Tue Feb 26
16:47:47 ] ...
```

You can wait for *ralph* to respond, or you can begin typing your message. If the other user responds, you see a similar message on your screen.

Type your message. As you press **<Return>**, each line of your message is displayed on the other user's screen.

Usually a *write* session is a dialogue, where each user takes turns writing. It is considered good etiquette to finish your turn with a punctuation mark on a line by itself, for example:

```
I noticed that you are using over 50 meg of disk space.Is
there anything I can do to help you reduce that?
>
```

Entering the greater-than symbol indicates you are through with your paragraph and are waiting for user *ralph* to respond. The other user should choose a different punctuation character to indicate when he is through with his turn.

You can prevent other users from writing to you with *write* by making their terminal or window unwriteable. Use the *mesg* command:

```
mesg n
```

The *n* argument makes your terminal or window unwriteable, and the *y* argument makes it writable. The superuser can write to any terminal or window, even if the user has made his or her terminal unwriteable with *mesg n*.

The *talk*(1) utility is similar to *write*(1), and is preferred by some users.

Write to All Users

The superuser can use the *wall*(1) command to write to all the users who are logged in on the system. This useful when you need to announce that you are bringing the system down.

To use *wall*, enter:

```
wall
```

Enter your message. Press **<Ctrl-D>** when you are finished, and *wall* sends the message.

You can also compose the message in a file, then send it using *wall*:

```
wall < messagefile
```

The *wall* command is not affected by a user's *mesg* setting. That is, a user cannot stop *wall* from displaying on his or her screen. On a graphics display with multiple windows, the message is displayed in all windows.

The /etc/nologin File

The */etc/nologin* file prevents any user from logging in. This feature of the *login*(1) program is designed to allow the system administrator to have the system running in full multiuser mode, but with no users logged in. This is useful when you wish to perform complete backups of the system or when you want to do some testing that may cause the operating system to halt unexpectedly. Of course, it is always best to do this sort of work during non-peak system usage hours.

To disable logins, simply create a file called *nologin* in the */etc* directory. (You must be logged in as *root* to create files in */etc*.) In addition to disallowing logins, the *login* program will display the contents of */etc/nologin* when it denies access to the user. To allow logins again, simply remove the */etc/nologin* file. A suggested format for the message in */etc/nologin* is:

```
The system is unavailable for a few moments while we perform
some routine maintenance. We will be done shortly and regret
any inconvenience this may cause you. -Norton
```

Anticipating User Requests

The following suggestions apply mostly to servers, although they are still applicable to workstations.

Keep a Log

In addition to the system log, as described in Chapter 2, “Operating the System,” you may find it helpful to keep a user trouble log. The problems that users encounter fall into patterns. If you keep a record of how problems are resolved, you do not have to start from scratch when a problem recurs. Also, a system log can be very useful for training new administrators in the specifics of your local system, and for helping them learn what to expect.

Hardware Affects Software

Be aware that changing hardware configurations can affect the system, even if the change you make seems simple. Make sure you are available to help users with problems after the system is changed in any way.

Leaving Users Stranded

Before you upgrade your system to new software, check your user community to see which parts of the old software they use, and if they might be inconvenienced by the upgrade. Often users need extra time to switch from one release of an application to a newer version.

If possible, do not strand your users by completely removing the old software. Try to keep both versions on the system until everyone switches to the new version.

Reporting Trouble

Provide a convenient way for your users to report problems. For example, set up a "trouble" mail alias, so that users with problems can simply send mail to *trouble* for assistance.

Creating Reference Pages

Reference pages are online reference manual entries. A full set of reference pages for the programs, utilities, and files in the standard IRIX distribution is provided on-line, and these pages are available through the *man*(1) command. In addition, you can create your own custom reference pages using the following procedure. Any time you create a script, utility, program, or application for your users, you should also create a reference page. This provides a convenient way for your users to learn to use your new tools, and also makes future maintenance easier.

Not all sites will have the optional Documenter's Workbench software product installed, but you can create a facsimile of a manual page using only the text editor of your choice. See the following section for details.

Creating a Pure-Text Man Page using vi

Note: To create a pure-text man page without Documenter's Workbench (no embedded *nroff*(1) commands that would format the text) simply use the *vi* editor (or the editor of your choice) and document your script or program according to the style found in the standard reference pages. Name your reference page file after the script or program it documents with the suffix ".l" at the end of the file name to designate the page as a local reference page.

Note: Use the letter "l" as your suffix, not the numeral one "1."

When you have completed your reference page, you must place it in the */usr/man* directory structure for the *man*(1) command to be able to display the new page. Place the man pages in a local directory, such as */usr/man/manl*. (Again using the character "l" to designate local reference pages.) If it does not already exist, create the directory with this command (you must be logged in as **root**):

```
mkdir /usr/man/manl
```

Long man pages should be packed to save disk space. Use the *pack*(1) command to pack the text file into a more compact form. For example:

```
pack program.1
mv program.1.z /usr/man/manl/program.z
```

Note: The *man* program automatically unpacks the pages for reading.

Chapter 4

The Command (PROM) Monitor

Chapter 4 covers the Command Monitor (also known as the PROM Monitor). From the Command Monitor, you can boot programs other than the default operating system. You can also check your hardware inventory and boot the special Stand-Alone Shell to begin major software installations. Specific tasks covered here include:

- *How to enter the Command Monitor*
- *Commands available from the Command Monitor*
- *Booting programs from the Command Monitor*

Chapter 4

The Command (PROM) Monitor

This chapter describes the Command (PROM) Monitor, which controls the boot environment for all IRIS workstations or servers. With the Command Monitor, you can boot and operate the CPU under controlled conditions, run the CPU in Command Monitor mode, and load programs (for example, the operating system kernel, */unix*).

PROM stands for Programmable Read-Only Memory. PROM chips are placed in your computer at the factory with software programmed into them that allows the CPU to boot and allows you to perform system administration and software installations. The PROMs are not part of your disk or your operating system; they are the lowest level of access available for your system. You cannot erase them or bypass them.

Note that there are numerous minor differences between machines, and you should refer to your owner's guide for information specific to your machine.

This chapter contains information on the following topics:

- Basic instruction on entering the Command Monitor. See "How to Enter the Command (PROM) Monitor" on page 118.
- A summary of the commands available through the general Command Monitor. See "Summary of Command Monitor Commands" on page 118.
- How to get help while using the Command Monitor. See "Getting Help in the Command Monitor" on page 120.
- Instructions for convenient use of the Command Monitor. See "Using Command Monitor Commands" on page 120 and "Running the Command Monitor" on page 123.
- Instructions for booting programs from the Command Monitor. See "Booting a Program from the Command Monitor" on page 132.

How to Enter the Command (PROM) Monitor

To get into the Command Monitor on most machines, follow these steps:

1. Reboot the system with the *reboot*(1M) command, or if it is already switched off, turn it on.

   You see the following prompt:

   ```
   Starting up the system....

   To perform system maintenance instead, press <Esc>
   ```

2. Press the **<Esc>** key. You see the following menu:

   ```
   System Maintenance Menu
   1   Start System
   2   Install System Software
   3   Run Diagnostics
   4   Recover System
   5   Enter Command Monitor
   ```

3. Enter the numeral **5**, and press **<Return>**. You see the Command Monitor prompt:

   ```
   >>
   ```

4. You have entered the Command Monitor.

Summary of Command Monitor Commands

Table 4-1 summarizes the Command Monitor commands and gives each command's syntax.

Table 4-1 Command Monitor Command Summary

Command	Description	Syntax
auto	Boots default operating system (no arguments)	auto
boot	Boots with arguments	**boot** [**-f** *file*][**-n**][**args**]
checksum		checksum RANGE

Table 4-1 (continued) Command Monitor Command Summary

Command	Description	Syntax
disable	Disables console; console can be gfx(0), tty(0), or tty(1)	**disable** *console_device*
dump		dump [-(b | h | w)] [-(o | d | u | x | c | B)] RANGE
eaddr		eaddr [xx:xx:xx:xx:xx:xx]
enable	Enables console; console can be gfx(0), tty(0), or tty(1)	**enable** console_device
exit	leave Command Monitor	exit
fill		fill [-(b | h | w)] [-v val] RANGE
get		g [-(b | h | w)] ADDRESS
go	Transfers program execution to <*pc*>or to entry point of last booted program if <*pc*> omitted	**go** [**INITIAL_PC**]
help or **?**	Prints a Command Monitor command summary	**help** [*command*] **?** [*command*]
hinv (inventory)	Prints an inventory of hardware on the system	hinv
init	Initializes the Command Monitor	**init**
mcopy		mcopy [-(b | h | w)] FROMRANGE TO
mcmp		mcmp [-(b | h | w)] FROMRANGE TO
mfind		mfind [-(b | h | w)] [-n] RANGE VALUE
passwd	Sets PROM password	passwd

Table 4-1 (continued) Command Monitor Command Summary

Command	Description	Syntax
put		p [-(b | h | w)] ADDRESS VALUE
printenv	Displays the current environment variables	**printenv** [*env_var_list*]
resetenv	Resets all environment variables to default	resetenv
setenv	Sets environment variables	**setenv** *env_var string*
unsetenv	Unsets an environment variable	**unsetenv** *env_var*
version	Displays Command Monitor version	version

Getting Help in the Command Monitor

The question mark (**?**) command displays a short description of a specified command. If you do not specify a command, the **?** command displays a summary of all Command Monitor commands. To get help, type either **help** or a question mark (**?**).

help [*command*]

? [*command*]

Using Command Monitor Commands

The following sections cover these subjects:

- The command syntax notation that this chapter uses
- The function of the commands listed in Table 4-1

Using the Command Line Editor in the Command Monitor

You can edit on the command line by using the commands shown in Table 4-2.

Table 4-2 Command Monitor Command Line Editor

Command	Description
<ctrl-h>, **<del>**,	Deletes previous character or **<backspace>**
<ctrl-u>	Deletes entire line; question mark (**?**) prompts for corrected line
<ctrl-c>	If a command is executing, kills current command

Syntax of Command Monitor Commands

The Command Monitor command syntax is designed to resemble the syntax of commands used with the IRIX operating system. This chapter uses IRIX notation for command descriptions:

- **Boldface** words are literals. Type them as they are shown.
- Square brackets ([]) surrounding an argument means that the argument is optional.
- Vertical lines (|) separating arguments mean that you can specify only one optional argument within a set of brackets.
- *file* means that you must specify a file name. A file name includes a device specification as described in "Syntax of Command Monitor File Names" on page 121.

Syntax of Command Monitor File Names

When you specify file names for Command Monitor commands, use this syntax:

device([*cntrlr*,[*unit*[,*partition*]]])*file*

- *device* specifies a device driver name known to the PROM.
- *cntrlr* specifies a controller number for devices that may have multiple controllers.
- *unit* specifies a unit number on the specified controller.
- *partition* specifies a partition number within a unit.
- *file* specifies a pathname for the file to be accessed.

If you do not specify *cntrlr, unit*, and *partition*, they default to zero. The notation shows that you can specify only a *cntrlr*, a *cntrlr* and *unit,* or all three variables. The commas are significant as place markers. For example, the root partition (partition 0) on a single SCSI disk system is shown as:

```
dksc(0,1,0)
```

where:

- dksc indicates the SCSI driver
- The first 0 indicates SCSI controller 0
- The 1 indicates drive number 1 on SCSI controller 0
- The final 0 indicates partition 0 (root partition) on drive 1 on SCSI controller 0.

The */usr* partition (partition 3) on the same disk would be written as:

```
dksc(0,1,3)
```

The Command Monitor defines the devices shown in Table 4-3.

Table 4-3 Device Names for Command Monitor Commands

Device Name	Description
dkip	the ESDI disk controller (ips in IRIX)
dksc	the SCSI disk controller (dks in IRIX)
tpsc	the SCSI tape controller (tps in IRIX)
xyl	the SMD disk controller (xyl in IRIX)
ipi	the IPI disk controller (ipi in IRIX)

Table 4-3 (continued) Device Names for Command Monitor Commands

Device Name	Description
tty	CPU board duart
tty(0)	the local console
tty(1)	the remote console
gfx	the graphics console
console	the "pseudo console" which may be one of gfx(0), tty(0), or tty(1). See "Enabling a Console in the Command Monitor" on page 124
bootp	Ethernet controller using bootp and TFTP protocols
tpqic	the quarter-inch QIC02 tape drive

The PROM device notation is different from IRIX device notation. Certain environment variables (such as *root* and *swap*) are passed to higher level programs, and often require IRIX notation for the */dev* device name. For example, in PROM notation, an ESDI disk partition most commonly used for swap is written:

```
dkip(0,0,1)
```

In IRIX notation, the same disk is:

```
ips0d0s1
```

Running the Command Monitor

This section describes the commands that you use to run the Command Monitor. The Command Monitor accepts the commands listed in Table 4-1, "Command Monitor Command Summary," on page 118.

Enabling a Console in the Command Monitor

The Command Monitor can support a local console and a remote console (alone or simultaneously) through a serial port. The *enable* command enables a device that you want to use as a console. The Command Monitor accepts commands from the enabled console and displays output to that console.

`enable` *console_device*

console_device can be **`gfx(0)`** for the graphics console, **`tty(0)`** for a terminal on port 1, or **`tty(1)`** for a terminal on port 2. The *disable* command works exactly the same way, disabling the specified console device.

Reinitializing the Processor from the Command Monitor

The *init* command reinitializes the processor from PROM memory, and returns you to the monitor program.

Setting a PROM Password

Your system may have a facility that allows you to require a password from users who attempt to gain access to the Command Monitor.

To determine if your system supports PROM passwords, select option 5 from the System Maintenance Menu to enter the Command Monitor. You see the Command Monitor prompt:

```
Command Monitor. Type "exit" to return to the menu.
>>
```

Enter the command:

`help`

The system prints a list of available commands for the Command Monitor. If the *passwd* command is among those listed, your system supports the PROM password. If it is not listed, your system hardware does not support passwording. If you would like to upgrade your system to support passwording, please contact your sales representative.

If your system supports PROM passwording, issue the *passwd* command:

```
passwd
```

You see the prompt:

```
Enter new password:
```

Enter the password you want for your machine and press **<Return>**. You see the prompt:

```
Confirm new password:
```

Enter the password again, exactly as you typed it before. If you typed the password the same as the first time, you next see the Command Monitor prompt again. If you made a mistake, the system prints an error message and you must begin again. If you see no error message, your password is now set. Whenever you access the Command Monitor, you will be required to enter this password.

It is very important that you choose and enter your password carefully, because if it is entered incorrectly or forgotten, you may have to remove a jumper on the CPU board of your system. This procedure is different for each system type, and is described in your owner's guide. Some systems, though, allow you to reset the PROM password from IRIX by logging in as *root* and issuing the following command:

```
nvram passwd_key ""
```

The quotation marks with no characters or space between them are essential to remove the PROM password. You must be *root* to perform this operation.

Copying Hard Disks From the Command Monitor

You can copy a hard disk onto another hard disk easily through the Command Monitor. You may want to do this to create a backup disk in case of failure, or perhaps you have a specific software setup that you wish to copy for a new system. In order for this procedure to work correctly, and for the new disk to be useful, the disks must be of identical size and manufacture. Also, the system that is to receive the new disk must use the same CPU and graphics board set that the existing system uses, because the kernel is custom configured for CPU and graphics type, and the kernel will be copied exactly to the new disk.

Follow these steps:

1. Bring your system down and install the new disk in the space provided for an additional disk. Select a SCSI device number that is not currently in use for the new disk. For this example, we will use device 2 on SCSI controller 0 (the integral SCSI controller) for the new disk, and device 1 on SCSI controller 0 for the disk to be copied.
2. Boot the system to the System Maintenance Menu.
3. Select option 5 from the System Maintenance Menu. You will see the Command Monitor prompt:

   ```
   >>
   ```

4. Give the command to load the sash:

   ```
   boot
   ```

5. From the sash prompt, give the command:

   ```
   cp -b 128k dksc(0,1,10) dksc(0,2,10)
   ```

You see a "read error" message when the copy is complete. This is the normal message that tells you that the copying software has read the entire disk. If you see a "write error" message, there was an error copying the disk and you will probably have to start over or install the disk by more conventional means.

The Command Monitor Environment

The Command Monitor maintains an environment, which is a list of variable names and corresponding values (the values are actually text strings). These *environment variables* contain information that the Command Monitor either uses itself or passes to booted programs. The system stores some environment variables—those that are important and unlikely to change frequently—in non-volatile RAM (nvram). If you turn off power to the machine or press the **Reset** button, the system remembers these variables. When you change the setting of these variables using the *setenv* command, the PROM code automatically stores the new values in non-volatile RAM.

You can also use the */etc/nvram* command to set or print the values of non-volatile RAM variables on your system. For complete information on the *nvram* command, see the *nvram*(1M) reference page.

Table 4-4, "Variables Stored in Non-volatile RAM," on page 128 shows a list of the environment variables that the system stores in non-volatile RAM.

Several environment variables also exist that affect IRIX's operation. These are not stored in non-volatile RAM, but they do affect the operation of the PROM and of IRIX.

See Table 4-5, “Environment Variables That Affect the IRIX Operating System,” on page 130. Table 4-4 lists non-volatile RAM variables.

Table 4-4 Variables Stored in Non-volatile RAM

Variable	Description
netaddr	Specifies the local network address for booting across the Ethernet. See the *bootp* protocol.
dbaud	Specifies the diagnostics console baud rate. You can change it by setting this variable (acceptable rates include 75, 110, 134, 150, 300, 600, 1200, 2400, 4800, 9600, and 19200), or by pressing the **<Break>** key. IRIS uses the *dbaud* rate for the diagnostics console during the entire system start-up. Pressing the **<Break>** key changes the baud rate only temporarily; the baud rate reverts to the value specified in *dbaud* or *rbaud* when you press the reset switch or issue an *init* command.
rbaud	Specifies the remote console baud rate. The list of acceptable baud rates is the same as for *dbaud*, above.
bootfile	Specifies the name of the file to use for autobooting, normally a stand-alone shell (sash).

Table 4-4 (continued) Variables Stored in Non-volatile RAM

Variable	Description
bootmode	Specifies the type of boot. The options have these meanings: c - performs a complete cold autoboot, using the file pointed to by the bootfile variable to boot the kernel; boots sash, then boots kernel; runs power-on diagnostics. m - (default) goes straight to the Command Monitor; clears memory; runs power-on diagnostics. d - go straight to the Command Monitor; do not clear memory; do not run power-on diagnostics (on IRIS-4D 100, 200 and 300 series systems, this has the same effect as bootmode m).
console	Specifies which console to use. The options have these meanings: G - graphics console with the Silicon Graphics, Inc., logo in the upper left corner g - (default) graphics console without the Silicon Graphics logo
root	Specifies (in IRIX notation, such as ips0d0s0) the disk that contains the root (/) file system.
keybd	Specifies the type of keyboard used. The default is "df"; it should not be more than two characters. This variable provides a hook to override the normal system mechanism for determining the kind of keyboard installed in the system.

Table 4-5 lists Command Monitor environment variables that directly affect the operating system. Note that these variables are not stored in non-volatile

RAM and are discarded if the machine is powered down.

Table 4-5 Environment Variables That Affect the IRIX Operating System

Variable	Description
showconfig	Prints extra information as IRIX boots. If set through *setenv*, its value must be *istrue*.
initstate	Passed to IRIX, where it overrides the initdefault line in */etc/inittab*. Permitted values are s and the numbers 0-6. See *init*(1M).
swap	Specifies in IRIX notation the swap partition to use. If not set, it defaults to the partition configured into the operating system, which is normally partition 1 on the drive specified by the root environment variable.
path	Specifies a list of device prefixes that tell the Command Monitor where to look for a file, if no device is specified.
verbose	Tells the system to display detailed error messages.

When you boot a program from the Command Monitor, it passes the current settings of all the environment variables to the booted program.

Displaying the Current Environment Variables

The *printenv* command displays the Command Monitor's current environment variables.

`printenv` [*env_var_list*]

To change (reset) the variables, see the next section.

Changing Environment Variables

The *setenv* command changes the values of existing environment variables or creates new environment variables.

`setenv` *env_var string*

env_var is the variable you're setting, and *string* is the value you assign to that variable. To see the current monitor settings, use *printenv.*

When you use *setenv* to change the value of one of the stored environment variables in Table 4-4, the system automatically saves the new value in non-volatile RAM. You do not need to re-enter the change the next time the machine is turned off and then on again.

Setting the Keyboard Variable

If the *keybd* variable is set to anything but the default *df*, the appropriate keyboard translation table is loaded from the volume header of the hard disk. If the table is missing or unable to load, then the default table stored in the PROMs is used. The *keybd* variable can be set to any value, but the keyboard translation table should be loaded from the volume header on the hard disk. This variable overrides the normal system mechanism for determining the kind of keyboard installed in the system. You should not change this variable unless you are performing keyboard diagnostics. Table 4-6 lists *keybd* variables suggested for international keyboards:

Table 4-6 *keybd* Variables for International Keyboards

Variable	Description
be	Belgian
da	Danish
de	German
df	the default
fr	French
it	Italian
no	Norwegian
sf	Swiss-French
sd	Swiss-German
es	Spanish

Table 4-6 (continued) *keybd* Variables for International Keyboards

Variable	Description
sv	Swedish
uk	United Kingdom
us	United States (available on all models)

Removing Environment Variables

The *unsetenv* command removes the definition of an environment variable.

`unsetenv` *env_var*

env_var is the variable whose definition you are removing (see *setenv*, above). Note that variables stored in non-volatile RAM cannot be unset.

Booting a Program from the Command Monitor

This section describes each Command Monitor boot command and shows you how to use it. When you reboot or press the **Reset** button, you start up the Command Monitor. Do not press the **Reset** button under normal circumstances, that is, when the workstation is running IRIX.

Booting a Default File

The *auto* command reboots the operating system. It uses the default boot file as though you were powering up the CPU. At the Command Monitor prompt (>>), type:

```
auto
```

The PROM's environment variable *bootfile* specifies the default boot file. In addition, you must set the environment variable *root* to the disk partition that IRIX uses as its root file system. The *auto* command assumes that the desired image of IRIX resides on the partition specified by *root* of the drive specified in the environment variable *bootfile*.

The *bootfile* name can contain no more than 14 characters. To select a different boot file, see "Changing Environment Variables" on page 130.

Booting a Specific Program

The *boot* command starts the system when you want to use a specific boot program and give optional arguments to that program. The syntax of the *boot* command is:

boot[**-f** *program*][**-n**][*args*]

-f specifies the program you want to boot. The program name must contain fewer than 20 characters. If you do not specify this option, the environment variable *bootfile* specifies the default program. *boot* normally loads *sash*.

When you specify a program, you can include a device specification. If you don't, the Command Monitor uses the device specifications in the environment variable *path*. The Command Monitor tries in turn each device that you specify in *path*, until it finds the program you request, or until it has tried all the devices listed in *path*.

- **-n** means no go: it loads the specified program, but does not transfer control to it. Instead, **-n** returns you to the Command Monitor command environment.
- *args* are variables that the Command Monitor passes to the program you're booting. For an *arg* that starts with a hyphen (-), you must prepend an additional hyphen so that the Command Monitor doesn't think that the argument is intended for itself. The Command Monitor removes the extra hyphen before it passes the argument to the booted program. For more information, see "Booting the Standalone Shell" on page 134.

For example, to boot the disk formatter/exerciser program (*fx*) from the cartridge tape drive, use this command:

```
boot -f tpsc(,7,)fx
```

Without any arguments, *boot* loads the program specified in *bootfile*.

Booting the Standalone Shell

The Command Monitor has been designed to keep it independent of operating systems and as small as possible. Therefore, the Command Monitor cannot directly boot files residing in IRIX or other operating system file trees. However, the Command Monitor does provide a two-level boot mechanism that lets it load an intermediary program that does understand file systems; this program can then find and load the desired boot file. The program is called the *standalone shell*, and is referred to as *sash*. *sash* is a reconfigured and expanded version of the Command Monitor program, and includes the modules needed to handle operating system file structures. It also has enhanced knowledge about devices.

After the system software is installed, a copy of *sash* is located in the volume header of the first disk. The header contains a very simple file structure that the Command Monitor understands. You can also boot *sash* from tape or across the network if need be. To boot *sash* from your disk, shut down the system, and when you see the message:

```
Starting up the system...
To perform system maintenance instead, press Esc
```

Press the escape key. You may have to enter your system's Command Monitor password, if your system has one. Next, you see a menu similar to the following:

```
System Maintenance Menu
      (1) Start System
      (2) Install System Software
      (3) Run Diagnostics
      (4) Recover System
      (5) Enter Command Monitor
```

Select option 5, "Enter Command Monitor" from the System Maintenance Menu. You see the following message and prompt:

```
Command Monitor. Type "exit" to return to the menu.
>>
```

To boot the standalone shell (*sash*), enter the command:

```
boot -f sash
```

sash operates in interactive command mode. You see the sash prompt:

```
sash:
```

To use the multi-level boot feature, set the PROM environment variable *bootfile* to refer to a specific copy of *sash*. In normal configurations, setting *bootfile* to *dkip(0,0,8)sash* tells the Command Monitor to load *sash* from the ESDI disk controller 0, disk unit 0, partition 8 (the volume header). Use this syntax:

```
setenv bootfile "dkip(0,0,8)sash"  for ESDI drives
setenv bootfile "dksc(0,1,8)sash"  for SCSI drives
setenv bootfile "xyl(0,0,8)sash"   for SMD drives
setenv bootfile "ipi(0,0,8)sash"   for IPI drives
```

Then issue a boot command, as in this example for an ESDI drive:

```
boot dkip()unix initstate=s
```

The following actions take place:

- *boot* loads *dkip(0,0,8)sash*, as specified by *bootfile*, since the boot command doesn't contain a **-f** argument. (A **-f** argument would override the default specified by *bootfile*.)
- *sash* gets two arguments: **dkip()unix** and **initstate=s**, which brings the IRIS up in single-user mode. (Note that the Command Monitor removes the leading hyphen [-] from any argument, so if you use the next layer of software, and need an argument with a leading hyphen, you should put two hyphens in front of it.)
- *sash* loads the file specified by the first argument (**dkip()unix**) and passes the next argument to that file.

Do not issue the *auto* command from *sash* with the *bootfile* set as shown above. If you do, the system tries to boot *sash* over itself and will exit with an error.

To be able to use the *auto* command from *sash*, set *bootfile* to refer to the kernel, for example, *dkip()unix*. Even better, return to the PROM level to use the *auto* command.

Booting across the Network

At the heart of the operation of diskless workstations is the *bootp* protocol. *bootp* is a DARPA standard protocol supported on all IRIS workstations. One of the devices that the Command Monitor can use for booting is the Ethernet network. Silicon Graphics provides a TCP/IP boot protocol that lets you boot files that reside on another host in the network, if the other host supports the booting protocol. The network booting protocol is the *bootp* protocol. It is a datagram protocol that uses the User Datagram Protocol (UDP) of TCP/IP to transfer files across the Ethernet network.

To boot across the network, you must first determine the Internet address of the machine you want to boot. The Internet address is a number assigned by the network administrator of the network to which the system is attached. The format of the number is four decimal numbers between 0 and 255, separated by periods; for example:

```
192.20.0.2
```

Use the *setenv* command to set the *netaddr* environment variable to this address; for example:

```
setenv netaddr 192.20.0.2
```

Booting across the Network with bootp

Once you have set the *netaddr* environment variable, you can use *bootp* to refer to a remote file by using a file name of the form:

bootp()[*hostname*:] *path*

- *hostname* is the name of the host where the file resides. The specified host must run the *bootp* server daemon, *bootp*. If you omit *hostname*, *bootp* broadcasts to get the file from any of the hosts on the same network as the machine making the request. The first host that answers fills the request. Only hosts that support *bootp* can respond to the request. It is safe to omit the hostname only when you know that the path is unique to a particular host, or when you know that all the copies of the file are interchangeable.

hostname can be the name of a host on a different Ethernet network from the machine that you are booting, if a *gateway* on the local Ethernet network provides a route to the remote host. The gateway must be an IRIS workstation running a *bootp* server that you have configured to do cross-network forwarding.

For more information about booting through gateways, see *bootp*(1M). For more information about the */etc/inetd.conf* configuration file, see *inetd*(1M).

- *path* is the pathname of a file on the remote host. For example, this command:

  ```
  boot -f bootp()wheeler:/usr/local/boot/unix
  ```

  boots the file */usr/local/boot/unix* from the remote host *wheeler*. The command:

  ```
  boot -f bootp()/usr/alice/help
  ```

  boots the file */usr/alice/help* from any host on the network responding to the *bootp* broadcast request that has a file of that name.

To configure the gateway to permit cross-network forwarding, follow these steps:

1. Log in as **root** or become the superuser by issuing the *su* command.
2. Edit the file */etc/inetd.conf* on the gateway machine. This file configures the *bootp* server, which is started by the *inetd*(1M) daemon.
3. Change the *bootp* description so that *inetd* invokes *bootp* with the **-f** flag. Find this line:

   ```
   bootp  dgram  udp  wait  root  /usr/etc/bootp  bootp
   ```

   Add the **-f** flag to the final *bootp* on the line:

   ```
   bootp  dgram  udp  wait  root  /usr/etc/bootp  bootp  -f
   ```

4. Change the *tftp* configuration line in one of the following ways:

   Remove the **-s** flag from the argument list for *tftpd*:

   ```
   tftp   dgram  udp   wait  guest   /usr/etc/tftpd  tftpd -s
   ```

   This allows *tftpd* access to all publicly readable directories. If you are concerned about a possible security compromise, you can instead explicitly list the directories to which *tftpd* needs access. In this case, you need to add */usr/etc*:

   ```
   tftp dgram udp wait guest /usr/etc/tftpd tftpd -s /usr/etc
   ```

   See *tftpd*(1M) and *tftp*(1C) for more information.

5. Signal *inetd* to re-read its configuration file.

   ```
   killall -1 inetd
   ```

Booting from a Resource List

To tell the Command Monitor to load standalone commands from various resources (such as disks or other devices), set the *path* environment variable. (See "Changing Environment Variables" on page 130.) Set the *path* variable as follows:

setenv path *"device_name alternate_path*"

For example, issue the following commands in order:

```
setenv path "dkip(0,0,8) bootp()/altdir/altbootfile"
```

This causes the Command Monitor to boot the file *dkip(0,0,8)altbootfile*. If that file fails, the Command Monitor boots *bootp()/altdir/altbootfile*. If that file also fails, the Command Monitor prints the message "`command not found`". Note that pathnames are separated with spaces. If the device specification is contained within a command or by *bootfile*, the Command Monitor ignores *path*. Only *bootp* or volume headers are understood by the PROM.

Chapter 5

Tuning System Performance

Chapter 5 describes the process by which you can improve your system performance. Your system comes configured to run as fast as possible under most circumstances. However, if your use of the system is out of the ordinary, you may find that adjusting certain parameters and operating system values may improve your total performance, or you may wish to optimize your system for some feature, such as disk access, to better make use of the graphics features or your application software. Topics described in this chapter include:

- *Measuring system performance.*
- *Improving general system performance.*
- *Tuning for specific hardware and software configurations.*

Chapter 5

Tuning System Performance

This chapter describes the basics of tuning the IRIX operating system for the best possible performance for your particular needs. Information provided includes the following topics:

- General information on system tuning and kernel parameters. See "Theory of System Tuning" on page 141.
- Tuning applications under development. See "Application Tuning" on page 144.
- Observing the operating system to determine if it should be tuned. See "Monitoring the Operating System" on page 149.
- Tuning and reconfiguring the operating system. See "Tuning The Operating System" on page 163.

Theory of System Tuning

The standard IRIX System configuration is designed for a broad range of uses, and adjusts itself to operate efficiently under all but the most unusual and extreme conditions. The operating system controls the execution of programs in memory and the movement of programs from disk to memory and back to disk.

The basic method of system tuning is as follows:

1. monitor system performance using various utilities,
2. adjust specific values (for example, the maximum number of processes),
3. reboot the system if necessary, and
4. test the performance of the new system to see if it is improved.

Note that performance tuning cannot expand the capabilities of a system beyond its hardware capacity. You may need to add hardware, in particular another disk or additional memory, to improve performance.

Files Used for Kernel Tuning

Table 5-1 lists the files/directories used for tuning and reconfiguring a system.

Table 5-1 Files and Directories Used for Tuning

File/Directory:	Purpose:
*/var/sysgen/system/**	File defining software modules
/var/sysgen/mtune/*	directory containing files defining tunable parameters
/var/sysgen/stune	File defining default parameter values.
/var/sysgen/boot/*	Directory of object files
/unix	File containing kernel image

Typically you tune a parameter in one of the files located in the *mtune* directory (for example, the *kernel* file) by using the *systune*(1M) command.

Overview of Kernel Tunable Parameters

Tunable parameters control characteristics of processes, files, and system activity. They set various table sizes and system thresholds to handle the expected system load. If certain system structures are too large, they waste memory space that would otherwise be used for other processes and can increase system overhead due to lengthy table searches. If they are set too low, they can cause excessive I/O, process aborts, or even a system crash, depending on the particular parameter.

This section briefly introduces the tunable parameters. Appendix A, “IRIX Kernel Tunable Parameters,” describes each parameter, gives its default value, provides suggestions on when to change it, and describes problems you may encounter.

The Types of Parameters

Tunable parameters are specified in separate configuration files in the */var/sysgen/mtune* directory. See the *mtune*(4) reference page.

The default values for the tunable parameters are usually acceptable for most configurations for a single-user workstation environment. However, if you have a lot of memory or your environment has special needs, you may want to adjust the size of a parameter to meet those needs. A few of the parameters you may want to adjust are listed below.

nproc	defines the maximum number of processes, system-wide. This parameter is typically auto-configured.
maxup	defines the maximum number of processes per UID.
rlimit-core-cur	the maximum size of a core file.
rlimit-data-cur	the maximum amount of data space available to a process.
rlimit-fsize-cur	the maximum file size available to a process.
rlimit-fsize-cur	the maximum file size available to a process.
rlimit-nofile-cur	the maximum number of file descriptors available to a process.
rlimit-rss-cur	the maximum resident set size available to a process.
rlimit-vmem-cur	the maximum amount of mapped memory for a process.
sshmseg	specifies the maximum number of attached shared memory segments per process.

Application Tuning

You can often increase system performance by tuning your applications to more closely follow your system's resource limits. If you are concerned about a decrease in your system's performance, you should first check your application software to see if it is making the best use of the operating system. If you are using an application of your own manufacture, there are steps you can take to improve performance. Even if a commercially purchased application is degrading system performance, you can identify the problem and use that information to make any decisions about system tuning or new hardware, or even simply when and how to use the application. The following sections explain how to examine and tune applications. The rest of this chapter assumes that your applications have been tuned as much as possible according to these suggestions.

Checking an Application

If your system seems slow, for example, an application runs slowly, first check the application. Poorly designed applications can perpetuate poor system performance. Conversely, an efficiently written application means reduced code size and execution time.

A good utility to use to try to determine the source of the problem is the *timex*(1) utility. *timex* will report how a particular application is using its CPU processing time. The format is:

`timex -s` *program*

which shows *program*'s real (actual elapsed time), user (time process took executing its own code), and sys (time of kernel services for system calls) time. For example:

```
timex -s ps -el
```

The above command executes the *ps -el* command and then displays that program's time spent as:

```
real 0.95
user 0.08
sys 0.41
```

Tuning an Application

There are many reasons why an application spends a majority of its time in either user or sys space. For our purposes, suspect excessive system calls or poor locality of code.

Typically, you can only tune applications that you are developing. Applications purchased for your system cannot be tuned in this manner, although there is usually a facility to correspond with the application vendor to report poor performance.

If the application is primarily spending its time in user space, the first approach to take is to tune the application to reduce its user time by using the *pixie*(1) and *prof*(1) commands. See the respective reference pages for more information about these commands. To reduce high *user* time, make sure that the program:

- makes only the necessary number of system calls. Use *timex -s* to find out the number of system calls/second the program is making. The key is to try to keep **scall/s** at a minimum. System calls are those like *read*(2), *exec*(2); they are listed in Section 2 of the reference pages.
- uses buffers of at least 4K for *read*(2) and *write*(2) system calls. Or use the standard I/O library routines *fread*(3) and *fwrite*(3), which buffer user data.
- uses shared memory rather than record locking where possible. Record locking checks for a record lock for every read and write to a file. To improve performance, use shared memory and semaphores to control access to common data (see *shmop*(2), *semop*(2), and *usinit*(3P)).
- defines efficient search paths ($PATH variable). Specify the most used directory paths first, and use only the required entries, so that infrequently used directories aren't searched every time.
- eliminates polling loops (see *select*(2)).
- eliminates busy wait (use *sginap*(0)).
- eliminates system errors. Look at */var/adm/SYSLOG*, the system error log, to check for errors that the program generated, and try to eliminate them.

Run *timex* again. If the application still shows a majority of either user or sys time, suspect excessive paging due to poor "locality" of text and data. An application that has locality of code executes instructions in a localized portion of text space by using program loops and subroutines. In this case, try to reduce high user/sys time by making sure that the program:

- groups its subroutines together. If often-used subroutines in a loaded program are mixed with seldom-used routines, the program could require more of the system's memory resources than if the routines were loaded in the order of likely use. This is because the seldom-used routines might be brought into memory as part of a page.
- has a working set that fits within physical memory. This minimizes the amount of paging and swapping the system must perform.
- has correctly ported FORTRAN-to-C code. FORTRAN arrays are structured differently from C arrays; FORTRAN is column major while C is row major. If you don't port the program correctly, the application will have poor data locality.

After you tune your program, run *timex* again. If sys time is still high, tuning the operating system may help reduce this time.

There are a few other things you can do to improve the application's I/O throughput. If you are on a single-user workstation;, make sure that the application:

- gains I/O bandwidth by using more than one drive (if applicable). If an application needs to concurrently do I/O on more than one file, try to set things up so that the files are in different file systems, preferably on different drives and ideally on different controllers.
- obtains unfragmented layout of a file. Try to arrange an application so that there is only one file currently being written to the file system where it resides. That is, if you have several files you need to write to a file system, and you have the choice of writing them either one after another or concurrently, you will actually get better space allocation (and consequently better I/O throughput) by writing these files singly, one after another.

If you are on a multi-user server, however, it's hard to control how other applications access the system. Use a large size I/O - 16Kb or more. You may also be able to set up separate file systems for different users. With high sys

time output from *timex*, you need to monitor the operating system to determine why this time is high.

Looking At/Reordering an Application

Many applications have routines that are executed over and over again. You can optimize program performance by modifying these heavily used routines in the source code. The following paragraphs describe the tools that will help tune your programs.

Analyzing Program Behavior with prof

Profiling allows you to monitor program behavior during execution and determine the amount of time spent in each of the routines in the program. There are two types of profiling:

- program counter (PC) sampling
- basic block counting

PC sampling is a statistical method that interrupts the program frequently and records the value of the program counter at each interrupt. Basic block counting, on the other hand, is done by using the *pixie*(1) utility to modify the program module by inserting code at the beginning of each basic block (a sequence of instructions containing no branch instructions) that counts the number of times that each block is entered. Both types of profiling are useful. The primary difference is that basic block counting is deterministic and PC sampling is statistical. To do PC sampling, compile the program with the **-p** option. When the resulting program is executed, it will generate output files with the PC sampling information that can then be analyzed using the *prof*(1) utility.

To do basic block counting, compile the program and then execute *pixie* on it to produce a new binary file that contains the extra instructions to do the counting. When the resulting program is executed, it will produce output files that are then used with *prof* to generate reports of the number of cycles consumed by each basic block. You can then use the output of *prof* to analyze the behavior of the program and optimize the algorithms that consume the majority of the program's time. Refer to the *cc*(1), *f77*(1), *pixie*(1), and *prof*(1) reference pages for more information about the mechanics of profiling.

Reordering a Program with pixie

User program text is demand-loaded a page (currently 4K) at a time. Thus, when a reference is made to an instruction that is not currently in memory and mapped to the user's address space, the encompassing page of instructions is read into memory and then mapped into the user's address space. If often-used subroutines in a loaded program are mixed with seldom-used routines, the program could require more of the system's memory resources than if the routines were loaded in the order of likely use. This is because the seldom-used routines might be brought into memory as part of a page of instructions from another routine.

Tools are available to analyze the execution history of a program and rearrange the program so that the routines are loaded in most-used order (according to the recorded execution history). These tools include *pixie, prof,* and *cc* By using these tools, you can maximize the cache hit ratio (checked by running *sar -b*) or minimize paging (checked by running *sar -p*), and effectively reduce a program's execution time. The following steps illustrate how to reorganize a program named *fetch*

1. Execute the *pixie* command, which will add profiling code to *fetch*:

   ```
   pixie fetch
   ```

   This creates an output file, *fetch.pixie* and a file that contains basic block addresses, *fetch.Addrs*.

2. Run *fetch.pixie* (created in the previous step) on a normal set or sets of data. This creates the file named *fetch.Counts*, which contains the basic block counts.

3. Next, create a feedback file that the compiler will pass to the loader. Do this by executing *prof*:

   ```
   prof -pixie -feedback fbfile fetch fetch.Addrs
   fetch.Counts
   ```

   This produces a feedback file named *fbfile*.

4. Compile the program with the original flags and options, and add the following two options:

   ```
   -feedback fbfile
   ```

For more information, see the *prof* and *pixie* reference pages.

What About Commercial Applications?

You cannot usually tune commercially available applications to any great degree. If your monitoring has told you that a commercially purchased application is causing your system to run at unacceptably slow levels, you have a few options:

- You can look for other areas to reduce system overhead and increase speed, such as reducing the system load in other areas to compensate for your application. Options such as batch processing of files and programs when system load levels permit often show a noticeable increase in performance. See “Automating Tasks with at(1), batch(1), and cron(1M)” on page 29.
- You can use the *nice*(1) and *renice*(1) utilities to change the priority of other processes to give your application a greater share of CPU time. See “Prioritizing Processes with nice” on page 48 and “Changing the Priority of a Running Process” on page 49.
- You can undertake a general program of system performance enhancement, which can include maximizing operating system i/o through disk striping and increased swap space. See “Logical Volumes and Disk Striping” on page 244 and “Swap Space” on page 237.
- You can add additional memory, disk space, or even upgrade to a faster CPU.
- You can find another application that performs the same function but that is less intensive on your system. (This is the least preferable option, of course.)

Monitoring the Operating System

Before you make any changes to your kernel parameters, you should know which parameters should be changed and why. Monitoring the functions of the operating system will help you determine if changing parameters will help your performance, or if new hardware is necessary.

Receiving Kernel Messages and Adjusting Table Sizes

In rare instances a table overflows because it isn't large enough to meet the needs of the system. In this case, an error message appears on the console and in */var/adm/SYSLOG*. If the console window is closed or stored, you'll want to check *SYSLOG* periodically.

Some system calls return an error message that can indicate a number of conditions, one of which is that you need to increase the size of a parameter. Table 5-2 lists the error messages and parameters that may need adjustment. These parameters are in */var/sysgen/master.d/kernel*.

Table 5-2 System Call Errors and Related Parameters

Message	System Call	Parameter
`EAGAIN` `No more processes`	*fork*(2)	increase *nproc* or swap space
`ELIBMAX` `linked more shared libraries than limit`	*exec*(2)	increase *shlbmax*
`E2BIG` `Arg list too long`	*shell*(1), *make*(1), *exec*(2)	increase *ncargs*)

Be aware that there can be other reasons for the errors in the previous table. For example, EAGAIN may appear because of insufficient virtual memory. In this case, you may need to add more swap space. For other conditions that can cause these messages, see the *Owner's Guide* appendix titled "Error Messages".

Other system calls will fail and return error messages that may indicate IPC (interprocess communication) structures need adjustment. These messages and the parameters to adjust are listed in Appendix A, "IRIX Kernel Tunable Parameters."

Using timex(1) and sar(1)

Two utilities you can use to monitor system performance are *timex* and *sar*. They provide very useful information about what's happening in the system.

The operating system has a number of counters that measure internal system activity. Each time an operation is performed, an associated counter is incremented. You can monitor internal system activity by reading the values of these counters.

These utilities monitor the value of the operating system counters, and thus sample system performance. Both utilities use *sadc*, the *sar* data collector, which collects data from the operating system counters and puts it in a file in binary format. The difference is that *timex* takes a sample over a single span of time, while *sar* takes a sample at specified time intervals. The *sar* program also has options which allow sampling of a specific function such as CPU usage (**-u** option) or paging (**-p** option). In addition, the utilities display the data some what differently.

When would you use one utility over the other? If you are running a single application or a couple of programs, use *timex*. If you have a multi-user/multi-processor system, and/or are running many programs, use *sar*.

As in all performance tuning, be sure to run these utilities at the same time you are running an application or a benchmark, and be concerned only when figures are outside the acceptable limits over a period of time.

Using timex

The *timex* utility is a useful troubleshooting tool when you are running a single application. For example:

`timex -s` *application*

The **`-s`** option reports total system activity (not just that due to the application) that occurred during the execution interval of *application*. To redirect *timex* output to a file, (assuming you use the Bourne shell, (*sh*(1)) enter:

`timex -s` *application* **`2> file`**

The same command, entered using the C shell, looks like this:

```
timex -s application > file
```

The *sar* utility is a useful troubleshooting tool when you're running many programs and processes and/or have a multi-user system such as a server. You can take a sample of the operating system counters over a period of time (for a day, a few days, or a week).

Using sar

Depending on your needs, you can choose the way in which you wish to examine system activity. You can monitor the system:

- during daily operation
- consecutively with an interval
- before and after an activity under your control
- during the execution of a command

You can set up the system so *sar* will automatically collect system activity data and put it into files for you. Just use the *chkconfig*(1M) command to turn on *sar*'s automatic reporting feature, which generates a *sar -A* listing. A *crontab* entry instructs the system to sample the system counters every 20 minutes during working hours and every hour at other times for the current day (data is kept for the last 7 days). To enable this feature, type:

```
/etc/chkconfig sar on
```

The data that is collected is put in */var/adm/sa* in the form *sann* and *sarnn*, where *nn* is the date of the report (*sarnn* is in ASCII format). You can use the *sar*(1M) command to output the results of system activity.

Using sar Consecutively with a Time Interval

You can use *sar* to generate consecutive reports about the current state of the system. On the command line, specify a time interval and a count. For example:

```
sar -u 5 8
```

This prints information about CPU use eight times at five-second intervals.

Using sar Before and After a User-Controlled Activity

You may find it useful to take a snapshot of the system activity counters before and after running an application (or after running several applications concurrently). To take a snapshot of system activity, instruct *sadc* (the data collector) to dump its output into a file. Then run the application(s) either under normal system load or restricted load, and when you are ready to stop recording, take another snapshot of system activity. Then compare results to see what happened.

The following is an example of commands that will sample the system counters before and after the application:

```
/usr/lib/sa/sadc 1 1 file
```

Run the application(s) or perform any work you want to monitor, then type:

```
/usr/lib/sa/sadc 1 1 file
sar -f file
```

If *file* does not exist, *sadc* will create it. If it does exist, *sadc* will append data to it.

Using sar and timex During the Execution of a Command

Often you want to examine system activity during the execution of a command or set of commands. The aforementioned method will allow you do to this. For example, to examine all system activity while running *nroff*(1), type:

```
/usr/lib/sa/sadc 1 1 sa.out
nroff -mm file.mm > file.out
/usr/lib/sa/sadc 1 1 sa.out
sar -A -f sa.out
```

By using *timex*, you can do the same thing with one line of code:

```
timex -s nroff -mm file.mm > file.out
```

Note that the *timex* also includes the real, user, and system time spent executing the *nroff* request.

There are two minor differences between *timex* and *sar*. The *sar* program has the ability to limit its output (e.g., the **-u** option reports only CPU activity),

while *timex* always prints the **-A** listing. Also, *sar* works a variety of ways, as discussed previously, but *timex* only works by executing a command. However, this command can be a shell file.

If you are interested in system activity during the execution of two or more commands running concurrently, put the commands into a shell file and run *timex -s* on the file. For example, suppose the file *nroff.sh* contained the following lines:

```
nroff -mm file1.mm > file1.out &
nroff -mm file2.mm > file2.out &
wait
```

To get a report of all system activity after both of the *nroff* requests (running concurrently) finish, invoke *timex* as follows:

```
timex -s nroff.sh
```

Now that you have learned when and how to use *sar* and *timex*, you can choose one of these utilities to monitor the operating system. Then examine the output and try to determine what's causing performance degradation. Look for numbers that show large fluctuation or change over a sustained period; don't be too concerned if numbers occasionally go beyond the maximum.

The first thing to check is how the system is handling the disk I/O process. After that, check for excessive paging/swapping. Finally look at CPU use and memory allocation.

The sections immediately following assume that the system you are tuning is active (with applications/benchmark executing).

Checking Disk I/O

The system uses disks to store data and transfers data between the disk and memory. This input/output (I/O) process consumes a lot of system resources, so you want the operating system to be as efficient as possible when it performs I/O.

If you are going to run a large application or have a heavy system load, the system will benefit from disk I/O tuning. Run *sar -A* or *timex -s* and look at

the **%busy**, **%rcache**, **%wcache**, and **%wio** fields.To see if your disk subsystem needs tuning, check your output of *sar -A* against the figures in the following table. (Note that in the tables that follow, the right column lists the *sar* option that will print only selected output, for example output for disk usage (*sar -d*) or CPU activity (*sar -u*).

Table 5-3 lists *sar* results that indicate an I/O-bound system.

Table 5-3 Indications of an I/O-Bound System

Field	Value	sar Option
%busy (% time disk is busy)	>85%	*sar -d*
%rcache (reads in buffer cache)	low, <85	*sar -b*
%wcache (writes in buffer cache)	low, <60%	*sar -b*
%wio (idle CPU waiting for disk I/O)	dev. system >30 fileserver >80	*sar -u*

Notice that for the %wio figures (indicates the percentage of time the CPU is idle while waiting for disk I/O), there are examples of two types of systems:

- a development system that has users who are running programs such as *make*. In this case, if %wio > 30, check the breakdown of %wio (*sar -u*). By looking at the %wfs (waiting for file system) and %wswp (waiting for swap), you can pinpoint exactly what the system is waiting for.
- an NFS system that is serving NFS clients and is running as a file server. In this case, if %wio > 80, %wfs > 90, the system is disk I/O bound.

There are many other factors to consider when you tune for maximum I/O performance. You may also be able to increase performance by:

- using logical volumes
- using partitions on different disks
- adding hardware (a disk, controller, memory)

Using Logical Volumes to Improve Disk I/O

By using logical volumes, you can:

- grow an existing file system to a larger size without having to disturb the existing file system contents.
- stripe file systems across multiple disks—You may be able to obtain up to 50% improvement in your I/O throughput by creating striped volumes on different disks.

  Striping works best on disks that are on different controllers. Logical volumes give you more space without remaking the first filesystem. Disk striping gives you more space with increased performance potential, but you do run the risk that if you lose one of the disks with striped data, you will lose all the data on the filesystem since the data is interspersed across all the disks.

  Contiguous logical volumes fill up one disk, and then write to the next. Striped logical volumes write to both disks equally, spreading each file across all disks in the volume, so it is impossible to recover from a bad disk if the data is striped, but it is possible if the data is in a contiguous logical volume. For information on creating a striped disk volume, see "Logical Volumes and Disk Striping" on page 244.

Using Partitions and Additional Disks to Improve Disk I/O

There are some obvious things you can do to increase your system's throughput, such as limiting the number of programs that can run at peak times, shifting processes to non-peak hours (run batch jobs at night), and shifting processes to another machine. You can also set up partitions on separate disks to redistribute the disk load.

Before continuing with the discussion about partitions, let's look at how a program uses a disk as it executes. Table 5-4 shows various reasons why an application may need to access the disk.

Table 5-4 An Application's Disk Access

Application	Disk Access
execute object code	text and data
uses swap space for data, stack	*/dev/swap*

Table 5-4 (continued) An Application's Disk Access

Application	Disk Access
writes temporary files	*/tmp* and */var/tmp*
reads/writes data files	data files

You can maximize I/O performance by using separate partitions on different disks for some of the aforementioned disk access areas. In effect, you are spreading out the application's disk access routines, which will speed up I/O.

By default, disks are partitioned to allow access in two ways, either:

- three partitions: partitions 0, 1 and 6, or
- one large partition, partition 7 (encompasses the three smaller partitions)

On the system disk, partition 0 is for root, 1 is for swap, and 6 is for */usr*.

For each additional disk, you need to decide if you want a number of partitions or one large one and what file systems (or swap) you want on each disk and partition. It's best to distribute file systems in the disk partitions so that different disks are being accessed concurrently.

The configuration depends on how you use the system, so it helps to look at a few examples.

- Consider a system that typically runs a single graphics application that often reads from a data file. The application is so large that its pages are often swapped out to the swap partition.

  In this case, it might make sense to have the application's data file on a disk separate from the swap area.
- If after configuring the system this way, you find that it doesn't have enough swap space, consider either obtaining more memory, or backing up everything on the second hard disk and creating partitions to contain both a swap area and a data area.
- Changing the size of a partition containing an existing file system may make any data in that file system inaccessible. Always do a complete and current backup (with verification) and document partition

information before making a change. If you change the wrong partition, you can change it back, providing you do not run *mkfs* on it or overwrite it. It is recommended that you print a copy of the *prtvtoc* command output after you have customized your disks, so that they may be more easily restored in the event of severe disk damage.

Also, if you have a very large application and have three disks, consider using partitions on the second and third disks for the application's executables (*/bin* and */usr/bin*) and for data files, respectively. Next, consider a system that mostly runs as a "compile-engine."

In this case, it might be best to place the */tmp* directory on a disk separate from the source code being compiled. Make sure that you check and mount the file system prior to creating any files on it. (If this is not feasible, you can instruct the compiler to use a directory on a different disk for temporary files. Just set the TMPDIR environment variable to the new directory for temporary files.) Now, look at a system that mainly runs many programs at the same time and does a lot of swapping.

In this case, it might be best to distribute the swap area in several partitions on different disks.

Adding Disk Hardware to Improve Disk I/O

If improved I/O performance still does not occur after you have tuned as described previously, you may want to consider adding more hardware: disks, controllers, or memory.

If you are going to add more hardware to your system, how do you know which disk/controller to add? You can compare hardware specifications for currently supported disks and controllers by turning to your hardware *Owner's Guide* and looking up the system specifications. By using this information, you can choose the right disk/controller to suit your particular needs.

By balancing the most active file systems across controllers/disks, you can speed up disk access.

Another way to reduce the number of reads and writes that go out to the disk is to add more memory. This will reduce swapping and paging.

Checking for Excessive Paging and Swapping

The CPU can only reference data and execute code that are loaded into memory. Because the CPU executes multiple processes, there may not be enough memory for all the processes. If you have very large programs, they may require more memory than is physically present in the system. So processes are brought into memory in pages; if there's not enough memory, the operating system frees memory by writing pages temporarily to a secondary memory area, the swap area.

Table 5-5 shows indications of excessive paging and swapping on a smaller system. Large servers will show much higher numbers.

Table 5-5 Indications of Excessive Swapping/Paging

Field	Value	sar Option
bswot/s (transfers from memory to disk swap area)	>200	*sar -w*
bswin/s (transfers to memory)	>200	*sar -w*
%swpocc (time swap queue is occupied)	>10	*sar -q*
rflt/s (page reference fault)	>0	*sar -t*
freemem (average pages for user processes)	<100	*sar -r*

There are several things you can do to reduce excessive paging/swapping:

- limit the number of programs that run at peak times and shift processes to non-peak hours (run batch jobs at night - see *at*(1)) or to another machine.
- run multiple jobs in sequence, not in parallel.
- if you increased various parameters (for example, *nproc*), decrease them again.
- reduce page faults. Construct programs with "locality" in mind (see "Tuning an Application" on page 145).
- use shared libraries.

- reduce resident set size limits with *systune*. See “System Limits Tunable Parameters” on page 763 for the names and characteristics of the appropriate parameters.
- add more memory.

You can also improve performance by adding swap partitions to spread swap activity across several disks or adding swap files on several partitions. For more information on swapping to files, see “Swap Space” on page 237.

Why does a system sometimes page excessively? The amount of memory any one program needs during any given instant (the working set) should fit within physical memory. If over a 5 to 10 second period of time, routines access more pages than can fit in physical memory, there will be excessive thrashing and paging. For example, a SCSI drive can read at a rate of 1.5 Mb (384 pages) per second. Remembering all the overhead plus other processes that are contending for memory, 50-100 pages per second is a reasonable number of pages to bring into memory. However, 100 pages per second over a sustained period will result in poor performance.

In summary, there will be excessive paging and thrashing if, (1) the number of pages brought into physical memory is over about 100 pages per second for a sustained period of time, and (2) the working set size of all processes is larger than physical memory.

Checking CPU Activity and Memory Allocation

After looking at disk I/O and paging, next check CPU activity and memory allocation.

Checking The CPU

A CPU can execute only one process at any given instant. If the CPU becomes overloaded, processes have to wait instead of executing. You can’t change the speed of the CPU (although you may be able to upgrade to a faster CPU or add CPU boards to your system if your hardware allows it),

but you can monitor CPU load and try to distribute the load. Table 5-6 shows the fields to check for indications that a system is CPU bound.

Table 5-6 Indications of a CPU-Bound System

Field	Value	sar Option
%idle (% of time CPU has no work to do)	<5	sar -u
runq-sz (processes in memory waiting for CPU)	>2	sar -q
%runocc (% run queue occupied and processes not executing)	>90	sar -q

You can also use the *top*(1) or *gr_top*(1) commands to display processes having the highest CPU usage. For each process, the output lists the user, process state flags, process ID and group ID, CPU cycles used, processor currently executing the process, process priority, process size (in pages), resident set size (in pages), amount of time used by the process, and the process name. For more information, see the *top*(1) or *gr_top*(1) reference pages.

To increase CPU performance, you'll want to make the following modifications:

- off-load jobs to non-peak times or to another machine, set efficient paths, and tune applications.
- eliminate polling loops (see *select*(2)).
- increase the *slice-size* parameter (the length of a process time slice). For example, change *slice-size* from Hz/30 to Hz/10. However, be aware that this may slow interactive response time.
- upgrade to a faster CPU or add another CPU.

Checking Available Memory

"Checking for Excessive Paging and Swapping" on page 159 described what happens when you don't have enough physical (main) memory for processes.

This section discusses a different problem - what happens when you don't have enough available memory (sometimes called virtual memory), which includes both physical memory and logical swap space.

The IRIX virtual memory subsystem allows programs that are larger than physical memory to execute successfully. It also allows several programs to run even if the combined memory needs of the programs exceed physical memory. It does this by storing the excess data on the swap device(s).

The allocation of swap space is done after program execution has begun. This allows programs with large a virtual address to run as long as the actual amount of virtual memory allocated does not exceed the memory and swap resources of the machine.

Usually it's very evident when you run out of memory, because a message is sent to the console that begins:

Out of logical swap space...

If you see this message:

- the process has exceeded ENOMEM or UMEM.
- there is not enough physical memory for the kernel to hold the required non-pageable data structures.
- there is not enough logical swap space.

You can add virtual swap space to your system at any time. See "Swap Space" on page 237 if you wish to add more swap. You need to add physical swap space, though, if you see the message:

```
Process killed due to insufficient memory
```

The following system calls will return EAGAIN if there is insufficient available memory: *exec*, *fork*, *brk*, *sbrk* (called by *malloc*), *mpin*, and *plock*. Applications should check the return status and exit gracefully with a useful message.

To check the size (in pages) of a process that is running, execute *ps -el* (you can also use *top*(1)). The SZ:RSS field will show you very large processes.

By checking this field, you can determine the amount of memory the process is using. A good strategy is to run very large processes at less busy times.

Determining the Amount of System Memory

To see the amount of main memory, use the *hinv*(1) command. It displays data about your system's configuration. For example:

```
Main memory size: 64 Mb
```

Maximizing Memory

To increase the amount of virtual memory, increase the amount of real memory and/or swap space. Note that most of the paging/swapping solutions also apply to ways to conserve available memory. These include:

- limiting the number of programs
- using shared libraries
- adding more memory
- decreasing the size of system tables

However, the most dramatic way to increase the amount of virtual memory is to add more swap space. The previous section that described using partitions explained how to do this and it is covered completely in "Swap Space" on page 237.

Tuning The Operating System

The process of tuning your operating system is not difficult, but it should be approached carefully. Make complete notes of your actions in case you need to reverse your changes later on. Understand what you are going to do before you do it, and do not expect miraculous results; IRIX has been engineered to provide the best possible performance under all but the most extreme conditions. Software that provides a great deal of graphics manipulation or data manipulation also carries a great deal of overhead for the system, and can seriously affect the speed of an otherwise robust system. No amount of tuning will change these situations.

Operating System Tuning Steps

To tune a system, you first monitor its performance with various system utilities as described in "Monitoring the Operating System" on page 149. This section describes the steps to take when you are tuning a system.

1. Determine the general area that needs tuning (for example, disk I/O or the CPU) and monitor system performance using utilities like *sar*(1) and *osview*(1M). If you have not already done so, see "Monitoring the Operating System" on page 149.
2. Pinpoint a specific area and monitor performance over a period of time. Look for numbers that show large fluctuation or change over a sustained period; don't be too concerned if numbers occasionally go beyond the maximum.
3. Modify one value/characteristic at a time (for example, change a parameter, add a controller) to determine its effect. It's good practice to document any changes in a system notebook.
4. Use the *systune*(1M) command to change parameter values.
5. Remeasure performance and compare the before and after results. Then evaluate the results (is system performance better?) and determine if further change is needed.

Keep in mind that the tuning procedure is more an *art* than a science; you may need to repeat the above steps as necessary to fine tune your system. You may find that you'll need to do more extensive monitoring and testing to thoroughly fine tune your system.

Finding Current Values of Parameters

Before you can tune your system, you need to know the current values of the tunable parameters. To find the current value of your kernel parameters, use the *systune*(1M) command. This command, entered with no arguments, prints the current values of all tunable parameters on your system. For complete information on this command, see the *systune*(1M) reference page.

Changing Parameters and Reconfiguring the System

After determining the parameter or parameters to adjust, you must change the parameters and you may need to reconfigure the system for the changes to take effect. The *systune*(1M) utility will tell you when you make parameter changes if you must reboot to activate those changes. There are several steps to reconfiguration procedure:

- back up the system
- copy your existing kernel to *unix.save*
- make your changes
- reboot your system, if necessary

Backing Up the System

Before you reconfigure the system by changing kernel parameters, it's a good idea to have a current and complete backup of the system. See Chapter 6, "Backing Up and Restoring Files," for back-up procedures.

Caution: Always back up the entire system before tuning.

Changing a Parameter

After determining the parameter you need to change (for example, you need to increase *nproc* because you have a large number of users) you must first back up the system and the kernel. Give the command:

```
cp /unix /unix.save
```

This command creates a copy of your kernel. Through the rest of this example, this is called your old saved kernel. If you make this copy, you can always go back to your original kernel if you are not satisfied with the results of your tuning.

Once your backups are complete, you can execute the *systune*(1M) command. An invocation of *systune*(1M) to increase *nproc* looks something like this:

```
systune -i
Updates will be made to running system and /unix.install
```

```
systune-> nproc
        nproc = 400 (0x190)
systune-> nproc = 500
        nproc = 400 (0x190)
        Do you really want to change nproc to 500 (0x1f4)?
(y/n) y
In order for the change in parameter nproc to become
effective /unix.install must be moved to /unix and the
system rebooted
systune-> quit
```

To boot the kernel you just made, give the command:

```
mv /unix.install /unix
```

Then reboot your system. Also, be sure to document the parameter change you made in your system log book.

Creating and Booting a New Kernel

The *systune* command creates a new kernel automatically. However, if you changed parameters without using systune, or if you have added new system hardware (such as a new CPU board on a multiprocessor system), you will need to use *autoconfig* to generate a new kernel. To build a new kernel after reconfiguring the system, follow these steps:

1. Become the Superuser by giving the command:

   ```
   su
   ```

2. Give the command:

   ```
   /etc/autoconfig -f
   ```

   This command creates a new kernel and places it in the file */unix.install*.

3. Make a copy of your current kernel with the command:

   ```
   cp /unix /unix.save
   ```

4. Reboot your system with the command:

   ```
   reboot
   ```

Caution: When you issue the *reboot* command, the system overwrites the current kernel (*/unix*) with the kernel you have just created (*/unix.install*). This is why you should always copy the current kernel to a safe place before rebooting.

An autoconfiguration script, found in */etc/rc2.d/S95autoconfig*, runs during system start-up. This script asks you if you would like to build a new kernel under the following conditions:

- A new board has been installed for which no driver exists in the current kernel.
- There have been changes to object files in */var/sysgen/mtune*, master files in */var/sysgen/master.d*, or the system files in */var/sysgen/system*. This is determined by the modification dates on these files and the kernel.

If any of these conditions is true, the system prompts you during startup to reconfigure the operating system:

Automatically reconfigure the operating system? **y**

If you answer **y** to the prompt, the script runs *lboot* and generates */unix.install* with the new image.You can disable the autoconfiguration script by renaming */etc/rc2.d/S95autoconfig* to something else that does not begin with the letter S, for example, */etc/rc2.d/wasS95autoconfig*.

Recovering from an Unbootable Kernel

The following procedure explains how to recover from an unbootable */unix*, and describes how to get a viable version of the software running after an unsuccessful reconfiguration attempt. If you use the *systune*(1M) utility, you should never have to use this information, since *systune* will not allow you to set your parameters to unworkable values.

1. If the system fails to reboot, try to reboot it a few more times. If it still fails, you need to interrupt the boot process and direct the boot PROM to boot from your old saved kernel (*unix.save*).
2. Press the **reset** button.You will see the System Maintenance Menu:

```
System Maintenance Menu
1) Start System.
2) Install System Software.
3) Run Diagnostics.
```

```
4) Recover System.
5) Enter Command Monitor.
```

3. Select option 5 to enter the Command Monitor. You see:

   ```
   Command Monitor. Type "exit" to return to the menu.
   >>
   ```

4. Now at the >> prompt, tell the PROM to boot your old saved kernel. The command is:

   `boot unix.save`

   The system will then boot the old saved kernel.

5. Once the system is running, use the following command to move your old saved kernel to the default */unix* name. This method also keeps a copy of your old saved kernel in *unix.save*:

   `cp /unix.save /unix`

Then you can normally boot the system while you investigate the problem with the new kernel. Try to figure out what went wrong. What was changed that stopped the kernel from booting? Review the changes that you made.

- Did you increase/decrease a parameter by a huge amount? If so, make the change less drastic.
- Did you change more than one parameter? If so, make a change to only one parameter at a time.

Chapter 6

Backing Up and Restoring Data

Chapter 6 describes the tasks you will perform to keep your system backed up in case of data loss. This is one of the most important chapters in the manual, as your system integrity depends on thorough backups. Tasks covered here include:

- *Backup up single files.*
- *Backing up filesystems.*
- *Backing up the whole system.*
- *Restoring files from backups.*
- *Creating and using a backup schedule.*

Chapter 6

Backing Up and Restoring Files

As a site administrator, you must make sure there are backups of the files at your site. Users depend on you to recover files that have been accidentally erased, or lost due to hardware problems.

This chapter describes how to create backups of system files. In particular, it discusses:

- The available backup utilities. See "Choosing a Backup Tool" on page 172.
- How to make a backup copy of a single file, or a small number of files. See "Making Backups" on page 177.
- How to back up a directory or series of directories. See "Backing Up File Systems" on page 177.
- How to back up entire file systems. See "Backing Up File Systems" on page 177.
- How to check your backups to make sure they are valid. See "Checking an Archive" on page 188.
- How to restore file systems and individual files and directories. See "Restoring Files and File Systems" on page 190.
- Recovering from a system crash. See "Recovery after System Corruption" on page 197.
- How to make a bootable backup tape. See "Copying the Software Distribution" on page 201.
- Backup strategies—how to keep important data from being lost. See "Backup Strategies" on page 203.

IRIX provides several different utilities for backing up your data:

- The System Manager
- *bru*(1M)

- *Backup*(1M) and *Restore*(1M), which use *bru*
- *tar*(1M)
- *cpio*(1M)
- *dump*(1M) and *restore*(1M)
- *dd*(1M)

The sections in this chapter present the most common tasks a site administrator must perform. Each section describes how to perform a task using one or more of the commands listed above.

For a complete list of options for a particular command, see the reference page for that command.

Types of Backup Media

Throughout this chapter we refer to *tapes*, *tape drives*, *media*, and *backup media*. These terms apply to whichever backup media you use at your site.

Common types of media include:

- 1/4" cartridge tape, 4-track
- 1/2" magnetic tape, reel, 9-track
- 8 mm cartridge
- DAT

It should be apparent when certain limitations or conditions do not apply to your specific media. For example, if you back up a 350 MB file system with an 8 mm cartridge drive (which can hold up to 1.2 GB), you probably don't have to worry about using more than one tape. For more information on tape capacities, see "Tape Capacities" on page 265.

Choosing a Backup Tool

The IRIX system provides different back-up tools, both file system-oriented and file and directory-oriented utilities. Differences between the two kinds

of tools are discussed in the next section. The most convenient tool to use is the System Manager, described in detail in the *Personal System Administration Guide*.

Use whichever programs you like. If many users at your site are already familiar with one backup program, you may wish to use that program consistently. If there are workstations at your site from other manufacturers, you may wish to use a back-up utility that is common to all the workstations.

Types of Backup Tools

Two basic types of backup tools are available with IRIX:

- File system-oriented programs, like *bru*, *Backup*, and *dump*; these are designed primarily to back up entire file systems, although *bru* can also back up individual files and directories.
- File- and directory-oriented programs, like *tar* and *cpio*; these can back up entire file systems, but are also convenient for individual files and directories.

In addition, you can use *dd*(1) to read images exactly as they are written, with or without conversions. The *dd* command is useful to read data that is written in a format incompatible with the other backup utilities. However, you do not normally use *dd* to create backups.

The System Manager

The System Manager is the default tool for performing backups on your system. See the *Personal System Administration Guide* for a complete description of the System Manager and its backup and recovery facilities.

bru

The *bru* utility is a flexible tool that contains such features as:

- file compression
- the ability to locate and back up files based on modification date

- the ability to calculate space requirements for backups
- the ability to check the integrity of the data on the tape
- varying levels of "verbosity" while performing operations

Because it not only backs up file systems but also individual files and directories, it combines the features of both file system-oriented and file-oriented backup utilities. See the *bru*(1M) reference page for a complete description of the program's capabilities.

Be aware that although *bru* is available on a variety of UNIX systems, it is not as widely used as the other backup utilities. At a site that has workstations from a variety of vendors, not all of which provide *bru*, you may wish to use one of the other IRIX backup utilities for consistency. *tar*(1M) is the most widely accepted format.

If your site has predominately IRIS workstations, and you don't need to move file system backups between different brands of computers, *bru* is probably a good choice.

Note that the System Manager backup menu uses the *Backup* interface to *bru* to back up workstations.

If a *bru* backup is made that requires more than one tape, *bru* stops and prompts you to insert another tape before it continues. Be aware that if *bru* (or any front end interface to *bru*) is used in a noninteractive mode (such as through a *cron* command) and the information fills a tape, *bru* cannot prompt for a new tape, and automatically overwrites the information it has just written on the tape. Obviously, this is not a good backup practice, and backups of large systems made with *bru* should be performed interactively.

Backup and *Restore*

The *Backup* and *Restore* utilities are front end interfaces to *bru*. They support the remote host name and tape device options, and *Backup* creates a volume header file listing that *Restore* uses for recovering the files and directories. For complete information, consult the *Backup*(1) and *Restore* reference pages. If you are planning to use the IRIS System Maintenance Menu System Recovery option, you should use *Backup* or the backup facility of the graphical System Manager, as those are the only formats accepted by the

System Maintenance Menu. The System Manager is described in detail in the *Personal System Administration Guide*.

dump and *restore*

The *dump* and *restore* programs are standard file system backup utilities used on many UNIX systems. The *dump* program makes incremental backups of entire file systems.

Use *restore* to retrieve files from a *dump* archive. With *restore*, you can restore an entire file system or specific files. It also has an interactive mode that lets you browse the contents of an archive, select specific files, and restore them.

tar

The *tar* utility backs up files and directories. You can copy files to tape, create *tar* files, compare files on tape to files on disk, read standard input, and pipe the output of *tar* to other processes. This command is widely used on UNIX systems worldwide.

cpio

Like *tar*, *cpio* archives files and directories. With *cpio*, you can copy files to tape or disk, archive empty directories, swap byte order, create portable ASCII archives, and read from and write to standard output. *cpio* is also useful for copying files and directories when the *cp*(1) command is unable to do so. For example, you cannot use *cp* to copy a directory to a different file system.

dd

The *dd* program reads from a specified input file (*stdin* is the default), performs whatever conversions you specify, and writes the result to a specified output file. (*stdout* is the default.) It is not specifically a backup tool, but has many extremely useful features, including the ability to:

- skip specific blocks in an archive

- skip blocks of output
- change input and output block size
- copy a specific number of blocks
- perform various data conversions such as byte swapping

Backup Procedure

Follow these steps when making a backup, no matter which backup utility you use:

1. Make sure the tape drive is clean. The hardware manual that came with your drive should state how, and how often, to clean the drive.

   Dirty tape heads can cause read and write errors. New tapes shed more oxide than older tapes, so you should clean your drive more frequently if you use a lot of new tapes.

2. Make sure you have enough backup media on hand. The *bru* utility has an option to check the size of a backup, so you can determine if you have enough media. You can also use such utilities as *du*(1) and *df*(1) to determine the size of directories and file systems, respectively.

3. Use good-quality media. Considering the value of your data, you should use the best quality media you can afford.

4. If you are backing up an entire file system, for example with *bru* or *dump*, run *fsck*(1M) on the file system to make sure you do not create a tape of a damaged file system. You must unmount a file system before checking it with *fsck*, so plan your backup schedule accordingly.

   This step is not necessary if you are backing up only a few files (for example, with *tar*).

5. The default tape device for any drives you may have is */dev/tape*. If you do not wish to use the default device, you must specify a device in your backup command line.

6. Label your backups. If you plan to reuse the media, use pencil. Include the date, time, name of the machine, the name of the utility. The exact command line used to make the backup (so you'll remember how to

extract the files later), and a general indication of the contents. If more than one administrator performs backups at your site, include your name.

7. Verify the backup when you are finished. Some utilities (such as *dump* and *bru*) provide explicit options to verify a backup. With other programs, you can simply list the contents of the archive—this is usually sufficient to catch errors in the backup.
8. Write-protect your media after you make the backup.
9. Note the number of times you use each tape. It's sufficient to keep a running tally on the tape label.

See "Storing Backups" on page 209 for information on safely storing your backups.

Making Backups

The following sections describe different ways to back up data.

Backing Up File Systems

The following sections describe how to back up file systems. Although this information is oriented toward file systems, you can also recover individual files from a file system backup.

A file system backup is most useful for:

- quickly reconstructing a workstation after a hardware failure
- backing up user files on a large multiuser server where the administrator may not be aware of all the subtleties of the users' data

A file system backup made from the System Manager is useful in reconstructing a workstation after a disk failure because it contains the various device nodes and symbolic links used by your system.

Saving a File System with the System Manager

To make a backup of your system with the System Manager on any system with a graphical user interface, bring up the System Manager and select the Backup and Restore icon. From this window, follow the prompts to perform your backup. A complete set of instructions for this procedure is available in the *Personal System Administration Guide*.

Backups made with the System Manager are the easiest to make and use, and are accessible from the System Maintenance Menu if they are full system backups. When you make a full system backup, the command also makes a backup of the files in the disk volume header and saves the information in a file that is stored on tape. This file is used during system recovery to restore a damaged volume header.

Saving a File System with *bru*

The *bru* command is the shell command used by the System Manager to create backups. If you are using a server and do not have access to the graphical System Manager, use *bru* instead. Backups made with *bru* are readable by the System Maintenance Menu and Command Monitor. This command backs up the */usr* file system:

```
bru -c /usr
```

Saving a File System with *Backup*

To back up a file system with *Backup*, place a tape or other media in the drive. Make sure the tape is locked in the drive. Enter:

```
Backup /
```

The *Backup* command archives the entire system. You can make a specific file system backup of */usr* by specifying */usr* as the directory name. The current date is saved in the file */etc/lastbackup*.

Note: In order to use a *Backup* tape to restore your system from the System Maintenance Menu, you must make a full system backup. When you make a full system backup, the command also makes a backup of the files in the disk volume header and saves the information in a file that is stored on tape. This file is used during system recovery to restore a damaged volume header.

To use a remote tape drive, use the **-h** option:

```
Backup -h guest@alice.cbs.tv.com:/dev/tape /usr/people/ralph
```

You must have at least *guest* login privileges on the remote system in order to use a remote tape drive. The file */tmp/volhdrlist* contains a list of the root volume header.

Saving a File System with *dump*

The *dump* utility archives not only regular files, but also device files and special files such as links and named pipes. To recover files from an archive, you use the *restore* command.The date on which you last ran the *dump* program is stored in the file */etc/dumpdates* when you specify the **u** key to indicate an update.

This command backs up all files on the */usr* file system:

```
dump 0 /dev/usr
```

The *dump* program is described in further detail in “Incremental Backups with dump” on page 207.

Saving a File System with *dd*

This command copies an image of a file system to the default tape device:

```
dd if=/dev/usr of=/dev/tape
```

Saving a File System with *cpio*

To back up a single file system (such as */usr*) use the following series of commands:

```
cd /usr
find . -mount -type f -print | cpio -ovBc > /dev/tape
```

You may wish to prevent certain directories, such as */usr/tmp*, from being backed up. In this case, you could use a command such as this one (type the command on one line):

```
find . -mount -type f -print |
grep -v '^./usr/tmp/' | cpio -ovBc > /dev/tape
```

This example uses *grep* to remove all occurrences of *./usr/tmp/* at the beginning of the line. Put additional *grep -v* commands in the pipeline to remove other unwanted files or directories. Because we use the **-mount** argument on the *find* command, we don't have to worry about data that is not mounted on */usr*.

For more information on this topic, see the reference pages *cpio*(1), *find*(1), and *grep*(1).

Saving a File System with *tar*

To back up the */usr* file system with *tar*, use the command series:

```
cd /usr
tar cv .
```

Backing Up Individual Files

The following sections provide instructions on backing up files and directories with the various utilities available to you. As always, it is recommended that you perform your backups using the graphical System Manager or with *bru*. If your backup tape is to be read by a different manufacturer's system, however, it is usually best to find out in advance what formats the receiving system is able to use. *tar* and *cpio* are the most commonly accepted formats worldwide.

Backing Up Files with *bru*

To back up individual files with *bru*, enter:

```
bru -c files
```

You can specify one or more files. You can also read file names from another file:

```
bru -c 'cat listfile'
```

Backing Up Files with *tar*

To back up individual files with *tar*, use the command:

```
tar c files
```

Backing Up Files with *cpio*

To back up files with *cpio*, use the command:

cat *filelist* **| cpio -o > /dev/tape**

Saving Files by Modification Date

To save specific files that have changed since a particular time, you can use *bru* with the **-n** option. The following command backs up files on the */usr* file system that have been modified since November 26, 1990:

```
bru -c -n 26-Nov-1990 /usr
```

Neither *tar* nor *cpio* has this capability built in. However, you can use the *find* command to archive files that have not been modified in a particular number of days. With *tar*:

tar cv 'find /usr -mtime 5 -type f -o -type *othertypes* **-print'**

The left quote marks cause the shell to execute the *find* command, then send its output to the *tar* command. The find command locates regular files that have not been modified in five days.

With *cpio*:

```
find /usr -mtime 5 -depth -print | cpio -oO /dev/tape
```

The **-depth** argument causes *find* to print the name of the directory after printing the files in that directory. This ensures that *cpio* has permission to place the files in the directory in case the directory is read-only.

Listing Files on an Archive

The ''table of contents'' flag, **-t**, displays the contents of a *bru* archive:

```
bru -t
```

You can combine this with the **-v** option for more information:

```
bru -tv
```

Use up to four "**v**" arguments for the most verbose output possible.

For *tar* archives, use the **v** keyword for verbose listing of the archive contents:

```
tar tv
```

For *cpio* archives, use the following command to obtain a verbose listing:

```
cpio -itvI /dev/tape
```

Estimating Space Required for Backup

Use the **-e** option with *bru* for an estimate of how much space is required for an archive:

```
bru -e /usr
```

Saving Files Using Data Compression

You can compress files as are they are archived. Use the **-Z** flag:

```
bru -Z
```

bru uses a 12-bit LZW file compression algorithm. Note that not all versions of *bru* support LZW compression. If you plan to transfer a *bru* archive to another vendor's workstation, make sure the other version of *bru* supports LZW data compression.

If you add the **-v** option, *bru* displays the compression ratio for each file (as a percentage). If you use **-t** and **-Z** to display the table of contents of an archive that contains compressed files, *bru* displays the current file names and compressed sizes, instead of the original file names and sizes before archiving.

Placing Multiple Backups on a Single Tape

The standard or normal default tape device, */dev/tape*, will rewind the tape after executing a command. To add additional tape files to the tape, you must use a tape device that will skip forward on the tape and then stop there without rewinding the tape. This is called the no-rewind tape device.

The default no-rewind tape device is */dev/nrtape*.

It is assumed that the default tape drive is being used for this operation.

Caution: If you are not careful when appending files to a tape, you can overwrite the existing files and these files will no longer be accessible. It is recommended that you try this procedure on a scratch tape before trying this on an important tape.

When you use *tar* to first create a tape archive, all the files written to the tape are put into a single tape file. At the end of this tape file is an end of file marker (EOF) and if this is the only tape file, an end of tape (EOT) marker. If additional files are added to this the tape, they will be placed into a second tape file after the first tape file's EOF marker and the final tape file will be followed by an EOT marker. For a more detailed explanation of this see the reference page for *tps* (7M). To add an additional tape file and subsequently read the tape file, you must skip past the first tape file.

Writing Additional Files to the Tape

There are two ways to write an additional tape file depending on how the previous tape file was created. These examples will assume that the previous tape file was created using *tar* unless otherwise specified.

For the first example, the previous tape file was created using the command:

```
tar cvf /dev/tape [files]
```

where `[files]` is a list of filenames on the tape.

This command writes the files onto the tape into a single tape file. When all the files are written, this command will cause the tape to rewind.

Now you want to write a second set of files onto the tape and so you must skip past the first tape file. To do this you need to use the *mt* command and skip to the end of tape marker (*feom*). This time you will specify the no-rewind tape device so that when the command is finished, it stops there and doesn't rewind back to the beginning of the tape. Once this is done, you can write your second set of files onto the tape.

```
mt -t /dev/nrtape feom
tar cvf /dev/tape [files]
```

This process can be repeated for as many tape files as you wish, constrained only by the capacity of your tape.

Another way of appending tape files is to use the no-rewind device when creating the tape files. In this way, the tape is never rewound.

Note: This process will only work if the tape is never removed from the tape drive and for each *tar* command issued, the no-rewind device is specified.

```
tar cvf /dev/nrtape [files]
```

This process can be repeated for as many tape files as you wish.

If at any time you are unsure of whether or not the tape has been rewound, follow the instruction in the first example shown above. If the tape has been rewound and you continue with these steps you will destroy the tar files written previously.

Reading Multiple Tape Files

Since there are now multiple tape files on the tape, you must skip forward to find the specific tape file from which you wish to read or extract files. To do this, you must use the *mt* command and specify how many tape files you want to skip forward (fsf #). Two examples of doing this are shown below.

For the first example, it is assumed that you know which tape file contains the file *john_file*.

You have 4 tape files on a tape and you want to read the third tape file. This means you want to skip past the first and second tape files and stop the tape there so that you can read the third tape file. To do this you need to use the no-rewind device with the following commands:

```
mt -t /dev/tape rewind
mt -t /dev/nrtape fsf 2
tar xvf /dev/tape john_file
```

Note, first you use the *mt* command to rewind the tape to the beginning to verify the starting point of the tape. The second time the *mt* command is used with the no-rewind device in order to skip past the first two tape files. The regular tape device is used with the *tar* command. Once the table of contents of this third tape file is listed, the tape will rewind to the beginning of the tape.

For the second example, you have a tape that has many tape files on it and you want to extract a specific file named *john_file*, but you don't know which tape file contains this file.

You could use the approach discussed above and continue to increase the number of file to skip past and then list the table of contents for the tape file. However, in this case, it would be easier to use the no-rewind device each time you list the table of contents. After each listing of the table of contents check to see if *john_file* is in that tape file. If yes, then note which tape file the file is in and use the *mt* and *tar* commands to extract the file. If no, read the next tape file with the *tar* command.

```
mt -t /dev/tape rewind
tar tvf /dev/nrtape
tar tvf /dev/nrtape
tar tvf /dev/nrtape
```

When you have found the correct tape file, then note how many times you had to execute the *tar* command. If you found your file after the third *tar* command, then the file is in the third tape file. At this point you will want to rewind your tape with the following command:

```
mt -t /dev/tape rewind
```

To extract this file, you will use the *mt* command to skip past the first and second tape file and then use the *tar* command to retrieve the file as follows:

```
mt -t /dev/nrtape fsf 2
tar xvf /dev/tape john_file
```

Remote Backup and Restore

To perform a backup to a tape device that is attached to a remote machine (either a Silicon Graphics system, or any UNIX system), you must know:

- the name of the machine
- the tape device on the remote machine
- the name (preferably un-passworded) of an account on that system.

Note: This procedure assumes that the remote machine is running UNIX and that it does support the BSD *rmt*(1M) (remote tape protocol).

Caution: Keep in mind that a backup won't do you any good if you can't recover it when you need it. If you make a backup on a non-Silicon Graphics machine, then the only way to recover that backup might depend on having the Silicon Graphics machine running, the non-Silicon Graphics machine running, and the network operational. You must adequately design and test your backup recovery procedure before you need it.

If you do not have a user login name without a password, you will have to set up a *.rhosts* file on the remote machine. Refer to the reference pages on *hosts*(4), *rhosts*(4), and *hosts.equiv*(4) for information on setting up a *.rhosts* file.

The reference page for each of the various tape utilities describe the method for accessing a remote tape drive. While it is not necessary for the remote tape drive to be on a Silicon Graphics system, some thought should be given to how these backups will be recovered if it is not a Silicon Graphics system.

The following examples use *tar*(1), *bru*(1), and *cpio*(1) to write to a remote tape drive. In these examples, the <host_name> is the name of the remote system, and the <tape_device> is the name of the device on the remote system. On a Silicon Graphics system, this could be */dev/tape*, or a device in */dev/rmt**. If you do not know the name(s) of the available tape devices on the remote system, contact whoever is responsible for maintaining the system, or read the reference page on *tps*(1).

Using tar

To use *tar*(1) to write, read, and extract a tape written on a remote system use the following commands (be sure to type each *tar* command on one line):

```
tar cvf guest@<host_name>:/dev/<tape_device>
<files_to_backup>
```

```
tar tvf guest@<host_name>:/dev/<tape_device>
```

```
tar xvf guest@<host_name>:/dev/<tape_device>
```

Using bru

To use *bru(1)* to write, read, and extract a tape written on a remote machine use the following commands be sure to type each *bru* command on one line):

```
bru cvf guest@<host_name>:/dev/<tape_device>
<files_to_backup>
```

```
bru tvf guest@<host_name>:/dev/<tape_device>
```

```
bru xvf guest@<host_name>:/dev/<tape_device>
```

Using cpio

To use *cpio*(1) to write, read and extract a tape written on a remote machine, use the following commands (be sure to type each *cpio* command on one line):

```
find <filelist_to_backup> -print | cpio -ovc -C1024000 -O
guest@<host_name>:/dev/<tape_device>
```

```
cpio -itvc -C1024000 -I guest@<host_name>:/dev/<tape_device>
```

```
cpio -ivc -C1024000 -I guest@<host_name>:/dev/<tape_device>
```

It is strongly recommended that full system backups be completed on either the local tape drive, or on another Silicon Graphics machine.

Checking an Archive

If you are performing a critical operation, such as changing the size of a file system, or upgrading hard disks, you should always completely back up the system.

In addition, you should check the integrity of the archive. Sometimes, a backup program will appear to function correctly, but the data is incorrectly copied to the archive.

The following are methods for checking data integrity.

Comparing Archived Files

You can compare files that are archived with the original files.

With *bru*, use the **-d** option. For example:

```
bru -d /usr
```

If you specify a single **-d**, *bru* reports when it discovers that a regular file's size or contents have changed since the archive was made.

If you use **-dd**, *bru* reports additional differences in modification dates, access modes, number of links for non-directory files, differences in the contents of symbolic links, owner IDs, and group IDs.

If you specify **-ddd**, *bru* reports additional differences in host devices, major and minor devices for special files, and time of last access for regular files.

If you use **-dddd**, *bru* reports all differences except the time of the last status change, major and minor device numbers for non-special files, and size differences for directories. Usually, **-dddd** provides information that is meaningful only when verifying a full backup of a relatively static file system.

You can also compare files using *tar*:

```
tar C
```

You see messages about the status of the files. Each message begins with a key character (a letter or symbol) that signifies the status of the file in the archive versus the original file. These characters are shown in Table 6-1.

Table 6-1 *tar* Comparison Key Characters

Key	Meaning:
=	The files compare
!	The files don't compare
?	Can't read the disk file
>	Disk file doesn't exist
L	Linked to an earlier file on the tape
S	Symbolic link
B	Block special file
C	Character special file
P	Named pipe

The *cpio* program does not have a built-in option to compare files. To compare the files on a *cpio* archive, you must extract the archive onto disk, then use a comparing program, such as *gdiff*(1), *diff*(1), *cmp*(1), or *dircmp*(1), or compare the checksum (*sum*(1)) of the extracted file with that of the original.

Inspecting an Archive for Consistency

The *bru* program provides an option, **-i**, to inspect an archive for internal consistency and data integrity. For example:

```
bru -i
```

If you add **-vv**, *bru* prints information from the archive header block:

```
bru -ivv
```

Neither *tar* nor *cpio* provides this sort of check. However, listing the contents of an archive is usually sufficient. Also, a reasonable check is to extract the files in the archive whilore sending the output to */dev/null*.

Restoring Files and File Systems

The principal reason to make backups of system files is to protect those files from loss in the event of human error or hardware failure. When a file is lost, it must then be restored from a backup. Sometimes entire file systems are lost and must be reconstructed from backups.

Restoring File Systems

You restore an entire file system if there has been data corruption (for example, due to bad tracks); if you remade the file system (for example, if you replaced a disk drive); or if all the files have been accidentally removed.

Restoring a File System From the System Maintenance Menu

If your root file system is damaged and your system cannot boot, you will need to restore your system from the System Maintenance Menu. This is the menu that appears when you interrupt the boot sequence before the operating system takes over the machine. To perform this recovery, you need two different tapes: your system backup tape and a bootable tape with the miniroot.

To be used with the System Recovery option of the System Maintenance Menu, the backup tape must have been created with the System Manager or with the *Backup*(1) command and must be a full system backup (beginning in the root directory (/) and containing all the files and directories on your system). Although the *Backup*(1) command is a front-end interface to the *bru*(1) command, *Backup* also writes the disk volume header on the tape so that the System Recovery option can reconstruct the boot blocks, which are not written to the tape using other backup tools.

For information on creating the system backup, see “Backing Up File Systems” on page 177. For information on creating the bootable tape, see “Making a Bootable Tape” on page 203.

If you do not have a full system backup made with the *Backup* command or System Manager, you will have to reinstall your system if your *root* or *usr* file systems are so badly damaged that the operating system cannot boot.

If you need to reinstall the system to read your tapes, install a minimal system configuration and then read your full system backup (made with any backup tool you prefer) over the freshly installed software. Existing files of the same path name on the disk are overwritten during a restore operation, even if they are more recent than the files on tape. This procedure should restore your system to its former state.

1. When you first start up your machine, you see the following prompt:

   ```
   Starting up the system....

   To perform system maintenance instead, press <Esc>
   ```

2. Press the <Esc> key. You see the following menu:

   ```
   System Maintenance Menu
   1   Start System
   2   Install System Software
   3   Run Diagnostics
   4   Recover System
   5   Enter Command Monitor
   ```

3. Enter the numeral 4, and press <Return>. You see the message:

   ```
   System Recovery...

   Press Esc to return to the menu.
   ```

   After a few moments, you see the message:

   ```
   Insert the installation tape, then press <enter>:
   ```

4. Insert your bootable tape or your original distribution (CD or tape) and press the <Enter> key. You see some messages while the miniroot is loaded. Next you see the message:

   ```
   Copying installation program to disk....
   ```

   Several lines of dots appear on your screen while this copy takes place.

5. You see the message:

   ```
   CRASH RECOVERY

   You may type sh to get a shell prompt at most questions.

   Remote or local restore: ([r]emote, [l]ocal): [l]
   ```

6. Press <Enter> for a local restoration. If your tape drive is on another system accessible by the network, press `r` and then the <Enter> key. You are prompted for the name of the remote host and the name of the tape device on that host. If you press <Enter> to select a local restoration, you see the message:

   ```
   Enter the name of the tape device: [/dev/tape]
   ```

   You may need to enter the exact device name of the tape device on your system, since the miniroot may not recognize the link to the convenient */dev/tape* file name. As an example, if your tape drive is drive #2 on your integral SCSI bus (bus 0), the most likely device name is */dev/rmt/tps0d2nr*. If it is drive #3, the device is */dev/rmt/tps0d3nr*.

7. The system prompts you to insert the backup tape. When the tape has been read back onto your system disks, you are prompted to reboot your system.

Procedure for System Recovery from a Remote Tape Drive

Ensure that you have a Silicon Graphics distribution tape with installation tools on it. It will say, *Contains Installation Tools*.

1. Determine the remote tape drive you are using.
2. Insert the tape in the remote drive.
3. Execute the *mt ret* command on all backup tapes you will be using.
4. You must edit the */usr/etc/inetd.conf* file on the remote system with the tape drive before continuing (each entry below will appear on one continuous line with commands separated by tabs):

   Find this entry:

   ```
   tftp dgram udp wait guest /usr/etc/tftpd tftpd -s /usr/
   local/boot
   ```

   And change it to read as follows:

   ```
   tftp dgram udp wait guest /usr/etc/tftpd tftpd
   ```

5. Bring the system you are recovering to the System Maintenance Menu.

6. Select option `5) Command Monitor` from the System Maintenance Menu.
7. Enter the commands:

   `setenv netaddr` ***`address`*** (*address* = your internet address)

   ```
   init
   exit
   ```

8. Select `4) Recover System` from the System Maintenance Menu.
9. Follow the directions for recovering from the backup tapes.
10. When done, exit the program and bring up the system.

Restoring a File System with *bru*

Complete information on using the *bru* command and all its options is available in the *bru*(1) reference page. This command extracts the entire contents of a backup tape:

```
bru -x
```

Restoring a File System with *Restore*

The *Restore* command is a shell script that uses *bru* to extract files from a backup. Tapes made using the graphical System Manager can also be read from the System Maintenance Menu, using *Restore*. The following are examples of the *Restore* command:

```
Restore
```

You are prompted to insert the tape into the drive. You can recover multi-volume backups with *Restore*.

To extract a single file, use this command:

```
Restore file1
```

With the **-h** option, you can specify the tape drive on a different host workstation:

```
Restore -h guest@alice.cbs.tv.com file1
```

You must have *guest* login privileges in order to use files from a remote drive.

Files are restored into the current directory if the backup was made with relative path names. Relative path names are those that do not begin with a slash (/) character. Path names that begin with a slash are known as *absolute* path names. For example, */usr/bin/vi* is an absolute path name. The leading slash indicates that the path name begins at the root directory of the system. In contrast, *work/special.project/chapter1* is a relative path name since the lack of a leading slash indicates that the path begins with a directory name in the current directory.

Existing files of the same path name on the disk are overwritten during a restore operation even if they are more recent than the files on tape.

Restoring a File System with *restore*

Use *restore* to recover files and file systems made with the *dump* program. There are two ways to use *restore*:

- interactively
- non-interactively

Use the interactive option to recover moderate numbers of files from a *dump* archive. With the interactive feature of *restore*, you can browse the contents of a tape to locate and extract specific files.

Use the non-interactive mode to recover an entire backup. For example, place the backup in the drive and enter:

```
restore -x
```

Note that you *cannot* restore an active root file system. If your root file system is damaged and needs to be completely restored, you should probably reinstall the system, then rebuild it by extracting selected files from backup tapes.

Restoring Individual Files

The most common type of restoration you can perform is replacing single files that have been removed due to human error.

Restoring Individual Files with *bru*

To restore an individual file, type:

```
bru -x filename
```

If the file already exists on the file system, *bru* compares its modification date with that of the copy on tape. If the version of the file in the file system is more recent than the one on tape, *bru* does not extract the archived file.

To overwrite a file no matter what the modification dates are, use the **-u** option. With **-u**, you must specify what kinds of files to overwrite:

- *b* for block special files
- *c* for character special files
- *d* for directories
- *l* for symbolic links
- *p* for fifos (named pipes)
- *r* for regular files

For example, to force updating of any regular files on the archive, enter:

```
bru -xur
```

Restoring Individual Files with *tar* and *cpio*

To recover individual files from a *tar* archive, specify the name of the files on the command line:

```
tar xv file1 file2 directory/file3
```

The *cpio* command works much the same way; for example, enter:

```
cpio -id file1 directory/file2 < /dev/tape
```

The **-i** option causes *cpio* to read input from the tape drive, and the **-d** option causes it to create the directory it is extracting, if it doesn't already exist.

Restoring Individual Files with *restore*

To recover individual files from a *dump* archive, follow these steps:

1. Place the tape in the tape drive. Make sure it is write protected.
2. Enter:

   ```
   restore vi
   ```

   You see something like this:

   ```
   Verify tape and initialize maps
   Tape block size is 32
   Dump  date: Wed Feb 13 10:18:59 1991
   Dumped from: the epoch
   Level 0 dump of an unlisted file system on ralph:/dev/rusr
   Label: none
   Extract directories from tape
   Initialize symbol table.
   restore >>
   ```

3. You are now at the *restore*> prompt. You can browse the tape with *cd* and *ls*:

   ```
   restore > ls
   ```

   You see something like this:

   ```
   2      *./          973     source        1502  net/
   2      *../         149     d2/           1445  os/
   10     .cshrc       155016  debug/        1437  proto3.5/
   1463   .gamma       69899   dev/          1494  revE
   1464   .gamtables   696     etc/          2122  stand/
   160    .kshrc       137     bin/          3     tmp/
   1540   .lastlogin   1311412 jake/         128   unix
   819    .login       424     lib/          128   unix.debug
   820    .profile     9       lost+found/   4     usr/
   ```

   To continue browsing, enter the following commands to the *restore* prompt:

   ```
   restore > cd etc
   restore > pwd
   /etc
   ```

4. Start building a list of files that you want to extract. Use the *add* command to add the names of the files you want to the extract list:

   ```
   restore > add fstab
   restore > add fsck
   ```

If you enter *ls* at this point, you see a list of files, and *fsck* and *fstab* are marked with an asterisk to show they will be extracted.

If you want to remove a file from the list of those to be extracted, use the *delete* command:

```
restore > delete fstab
```

5. To restore the specified files, use the *extract* command:

```
restore > extract
Extract requested files
You have not read any tapes yet.
Unless you know which volume your file(s) are on you
should
start with the last volume and work towards the first.
Specify next volume #: 1
Mount tape volume 1
then enter tape name (default: /dev/tape) <Return>
extract file ./etc/fsck
Add links
Set directory mode, owner, and times.
set owner/mode for '.'? [yn] n
restore > q
```

To recover only a few files, you may wish to use the non-interactive options of *restore*. For example, enter:

```
restore -x ./usr/people/ralph/bus.schedule ./etc/passwd
```

This recovers the files *bus.schedule* and *passwd* from the archive.

Recovery after System Corruption

From time to time you may experience a system crash due to file corruption. Systems cease operating (''crash'') for a variety of reasons. Most common are software crashes, followed by power failures of some sort, and least common are actual hardware failures. Regardless of the type of system crash, if your system files are lost or corrupted, you may need to recover your system from backups to its pre-crash configuration.

Regardless of the nature of your crash, once you repair or replace any damaged hardware and you are ready to recover the system, see "Restoring a File System From the System Maintenance Menu" on page 190.

The System Maintenance Menu recovery command is designed for use as a full backup system recovery. After you have done a full restore from your last complete backup, you may restore newer files from incremental backups at your convenience. This command is designed to be used with archives made using the *Backup*(1) utility or through the System Manager. The System Manager is described in detail in the *Personal System Administration Guide*. System Recovery from the System Maintenance Menu is not intended for use with the *tar*(1), *cpio*(1), *dd*(1), or *dump*(1) utilities. You can use these other utilities after you have recovered your system.

Troubleshooting System Crash and Recovery

Caution: This section does not contain step-by-step instructions. The information in this article provides suggestions or tips designed for the advanced system administrator.

For more information on system recovery, see the reference pages for *savecore*(1M) and *crpt*(1).

If your system crashes, it will attempt to save the contents of physical memory to aid in debugging the problem. The *savecore* command will recover the information needed to troubleshoot the problem in the directory */usr/adm/crash*. *crpt* will automatically do some basic debugging and generate a report about the crash.

savecore

Sometimes, a system error causes the kernel to panic and *savecore* is called to save a copy of the contents of system memory. An error message similar to the following is displayed:

```
savecore: reboot after panic:
```

The contents of system memory is called core dump, and *savecore* writes the files to */usr/adm/crash*, and writes a reboot message in the shutdown log. These files are large and can be removed after analysis. Panics can be caused by software bugs or hardware failures.

To determine the cause of a panic or crash requires some investigative work. These suggestions may help to isolate what caused the crash:

- Check */usr/adm/SYSLOG* for other error messages.
- Save files in */usr/adm/crash* to tape before deleting them.
- Contact your support provider.

Running diagnostics and getting a backtrace of the crash can help to isolate the problem.

- Make sure that versions of the operating system are consistent, particularly if you recently upgraded system software.
- Remove any third party hardware.
- Run system diagnostics.
- Try to isolate what software was running on the machine when it crashed.
- Examine the core files using *crpt* (available on some machines).
- Contact your support provider.

For more information on this error, see the reference pages for *crpt*, *dbx(1)*, and *savecore(1M)*.

Changing the Default Backup Device

At some point in the life of your workstation, you may choose to add a new storage media device. If you wish to change the default backup device to use your new hardware, the following instructions provide complete information. You can also use the graphical System Manager; it is the preferred tool for this operation and is described completely in the *Personal System Administration Guide*. Note, however, that no matter which method you use to select your preferred device, installing new system software or using the *MAKEDEV*(1M) command may reset the default Backup device.

For more information on adding a storage media device, see "Tape Devices" on page 263.

Changing Device Nodes Manually

The method of changing the system default tape device is to relink */dev/nrtape* to the desired device.

Jot down in the system log book the name of the current tape device to which */dev/nrtape* is linked. To identify the device, examine the major and minor device numbers of */dev/nrtape* and the device nodes in */dev/mt*. Next, remove the link */dev/nrtape* and create a new link to the new device. In this case, the new device will be *mt/tps0d4*. Use the following procedure:

1. Enter the commands:

   ```
   cd /dev
   ls -l nrtape
   ```

   You see something similar to this:

   ```
   crw-rw-rw-  3 root   sys    144, 65 Mar 15 16:10 nrtape
   ```

   The major and minor device numbers are 144 and 65, respectively.

2. Examine the device numbers of all tape devices by entering the command:

   ```
   ls -l mt
   ```

   You see something similar to this:

   ```
   total 0
   crw-rw-rw-  2 root sys  144, 32 Mar 23 1993 tps0d4
   crw-rw-rw-  2 root sys  144, 33 Mar 23 1993 tps0d4nr
   crw-rw-rw-  2 root sys  144, 35 Mar 23 1993 tps0d4nrns
   crw-rw-rw-  2 root sys  144, 34 Mar 23 1993 tps0d4nrs
   crw-rw-rw-  3 root sys  144, 64 Mar 23 1993 tps0d2
   crw-rw-rw-  3 root sys  144, 65 Mar 23 1993 tps0d2nr
   ```

   The device at the bottom of this listing has the matching major and minor device numbers and therefore must be the device you're looking for. If more than one device has the correct major and minor numbers, then either device will do.

3. Remove the */dev/nrtape* link and create the new link with the same name. Use the commands:

```
rm /dev/nrtape
ln mt/tps0d4 /dev/nrtape
```

Most programs use */dev/nrtape* as the default tape device. If a program does not seem to be working correctly, first ensure that it is using the correct tape device.

Copying the Software Distribution

From time to time you may need to recover your system from a particularly bad failure, such as a hard disk failure. For most circumstances, you should need to use only your existing distribution media to boot your system for recovery purposes. However, if you have made a complete backup of your system, including any system files that may have changed in the course of your system tuning and development, it is a simple matter to restore your system.

As part of your IRIX software distribution, you received media (tape or CD) containing the ''miniroot.'' This media is specially formatted to allow you to boot the miniroot operating system to allow installation of the IRIX software.

If you do not have constant access to the distribution media, you can make a bootable tape. This process requires access to the distribution from the media at least once, or access to a distribution directory. The bootable software on the distribution is saved in a special format that cannot be archived like normal files and directories.

Copying the Distribution Media

You may wish to make a copy of your entire distribution media to save wear on your original distribution. (This is more of a concern for tape media than for CD-ROM.)

If your distribution media is a compact disc, you must have a CD-ROM device and a tape drive. If your distribution media is a tape, you must have either two tape drives, or a tape drive and significant free disk space on your system. However, if the distribution software already exists in a distribution

directory on your workstation or another workstation, you can copy it directly to your new media.

When you are ready to create the new bootable media, you will need your distribution media and your new tape. To effect a tape-to-tape transfer with one tape drive, perform the following steps:

1. Use the command:

   /bin/su

   To become the superuser on your system, you need to know the root password for the workstation. If the distribution software is present on a remote system, you should also be logged in to that system as the superuser.

2. Use the *distcp*(1M) command to copy the bootable image from the original distribution media to your disk or new media. The following example command copies the distribution from the original tape media to a distribution directory on your system:

   distcp /dev/nrtape /usr/tmp/dist

3. Remove the original media from the tape drive and insert your new tape. Use the *distcp* command to copy the distribution to the tape:

   distcp /usr/tmp/dist/* /dev/tape

   Your bootable tape should now be ready.

If your system has two tape drives, the *distcp* command to use is:

distcp /dev/nrtape1 /dev/tape2

If your distribution media is on compact disc, mount the CD-ROM file system on your system, and use *distcp* to copy the distribution software to your tape drive. The following command performs the copy:

distcp /CDROM/dist/* /dev/nrtape

If the distribution already exists in a directory on your system or a network host, the *distcp* command is similar to that for a compact disc, except that the directory where the CD file system is mounted should be replaced with the name of the host system and the distribution directory on the host system. The following example command illustrates this:

distcp hostname:/dist/* /dev/tape

For complete information on the use of the *distcp* and *mkboottape* commands, see the *distcp*(1M) and *mkboottape*(1M) reference pages.

Making a Bootable Tape

It is likely that a current software distribution will be too large to fit on an average 1/4 inch cartridge tape. It is often valuable (especially for system recovery) to have a tape that includes only the stand-alone shell (SASH). Using this tape, you can boot your system from the System Recovery option of the System Maintenance Menu and restore your system from full system backups made with the *Backup*(1M) command or the System Manager. See "Restoring a File System From the System Maintenance Menu" on page 190 for more information.

To make a miniroot (SASH) tape, follow the instructions for copying the distribution but instead of giving the asterisk to indicate all files in the distribution, specify the *sa* file alone. For example, from a CD-ROM distribution, give the following command:

```
distcp hostname:/CDROM/dist/sa /dev/tape
```

Backup Strategies

You should develop a regimen for backing up the system or systems at your site and follow it closely. That way, you can accurately assess which data you can and cannot recover in the event of a mishap.

Exactly how you perform backups depends upon your workstation configuration and other factors. Regardless of the strategy you choose, though, you should always keep at least two full sets of reasonably current backups. You should also encourage users to make their own backups, particularly of critical, rapidly-changing files. Users' needs can change overnight, and they know best the value of their data.

Workstation users can back up important files using the System Manager, found in the "System" tile on your screen. The System Manager is described in detail in the *Personal System Administration Guide*. Make sure users have access to an adequate supply of media (for example, cartridge tapes), whether new or used.

If your media can handle your largest file system with a single volume, you don't have to use an incremental backup scheme, though such a system reduces the amount of time you spend making backups. However, if you must regularly use multiple volumes to back up your file systems, then an incremental backup system will reduce the number of tapes you use.

The following sections discuss the different aspects of backing up data.

When to Back Up Data

How often you back up your data depends upon how busy a system is and how critical the data is. A simple rule of thumb is to back up any data on the system that is irreplaceable or that someone would not want to reenter.

Root File Systems

On most systems, the root file system is fairly static. You do not need to back it up as frequently as the */usr* file system.

Changes may occur when you add software, reconfigure hardware , change the site-networking (and the system is a server or network information service [NIS] master workstation), or change some aspect of the workstation configuration. In some cases, you can maintain backups only of the individual files that change, for example, */unix*, */etc/passwd*, and so forth.

This process of backing up single files is not always simple. Even a minor system change such as adding a user affects files all over the system, and if you use the graphical System Manager, you may tend to forget all the files that may have changed. Also, if you are not the only administrator at the site, you may not be aware of changes made by your coworkers. Using complete file system backup utilities, such as the System Manager or *bru*, on a regular schedule avoids these problems.

A reasonable approach would be to back up the root partition once a month. In addition to regular backups, here are some specific times to back up a root file system:

- whenever you add users to the system, especially if the workstation is an NIS master workstation
- just before installing new software

- after installing new software and when you are certain the software is working properly

If your system is very active, or if you are not the only administrator, you should back up the root file system regularly.

User File Systems

The */usr* file system, which often contains both system programs (such as in */usr/bin*) and user accounts, is usually more active than a root file system. Therefore, you should back it up more frequently.

At a typical multiuser installation, backing up once per day, using an incremental scheme, should be sufficient.

Incremental Backups

Incremental backups can use fewer tapes to provide the same level of protection as repeatedly backing up the entire file system. They are also faster than backing up every file on the system.

An incremental scheme for a particular file system looks something like this:

1. On the first day, back up the entire file system. This is a monthly backup.
2. On the second through seventh days, back up only the files that changed from the previous day. These are daily backups.
3. On the eighth day, back up all the files that changed the previous week. This is a weekly backup.
4. Repeat steps 2 through 3 for four weeks (about one month).
5. After four weeks (about a month), start over, repeating steps 1 through 4.

You can recycle daily tapes every month, or whenever you feel safe about doing so. You can keep the weekly tapes for a few months. You should keep the monthly tapes for about one year before recycling them.

Incremental Backups with *bru*

You can use the incremental option *bru* to create incremental backups. For example:

1. Create a complete backup of the */usr* file system:

   ```
   bru -c
   ```

2. Each day, back up the files that have changed since the previous daily backup:

   ```
   bru -c -n 25-Nov-1992 /usr
   bru -c -n 26-Nov-1992 /usr
   bru -c -n 27-Nov-1992 /usr
   bru -c -n 28-Nov-1992 /usr
   bru -c -n 29-Nov-1992 /usr
   bru -c -n 30-Nov-1992 /usr
   bru -c -n 1-Dec-1992 /usr
   ```

3. Every week, back up the files that have changed since the last weekly backup:

   ```
   bru -c -n 25-Nov-1992 /usr
   ```

   Note that the dates listed in the command examples above are place holders. Use appropriate current dates in your command lines.

4. At the end of four weeks, perform a complete backup and start the process over.

This is a common incremental backup scheme.

Incremental Backups with *tar* and *cpio*

Although *tar* and *cpio* do not have built-in mechanisms for incremental backups, you can use other system commands to accomplish this task.

The following example uses the same incremental scheme as presented in the preceding section to back up the */usr* file system. It uses the *find* command to determine which files to archive:

1. Go to the top of the file system that you want to back up. For example:

   ```
   cd /usr
   ```

2. Create a complete backup of the file system. With *tar*:

```
tar cv .
```

With *cpio*:

```
cpio -oLp .
```

3. Each day, back up the files that have changed since the previous daily backup. With *tar*:

   ```
   find /usr -mtime 1 -print | tar cvf -
   ```

   With *cpio*:

   ```
   find /usr -mtime 1 -print | cpio -pdL
   ```

4. Every week, back up the files that have changed since the last weekly backup. With *tar*:

   ```
   find /usr -mtime 7 -type f -print | tar cvf -
   ```

   With *cpio*:

   ```
   find /usr -mtime 7 -type f -print | cpio -pdL
   ```

5. At the end of four weeks, perform a complete backup and start the process over.

Incremental Backups with *dump*

The *dump* utility is designed for incremental backups, and it archives not only regular files and directories, but also special files, links, and pipes.

To create an incremental backup, specify an increment number when you use *dump*. The *dump* program archives all files that have changed since the last appropriate increment and special files such as links and named pipes. To recover files from an archive, use the *restore* command.

The *dump* program is designed specifically to create incremental backups. It refers to the increments as *levels*, and each level is assigned a number:

- A level 0 backup archives all files in a file system.
- Backup levels 1–9 archive all files that have changed since the previous backup of the same or lesser level.

For example, this command backs up all files on the */usr* file system:

```
dump 0 /dev/usr
```

This command backs up those files that have changed since the previous level 0 dump:

```
dump 1 /dev/usr
```

This command archives those files that have changed since the previous level 1 dump:

```
dump 2 /dev/usr
```

If the next *dump* command you give specifies level 1, *dump* backs up the files that have changed since the last level 0, but not those that have changed since the last level 2.

This numbering system gives you enormous flexibility so you can create a backup schedule to fit your specific needs.

Backing Up Files Across a Network

If you are managing a site with many networked workstations, you may wish to save backups on a device located on a central workstation.

To back up across a network, use the same basic backup commands, but with a slight change. Enter:

systemname`:/dev/tape`

If required, specify an account on the remote device:

user@systemname`:/dev/tape`

Users can use a central tape drive from their workstations with this method. Note that if you are backing up to a remote tape drive on a workstation that is not made by Silicon Graphics, the device name */dev/tape* may not be the correct name for the tape drive. Always learn the pathname of the tape device before executing the backup commands.

For example:

```
tar cvf guest@alice:/dev/tape ./bus.schedule
```

or

```
cpio -ovcO guest@alice:/dev/tape ./bus.schedule
```

Automatic Backups

You can use the *cron* utility to automatically back up file systems at predetermined times. The backup media must be already mounted in the drive, and, if you want this to be truly automatic, it should have enough capacity to store all the data being backed up on a single volume. If all the data won't fit on a single volume, then someone must manually change volumes.

Here is an example *cron* command to back up the */usr/src* hierarchy to */dev/tape* (tape drive) every morning at 03:00 using *bru*:

```
0 3 * * * bru -c -f /dev/tape /usr/src
```

Place this line in a *crontabs* file, such as */var/spool/cron/crontabs/root*.

This sort of command is useful as a safety net, but you should not rely on automatic backups. There is no substitute for having a person monitor backup process from start to finish and properly archive and label the media when the backup is finished. For more information on using *cron* to perform jobs automatically, see "Automating Tasks with at(1), batch(1), and cron(1M)" on page 29.

Storing Backups

Store your backup tapes carefully. Even if you create backups on more durable media, such as optical disks, take care not to abuse them.

Do not subject backups to extremes of temperature and humidity, and keep tapes away from strong electromagnetic fields. If there are a large number of workstations at your site, you may wish to devote a special room to storing backups.

Store magnetic tapes, including 1/4 in. and 8 mm cartridges, upright. Do not store tapes on their sides, as this can deform the tape material and cause the tapes to read incorrectly.

Make sure the media is clearly labeled and, if applicable, write-protected. Choose a label-color scheme to identify such aspects of the backup as what

system it is from, what level of backup (complete versus partial), what file system, and so forth.

To minimize the impact of a disaster at your site, such as a fire, you may want to store main copies of backups in a different building from the actual workstations. You will have to balance this practice, though, with the need to have backups handy for recovering files.

If backups contain sensitive data, take the appropriate security precautions, such as placing them in a locked, secure room. Anyone can read a backup tape on a system that has the appropriate utilities.

How Long to Keep Backups

You can keep backups as long as you think you need to. In practice, few sites keep system backup tapes longer than about a year before recycling the tape for new backups. Usually, data for specific purposes and projects is backed up at specific project milestones (for example, when a project is started or finished). As site administrator, you should consult with your users to determine how long to keep file system backups.

With magnetic tapes, however, there are certain physical limitations. Tape gradually loses its flux (magnetism) over time. After about two years, tape can start to lose data.

For long-term storage, you should re-copy magnetic tapes every year to year-and-a-half to prevent data loss through deterioration. When possible, use checksum programs, such as the *sum*(1) utility, to make sure data hasn't deteriorated or altered in the copying process. If you want to reliably store data for several years, consider using optical disk.

Reusing Tapes

You can reuse tapes, but with wear, the quality of a tape degrades. The more important the data, the more precautions you should take, including using new tapes.

If a tape goes bad, mark it as "bad" and discard it. You should write "bad" on the tape case before you throw it out so that someone doesn't accidentally

try to use it. You should never try to reuse an obviously bad tape. The cost of a new tape is minimal compared to the value of the data you are storing on it.

Troubleshooting

From time to time you will experience backup failures. It is vitally important that you determine the cause of the failure. Most often, the failure will be due to worn or faulty media. Proceeding without determining the cause of a failure makes all your future backups suspect and defeats the purpose of backups.

Troubleshooting Unreadable Backup Tapes

The reasons a backup tape might be unreadable include:

- the data on the backup tape is corrupted due to age or media fault
- the tape head is misaligned now, or was when the backup was made
- the tape head is dirty now, or was when the backup was made
- A wrong density tape drive.
- The wrong program is being used to restore the tape to disk.

If you suspect the problem is that the tape drive is the wrong density, enter the command:

```
mt stat
The command will return information about the tape drive,
including the drive type.
```

Use the following table to determine which tapes that can be read given a particular tape drive.

Table 6-2 Tapes that can be read given a particular tape drive

Tape Drive	Tapes That Can Be Read
QIC150	QIC150, QIC24, QIC 1000
QIC24	QIC24
QIC02	QIC02

If the problem is the wrong program being used to restore the tape to disk, it is most likely that the tape is mislabeled as to the command used to create the archive, or you are attempting to restore a System Manager backup onto a running system. To install a system backup tape, you must shut down the system first. A backup made with the System Manager tool can only be restored by following the directions in "Restoring a File System From the System Maintenance Menu" on page 190.

The following is a table of commands and tools for making a tape backup and the corresponding command and tool to restore the backup tape:

Table 6-3 Commands and tools used to backup or restore on tape

Backup	Restore
System Manager (Full)	prom (monitor menu)
System Manager (Partial)	System Manager
Backup	bru or Restore (note the upper case 'R')
tar	tar
bru	bru
cpio	cpio

Reading Media from Other Systems

You may not be able to read data created on another vendor's workstation, even if it was made using a standard utility, such as *tar* or *cpio*. One problem may be that the tape format is incompatible. Make sure the tape drive where the media originated is compatible with your drive.

If you are unable to verify that the drives are completely compatible, use *dd* to see if you can read the tape at the lowest possible level. Place the tape in the drive and enter the command:

```
mt blksize output
```

The *mt*(1M) command with these options will tell you the blocksize used to write the tape. Set the blocksize correspondingly (or larger) when you use *dd* to read the tape. For example, if the blocksize used was 1024 bytes, use the command:

```
dd if=/dev/tape of=/usr/tmp/outfile bs=1024
```

If *dd* can read the tape, it displays a count of the number of records it read in and wrote out. If *dd* cannot read the tape, make sure your drive is clean and in good working order. Test the drive with a tape you made on your system.

If you can read the tape with *dd*, and the tape was created using a standard utility, such as *tar* or *cpio*, you may be able to convert the data format with *dd*. Several conversions may help:

- swab–swap every pair of bytes
- sync–pad every input block to ibs
- block–convert ASCII to blocked ASCII
- unblock–convert blocked ASCII to ASCII
- noerror–do not stop processing on an error

The *dd* program can convert some completely different formats:

- ascii–convert EBCDIC to ASCII
- ebcdic–convert ASCII to EBCDIC
- ibm–slightly different map of ASCII to EBCDIC

Converting case of letters:

- lcase–map alphabetics to lower case
- ucase–map alphabetics to upper case

If the data was written on another vendor's system, you may be able to convert it using *dd*, then pipe the converted output to another utility to read it.

Many other vendors use byte-ordering that is the reverse of the order used by IRIX. If this is the case, you can swap them with the following command:

```
dd if=/dev/tape conv=swab of=/usr/tmp.O/tapefile
```

Then use the appropriate archiving utility to extract the information from */tmp/tapefile* (or whatever filename you choose). For example, use this command to extract information if the *tar* utility was used to make the tape on a byte-swapped system:

```
tar xvf /usr/tmp.O/tapefile .
```

Or you can use the no-swap tape device to read your files with the following *tar* command line:

```
tar xvf /dev/rmt/tps0d4ns
```

Of course, if your tape device is not configured on SCSI channel 4, the exact */dev/rmt* device name may be slightly different. For example, it could be */dev/rmt/tps0d3ns*.

It is good practice to preview the contents of a *tar* archive with the **t** keyword before extracting. If the tape contains a system file and was made with absolute pathnames, that system file on your system could be overwritten. For example, if the tape contains a kernel, */unix*, and you extract it, your own system kernel will be destroyed. The following command previews our above example archive:

```
tar tvf /tmp/tarfile
```

If you wish to extract such a tape on your system without overwriting your current files, use this command to force the extraction to use relative pathnames:

```
tar Rx
```

or the corresponding *bru* command:

```
bru -j
```

Errors Creating the Backup

If you see errors on the system console when trying to create a backup, some causes are:

- The tape is not locked in the drive. You may see an error message similar to this:

  ```
  /dev/nrtape rewind 1 failed:Resource temporarily
  unavailable
  ```

  Make sure the tape is locked in the drive properly. See your *Owner's Guide* if you do not know how to lock the tape in the drive.
- File permission problems. These are especially likely with file-oriented backup programs; make sure you have permission to access all the files in the hierarchy you are backing up.
- The drive requires cleaning and maintenance.
- Bad media; see "Testing for Bad Media" on page 217.

If you encounter problems creating backups, fixing the problem should be your top priority.

Restoring the Wrong Backup

If you accidentally restore the wrong backup, you should rebuild the system from backups. Unless you are very sure of what you are doing, you should not simply restore the correct backup version over the incorrect version. This is because the incorrect backup may have altered files that the correct backup won't restore.

In the worst possible case, you may have to reinstall the system, then apply backups to bring it to the desired state.

Here are some basic steps to recovering a file system. If you used incremental backups, such as from *backup* or *bru*:

1. Make a complete backup of the current state of the file system. If you successfully recover the file system, you will not need this particular backup. But if there is a problem, you may need to return to the current, though undesirable, state.
2. Start with the first complete backup of the file system that was made prior to the backup that you want to have when you're finished. Restore this complete backup.
3. Apply the series of incremental backups until you reach the desired (correct) backup.

If you accidentally restored the wrong file-oriented backup (such as a *tar* or *cpio* archive):

1. Make a complete backup of the affected file system or directory hierarchy. You may need this not only as protection against an unforeseen problem, but to fill any gaps in your backups.
2. Bring the system to the condition it was in just before you applied the wrong backup.

   If you use an incremental backup scheme, follow steps 2 and 3 above (recovering from the wrong incremental backup).

   If you use only utilities such as *tar* and *cpio* for backups, use what backups you have to get the system to the desired state.
3. Once the system is as close as possible to the correct state, restore the correct backup. You are finished. If the system is in the desired state, skip the remaining steps.

   If you cannot bring the system to the state it was in just before you applied the wrong backup, continue with the next series of steps.
4. If you cannot manage to bring the system to the correct state (where it was just before you restored the wrong backup), get it as close as possible.
5. Make a backup of this interim state.
6. Compare the current interim state with the backup you made at the outset of this process (with the incorrect backup applied) and with the backup you wish to restore. Note which files changed, which were added and removed, and which files remain unchanged in the process of bringing the system to the desired state.

Using these notes, manually extract the correct versions of the files from the various tapes.

Testing for Bad Media

Even the best media can go bad over time. Symptoms are:

- Data appears to load onto the tape correctly, but the backup fails verification tests. (This is a good reason to always verify backups immediately after you make them.)

  Another tape is then able to back up the data successfully and pass verification tests.
- Data retrieved from the tape is corrupted, while the same data loaded onto a different tape is retrieved without problems.
- The backup media device driver (such as the SCSI tape driver) displays errors on the system console when trying to access the tape.
- You are unable to write information onto the tape.

If errors occur when you try to write information on a tape, make sure the tape is not simply write-protected. Be sure you are using the correct length and density tape for your drive.

Make sure that your drive is clean and that tape heads are aligned properly. It is especially important to check tape head alignment if a series of formerly good tapes suddenly appear to go bad.

Once you are satisfied that a tape is bad, mark it as a bad tape and discard it. Be sure to mark it ‘‘bad’’ to prevent someone else from accidentally using it.

Chapter 7

Disks and Tape Drives

Chapter 7 describes the administration of hard disks and tape drives under IRIX. Some tasks described in this chapter can be performed using the graphical System Manager, described in the Personal System Administration Guide. Most IRIX systems use hard disks and tape drives, and those that do not must be connected to a system that does use them. Every Administrator will deal with disk configuration at some point, and this chapter describes in detail the method of performing the following tasks:

- *Disk installation and configuration.*
- *Repartitioning a hard disk.*
- *Adding swap space to the system.*
- *Using logical volumes and disk striping to make the most of your disk resources.*
- *Using the bad block tools to eliminate faulty disk blocks from your environment.*
- *Installing, configuring, and maintaining tape drives.*

Chapter 7

Disks and Tape Drives

This chapter covers what you need to know about the disk devices and tape drives on your workstation or server. All system software and user files are kept on the hard disk device(s). The cartridge tape device is used primarily for file system backups and data transfer.

Topics covered in this chapter are:

- Identifying disk and tape devices to IRIX so that the system can use them. See "Identifying Devices to IRIX" on page 222.
- General information on the use of hard disks under IRIX. See "Hard Disks under IRIX" on page 225.
- Instructions for adding a hard disk to your system. See "Adding a Hard Disk" on page 226.
- Instructions on using the disk formatting tool. See "Formatting Disks Using fx" on page 229.
- Instructions for changing the partitions for file systems on the disk. See "Repartitioning a Hard Disk" on page 231.
- How to check, increase, and make the best use of system swap space. See "Swap Space" on page 237.
- Instructions for creating logical volumes to allow file systems to grow over several disks, and how to use disk striping to improve performance. See "Logical Volumes and Disk Striping" on page 244.
- Instructions for using the bad disk block handling tool to remove bad blocks from your system. See "The Bad-Block Handling Feature" on page 251.
- Instructions for the installation and use of floppy disk drives. See "Using Floppy Disks Under IRIX" on page 259.
- Instructions for the installation and use of tape drives. See "Tape Devices" on page 263.

For information on creating and administering file systems, see Chapter 8, "File System Administration." For information on backing up data onto tapes, see Chapter 6, "Backing Up and Restoring Files." If you are installing a hard disk or tape drive, see the installation instructions furnished with the hardware.

Identifying Devices to IRIX

Before a disk or tape device can be used with IRIX, it must be made known to the system. For equipment that comes with your computer, the process of identifying devices is part of configuration and is done automatically as the system is booted.

The standard method of handling the identification is through an entry in the */dev* directory of the *root* file system. Since an entry in a directory is a file (or another directory), conceptually a disk or tape device is treated as if it were a file. In practice, there are differences between standard files and device files, so the latter are referred to as *special* files.

The following output is the result of the *ls -l* command invoked on a user's ordinary file and the */dev* directory. It shows the difference in structure between regular and device files. This is a regular file:

```
-rw-r----- 1 ralph raccoons 1050 Apr 23 08:14 scheme.notes
```

These are device files:

```
brw------- 2 root sys 22,32 Apr 15 10:59 /dev/dsk/dks0d1s01
brw------- 2 root sys 22,32 Apr 15 10:59 /dev/root
brw------- 2 root sys 22,38 Apr 12 13:51 /dev/dsk/dks0d1s1
brw------- 2 root sys 22,38 Apr 12 13:51 /dev/usr
crw------- 2 root sys 22,32 Apr 15 10:58 /dev/rdsk/dks0d1s0
crw------- 2 root sys 22,32 Apr 15 10:58 /dev/rroot
crw------- 2 root sys 22,38 Apr 12 13:51 /dev/rdsk/dks0d1s1
crw------- 2 root sys 22,38 Apr 12 13:51 /dev/rusr
```

The above listing shows that the file *scheme.notes* is a regular file (indicated by the dash "-" in the first position of the line) with these characteristics:

- The file has 1050 characters.
- The file name is *scheme.notes*.

- It is owned by user *ralph* who is a member of the *raccoons* group.
- The owner has read/write permission, group members have read permission only, and other users have permissions for nothing at all.

The device file listing has some similar information to the listing of a normal file, but also contains additional information. In the field of a long listing where a regular file shows the byte count of the file, a device file displays two numerals called the *major* and *minor device numbers*.

Table 7-1 displays the parts of a device filename in bold face to illustrate its construction.

Table 7-1 Device Name Construction

Device Name	Component
/dev/dsk/dks0d1s0	/dev = device files directory
/dev/**dsk**/dks0d1s0	dsk = subdirectory for hard disk files
/dev/dsk/**dks**0d1s0	dks = SCSI device ips = ESDI device xyl = SMD device ipi = IPI device jag = VME SCSI device (Jaguar disk)
/dev/dsk/dks**0**d1s0	0 = disk controller 0
/dev/dsk/dks0**d1**s0	d0 = main hard disk (d1 for SCSI) d1 = second hard disk (d2 for SCSI)
/dev/dsk/dks0d1**s0**	s0 = partition 0

The device files shown have the following characteristics:

- Major and minor device numbers appear where the character count appears in the listing of a normal file.

  The major device number is the number of the device controller or driver; it indicates the device type, such as a terminal or hard disk. It is also an index into a table of devices in the kernel.

The minor device number is passed as a parameter to the driver. For example, an ESDI hard disk device name with a major number of 4 and a minor number of 0 indicates a primary drive and the root partition. The major and minor device numbers are found in the inode.

- There are devices that have identical major and minor numbers, but they are designated in one entry as a block device (a **b** in the first column) and in another entry as a character device (a **c** in the first column). Notice that such pairs of files have different file names or are in different directories (for example, */dev/dsk/dks0d1s0* and */dev/rdsk/dks0d1s0*).
- The files are owned by *root*, and no other user or group has permission to use them. This means that only processes with the *root* ID can read from and write to the device files. Tape devices, floppy drives, and tty terminals are exceptions to this rule.

  In this area, there are distinct differences between block and character devices. For more information on block and character devices, see the following section.

Block and Character Devices

Block devices or character devices are identified by the way in which they are accessed. Block devices are accessed through the integrated page cache, and character devices are accessed directly. The integrated page cache holds input to the block device until a predetermined amount of data has accumulated. Then the data is transmitted all at once. Similarly, only blocks of data of a certain size are read from a block device. The size of the block is generally configurable in the kernel. For more information on changing block sizes and on changing kernel parameters in general, see Chapter 5, "Tuning System Performance" and Appendix A, "IRIX Kernel Tunable Parameters."

When the device is read from or written to in blocks of varying sizes, a character device name is used. For example, some file maintenance routines do I/O in this manner. Character devices are also referred to as *raw* devices. The device file directory listing in "Identifying Devices to IRIX" on page 222 gives an example of the naming difference: the raw device name of the disk drive is in directory */dev/rdsk*, where the *r* indicates raw.

Hard Disks under IRIX

IRIX offers a choice of five disk interfaces: SMD, ESDI, SCSI, JAG, and IPI. These drives provide maximum flexibility in meeting mass storage requirements.

Each kind of disk drive is managed by a controller. Each type of controller can support a fixed number of drives. Your workstation can support a fixed number of controllers. (For the number and type of controllers supported by your model of workstation, see your hardware owner's guide.) The different types of controllers support the following number of disks per controller:

- SCSI (built-in controller plus one or more additional controllers): 7 disks per controller
- ESDI (up to 2 controllers): up to 4 disks per controller
- SMD (up to 4 controllers): 4 disks per controller
- JAG (up to 6 controllers): 14 disks per controller
- IPI (up to 4 controllers): 16 disks per controller

Table 8-2 provides a look at the relative, best-case characteristics of each interface.

Table 7-2 Disk Drive Performance

Disk	Form Factor	Avg. Seek	Transfer Rate	Raw Data Rate
SCSI	5-1/4″ or 3.5″	16 ms	.5-1.5MB/sec	1.25-2.50MB/sec
VME/SCSI	5-1/4″	16 ms	.5-1.5MB/sec	1.25-2.50MB/sec
VME/ESDI	5-1/4″	16 ms	.9-1.7MB/sec	1.25-2.50MB/sec
VME/SMD	8″	16 ms	2.0-2.1MB/sec	3.0MB/sec
VME/IPI	8″	16 ms	3.6MB/sec	6.0MB/sec

Note: Average seek time may vary, and the Transfer Rate column indicates sustained data transfer through the file system measured for sequential reads. Your perceived performance will probably not match these figures.

Rates vary for different types of drives within each class. Larger drives tend to be faster. Other drives and some types of systems will be slower.

Adding a Hard Disk

The process of adding a hard disk to your system does not end when the disk hardware has been installed and the system powered up again. There are software operations that must be performed to allow IRIX to see the new disk.

Configuring SCSI Disks With Add_disk(1M)

The *Add_disk*(1M) command performs all the necessary software operations to configure your system to use your new SCSI disk on your integral SCSI controllers. *Add_disk* cannot be used on VME bus SCSI controllers. For Jaguar disks attached to VME controllers, use *MAKEDEV*(1M) and *mknod*(1M). Most systems have only integral SCSI controllers, and are eligible to use *Add_disk*. If you are not sure if your SCSI controllers are eligible, use the *hinv*(1M) command to check your SCSI controller. You can use *Add_disk* if the entry for your SCSI controller indicates an integral controller, such as the following:

```
Integral SCSI controller 0: Version WD33C93A, revision 9
```

The syntax of the command is:

Add_disk [*controller number*] [*disk number*]

If you are adding a second disk on controller 0 to your system, you do not have to specify the disk or controller number, as adding disk 2 on controller 0 is the default. If you are adding a third (or greater) disk, or if you are adding a disk on a controller other than controller 0, you must specify the disk and controller.

Add_disk checks for valid file systems on the disk, and if any file systems are present, you are warned and asked for permission before the existing file systems are destroyed and a new file system is made.

The *Add_disk* command performs the following tasks:

- Links the appropriate device files
- Creates a file system on the disk
- Creates the mount directory
- Mounts the file system
- Adds the mount order to the */etc/fstab* file

Configuring Disks With MAKEDEV(1M)

If you need to create hard disk or other device files, use the *MAKEDEV*(1M) command:

```
/dev/MAKEDEV
```

MAKEDEV with no arguments creates the standard set of device files in the */dev* directory. You can create specific types of device files by supplying the appropriate arguments to *MAKEDEV*. This is a list of the disk devices that *MAKEDEV* can create:

ips	Creates special files for ESDI disks connected to an Interphase ESDI disk controller. See *ips*(7M) for details.
ipi	Creates special files for IPI disks connected to a Xylogics IPI disk controller. See *ipi*(7M) for details.
dks	Creates special files for SCSI disks. See *dksc*(7M) for details.
jag	Creates special files for Jaguar VME/SCSI disks. See *jag*(7M) for details.
xyl	Creates special files for SMD disks connected to a Xylogics SMD disk controller. See *xyl*(7M) for details.
disk	Creates all the disk device special files for the *dks*, *ips*, *ipi*, and *xyl* drives, and then creates links by which you can conveniently reference them without knowing the configuration of the particular workstation. The links *root*, *rroot*, *swap*, *rswap*, *usr*, *rusr*, *vh* and *rvh* are created to reference the current *root*, *swap*, *usr*, and volume header partitions. Note that if you have changed the partitions, for example by extending them using logical

volumes, these links will not be correct for your system and you must make them by hand. Use the *link*(1M) command to link the disk partitions to the appropriate names.

links Creates both disk and tape devices.

For a complete list of *MAKEDEV* options, see the *MAKEDEV* reference page. For a list of tape device options, see "Tape Devices" on page 263.

You may need to create specific files that are not created by *MAKEDEV*; for example, if you have extended a partition using a logical volume, use the *mknod*(1M) command.

The general format of the *mknod* command is:

mknod *name* **b** | **c** *major minor*

mknod *name* **p**

The options of *mknod* are:

name Specifies the *name* of the special file.

b Specifies a block device.

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

major The *major* number indicates a device type that corresponds to the appropriate entry in the block or character device switch tables.

minor The *minor* number indicates a unit of the device. It distinguishes peripheral devices from each other.

p Specifies the special file as a first-in, first-out device. This is also known as a *named pipe*.

When you format a disk you write basic address information and timing marks, and you identify bad areas on the disk (called *bad blocks*). This type of formatting should never be performed on SCSI disks. When you partition the disk, you divide the disk into various logical units. Partitioning can be done on any disk.

Formatting Disks Using fx

Many drive manufacturers ship their drives preformatted, and no further formatting is necessary. This is true of virtually all SCSI disks. However, if you must format a disk, use the *fx*(1M) command with the **-x** option to allow it to write to the disk. For disks, *fx* formats both sides of the disk into tracks and sectors that can be addressed by the disk controller. A portion of the disk called the *disk label* is reserved for data associated with the disk layout. For complete information on disk labels and their contents and management, see the *fx*(1M) reference page. The volume table of contents (vtoc) resides in the disk label. The vtoc (also called the volume header) shows how the partitions on the disk are allocated. On a hard disk, labels also 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.

The utility *prtvtoc*(1) is used to print out the current vtoc.

Formatting a New Disk

From time to time, you may find that you must install a new disk completely from scratch. The new disk may have no formatting or partitioning information on it at all. To format the disk and create partitions and file systems, perform the following steps:

1. Bring the system up into the System Maintenance Menu. Choose option 5 to enter the Command Monitor.
2. Once in the Command Monitor, give the *hinv* command and determine the correct piece of hardware from which you will boot *fx*. For example, a system with an IP19 (ARCS) processor and a single CD-ROM drive on SCSI address 4 would use the device `dksc(0,4)`. For the remainder of this example, we will use that device, although your device may be different. If your device is different, use your device information where applicable. See Appendix A of the *Software Installation Administrator's Guide* for a complete listing of all appropriate commands to boot *fx* from CD-ROM.
3. Insert a CD containing the installation tools into your system's CD-ROM drive. (If you have a tape drive, insert a tape with installation tools. If you are installing over a network connection, have the network

address of the workstation with the installation tools available. For detailed information on installations, see the *Software Installation Administrator's Guide*.)

4. Give the following command in the Command Monitor:

   ```
   boot -f dksc(0,4,8)sashARCS dksc(0,4,7)stand/fx.ARCS --x
   ```

5. You see the following messages and prompts. Your expected responses are in **bold face** type. These responses (and the default prompts) select the first disk:

   ```
   SGI Version 5.1 ARCS Nov 19, 1993
   fx: "device-name" = (dksc) <Enter>
   fx:  ctlr# = (0) <Enter>
   fx: drive# = (1) <Enter>
   ...opening dksc(0,1)
   ...controller test...OK
   Scsi drive type == CDC   94171-9 0184
   ---please choose one(? for help) .. to quit this menu)---
   [exi]t         [d]ebug       [l]abel         [a]uto
   [b]adblock     [exe]rcise    [r]epartition   [f]ormat
   ```

6. Select the *auto* command, and your disk will be prepared for file systems with a default set of partitions. To perform custom formatting, please refer to the *fx*(1M) reference page for complete instructions.

7. Once the formatting and partitioning is complete, exit *fx* and choose the option "Install System Software" from the System Maintenance Menu. The Install System Software option brings up *inst*(1M).

8. Once in *inst*, choose option 10, the *admin* option, from the *inst* main menu.

9. When you see the Administrative Commands Menu, choose option 10, the *mkfs* option, to make file systems on the disk you previously formatted with *fx*.

10. When the file systems have been made, you may proceed with your installation.

Complete information on using *inst*(1M) and on booting *inst* from a variety of media (local and remote CD-ROM network distribution directories) is available in the *Software Installation Administrator's Guide* and the *inst*(1M) reference page.

Repartitioning a Hard Disk

Partitions on the hard disk devices on your workstation are allocated in a standard arrangement. The arrangement varies according to the number of disk drives that you have and the size of those drives.

The first disk is typically partitioned to accommodate the *root*, *swap*, and */usr* partitions. Other disks may be partitioned to add additional space to the file systems based on the first disk, or may be partitioned into entirely new file systems.

The default partitions are generic in nature and should be evaluated by the system administrator. After your system has been in operation for a few months, you may decide that a different arrangement would better serve your users' needs.

Changing Hard Disk Partitions

This section describes how to change disk drive partitions. Once disks have been formatted and partitioned, these partitions may be used as file systems, parts of a logical volume, or as raw disk space.

Follow these steps before changing any partition:

1. If there is *any* valuable data on the disk to be repartitioned, make a complete backup using either the System Manager or the *Backup*(1) utility. The System Manager is the preferred utility and is described completely in the *Personal System Administration Guide*. Only backups made with *Backup*(1) or the System Manager will be available to the system from the System Recovery menu of the System Maintenance Menu. Other utilities require a full system installation to operate correctly. Repartitioning will make the current data on your disk inaccessible.
2. Make sure that the disk drive to be partitioned is not in use. That is, make sure that no file systems are mounted and no programs are accessing the drive.

The disk should be partitioned according to the use of the drive. Many users use the disk just as a single shared file system. Other users may want to have a smaller partition for private work. Before starting the partitioning process,

determine how much space is available on the drive, and then decide how to divide up that space.

You can partition the disk into as many as 16 sections, each one of any size. The size of a partition is measured in 512-byte blocks. Partitions are numbered starting at 0. Partition numbers 8, 9, and 10 are reserved for internal information.

As an example, you will repartition a 380MB SCSI drive to raise the size of the root partition. First, use *prtvtoc*(1) to print out the current information about the drive:

```
prtvtoc /dev/dks/dks0d1vh
```

The output looks like this:

```
* /dev/dsk/dks0d1vh (bootfile "/unix")
*     512 bytes/sector
*      45 sectors/track
*       9 tracks/cylinder
*    1558 cylinders
*       7 cylinders occupied by header
*    1551 accessible cylinders
*
* No space unallocated to partitions

Partition Type Fs   Start:sec (cyl) Size:sec  (cyl) Mount Dir
 0        efs  yes  2835      (7)   32400     (80)   /
 1        raw       35235     (87)  81810     (202)
 6        efs  yes  117045    (289) 513945    (1269) /usr
 7        efs       2835      (7)   628155    (1551)
 8        volhdr    0         (0)   2835      (7)
10        volume    0         (0)   630990    (1558)
```

If *prtvtoc* fails, use *hinv*(1) to make sure that you are examining the correct drive, or use *fx*(1) to make sure that the drive is formatted.

Look at the size column for partitions 0, 1, and 6. In this example, you have 32400 + 81810 + 513945 = 628155 sectors to use. (Recall that a sector is a 512-byte block.) Look at the start sector numbers, and notice that partition 7 overlaps 0, 1, and 6. Partition 0 is your *root* file system, and is mounted on the system's root directory (/). Partition 1 is your system's swap space, and is used by the IRIX kernel. Partition 6 is your */usr* file system, and it is mounted on the */usr* directory. In this example, you will take space from the

/usr file system and expand the *root* file system. Remember to back up *any* data you wish to preserve.

Now bring the system down and boot *fx* from the Command Monitor. Choose option 5 from the System Maintenance Menu. You see the Command Monitor prompt:

```
>>
```

Enter the following command at the Command Monitor prompt:

```
boot stand/fx --x
```

Next you see the initial *fx* prompts:

```
fx: "device-name" = (dksc)
```

fx prompts you for each part of the disk name. You are offered a default option for each part of the disk name. The default option is **dksc**, which indicates a SCSI disk. To choose the default option, simply press **<Return>** at the prompt. To specify another disk type, enter the appropriate part of the disk name at the prompt. For example, at the first prompt, you might want to enter **ipi** if you are repartitioning an ipi disk, as opposed to a SCSI disk.

You see several other prompts of the type shown above before you see the first *fx* menu. The prompts ask you to specify the disk controller number and drive number. The default values at these prompts always specify the root disk of the workstation or server. Once you have specified the controller and disk number, you see the *fx* main menu:

```
---- please choose one (? for help. .. to quit this menu)----
[exi]t               [d]ebug/               [l]abel/
[b]adblock/          [exe]rcise/            [r]epartition/
fx>
```

The *exit* option quits *fx*, while the other commands take you to sub-menus. (The slash [/] character after a menu option indicates that choosing that option leads to a sub-menu.) For complete information on all *fx* options, see the *fx* reference page. To partition the drive, choose the `[r]epartition` option by entering **r** and then pressing **<Return>**. Next you see the volume table of contents for the root disk, or the disk you specified when *fx* was started, followed by the **repartition** menu:

```
----- partitions-----
part   type      cyls     blocks        Megabytes(base+size)
```

```
0:     efs       7+80     2835+32400    1+16
1:     rawdata   87+202   35235+81810   17+40
6:     efs       289+1269 117045+513945 57+251
7:     efs       7+1551   2835+628155   1+307
8:     volhdr    0+7      0+2835        0+1
10:    entire    0+1550   0 + 630990    0+308
capacity is 631017 blocks
---- please choose one (?for help .. to quit this menu)----
[ro]otdrive      [o]ptiondrive      [re]size      [e]xpert
fx/repartition>
```

Choose the **resize** option to change the size of partitions on the disk. You see the following warning message:

```
Warning: you will need to re-install all software and
restore use data from backups after changing the partition
layout. Changing partitions will cause all data on the drive
to be lost. Be sure you have the drive backed up if it
contains any user data. Continue?
```

Answer **y** to continue with the repartition. You see the next warning message:

```
After changing the partition, the other partitions will be
adjusted around it to fit the change. The result will be
displayed and you will be asked whether it is OK, before the
change is committed to disk. Only the standard partitions
may be changed with this function. Type ? at prompts for a
list of possible choices.
fx/repartition/resize: partition to change = (swap)
```

The prompt after the warning message offers the swap space partition as the default partition to change, but in this example we will designate the root partition to be resized. Enter **root** and press **<Return>** at the prompt. You see the current information for the root partition:

```
current:  type efs   base: 7 cyls,   2835 blks   1 Mb
                    len: 80 cyls,   32400 blks  16 Mb
fx/repartition/resize: partitioning method = (megabytes)
```

Press **<Return>** to use megabytes as the method of repartitioning. Other

options are to use percentages of total disk space, numbers of disk blocks, or numbers of disk cylinders. Megabytes and percentages are the easiest methods to use to partition your disk. Next you see the following prompt:

```
fx/repartition/resize: size in megabytes (max 307) = (16)
```

Now you must select the new size of the root partition. The default option presented to you is to retain the current size of 16MB. For this example, we will increase the size to 20MB. Enter **20** at the prompt and press **<Return>**. Next, the new partition map is displayed:

```
-----partitions-----
part     type     cyls      blocks           Megabytes
0        efs      7+101     2835+40960       1+20
1        rawdata  108+180   43795+73250      21+36
6        efs      289+1269  117045+513945    57+251
8        volhdr   0+7       0+2835           0+1
10       entire   0+1558    0+630990         0+308

Use the new partition layout? (no)
```

Note that the 4 megabytes that you added to your root partition were taken from the swap partition. Ultimately, we want those megabytes to come from the */usr* partition, but for the moment, accept the new partition layout. Enter **yes** at the prompt and press **<Return>**. The new partition table is printed again, along with the total disk capacity. Then you are returned to the repartition menu. Select **resize** again to transfer space from the */usr* partition to the swap area.

```
fx/repartition>resize
```

You see the same warning messages again, followed by the prompt:

```
fx/repartition/resize: partition to change = (swap)
```

This time press **<Return>** to change the size of the swap partition. You see:

```
current: type rawdata  base:108 cyls, 43795 blks   21 Mb
                       len: 180 cyls, 73250 blks   36 Mb
fx/repartition/resize: partitioning method = (megabytes)
```

Press **<Return>** again to use megabytes as the method of repartition. You see the following prompt:

```
fx/repartition/resize: size in megabytes (max 307) = (36)
```

Since we added 4 megabytes to expand the root file system from 16 to 20 megabytes, enter **40** at this prompt to expand the swap space to its original size. (If your system is chronically short of swap space, you can take this opportunity to add some space by entering a higher number.) Press **<Return>** after you enter the new size of the partition. You see the new partition table:

```
-----partitions-----
part     type     cyls      blocks         Megabytes
0        efs      7+101     2835+40960     1+20
1        rawdata  108+202   43795+81920    21+40
6        efs      310+1247  125715+505275  61+247
8        volhdr   0+7       0+2835         0+1
10       entire   0+1558    0+630990       0+308

Use the new partition layout? (no)
```

Note that the partition table now reflects that 4 megabytes have been taken from partition 6 (/usr) and placed in the swap partition. Enter **yes** to accept the new partition table and press **<Return>**. The new partition table is displayed again. Enter **..** at the prompt to move back to the *fx* main menu. Once you see the main menu, enter **exit** to quit *fx*.

Your disk is now repartitioned, but new file systems must be made on the partitions. Boot *inst*(1M) from your distribution media or over the network and choose the **admin** option from the main menu. From the admin menu, choose the **mkfs** option to make new file systems on your *root* and */usr* file systems before you reinstall your software and user files. The commands you use are similar to these:

```
mkfs /dev/dsk/dks0d1s0
mkfs /dev/dsk/dks0d1s6
```

When this is done, boot your system and enter the System Recovery menu of the System Maintenance Menu. From this menu, you should be able to recover your system using the *Backup*(1) or System Manager backup tape you made earlier. If you did not back up your system using one of those two options, you must reinstall the operating system on your disk and then use the utility you used to make the tape to restore your files.

Swap Space

The IRIX operating system uses a portion of the disk as *swap space* for temporarily saving part or all of a user's program when there is not enough physical memory to contain all of the running programs. If you run many very large programs, you might run out of swap space.

Use *swap -l* to monitor swap space use. If you find that you are running out of swap space, two solutions are available: you can add more memory, or you can add more swap space. Adding swap space does not improve the performance of large programs, but it permits them to run successfully.

IRIX allows programs occupying more space than the system limit to run, since each program is only partially loaded into memory at any given time. One of the effects of this policy is that IRIX has to preallocate swap space based on likely future usage, and sometimes this prediction is incorrect. When the swap space is actually needed, IRIX allocates the most convenient available space, *not* the actual space allocated. So the physical allocation is separate from the accounting allocation.

If your system preallocates all your swap space, but the space has not yet been used, it may appear that your system is running out of swap space when it is not. It is possible that your system has simply preallocated the rights to future swap space to existing processes, and no new processes can allocate space due to the strict swap space accounting in this version of IRIX.

In previous versions of IRIX, available swap space accounting was collected, but ignored by default. If a system actually ran out of physical swap space and more swap space was immediately needed, the system made the space available by terminating processes with low priority (such as batch jobs). This strategy is call ''lazy'' accounting. The kernel tunable parameter *availsmem_accounting* could be set to enforce strict swap space accounting. The drawback of lazy accounting is that when IRIX runs out of physical swap space, processes are terminated before they are complete.

In the current version of IRIX, strict swap space accounting is always in effect, but the ability to add both physical and virtual swap space through ordinary system files allows the administrator to add swap space or to effectively turn off strict swap space accounting, without having to either repartition the disk or reconfigure and reboot the system.

If your system is experiencing processes being denied stack growth or new processes due to a stated lack of swap space, and you believe that there is adequate physical space, add the following entry to your */etc/fstab* file:

```
/usr/swap swap swap pri=4,vlength=204800 0 0
```

Then give the command:

```
mkfile -v 0b /usr/swap
```

The file (**/usr/swap**) will be zero-length, so you have added only virtual swap space, and no real swap area. Your kernel should then allow more processes to execute. However, when an attempt is made to access more than the system limit, IRIX swaps the largest running program out of memory.

To determine how much swap space is already configured in your workstation, use the *swap*(1M) command:

```
swap -l
```

If you are running applications that require the system to swap out programs frequently, you may also want to fine-tune the swap area of the disk used by the operating system. For more information on this process, see Chapter 5, "Tuning System Performance."

Swap -s command

The *swap -s* command is a very useful tool for determining if you need to add swap space of some sort. The output of the *swap -s* command looks something like:

```
total: 0 allocated + 64248 reserved = 64248 blocks used,
17400 blocks available
```

Where the fields displayed are as follows (see the *swap(1M)* reference page for more details):

allocated — The number of 512 bytes blocks allocated to private pages (for example, pages that contain data that is in use.)

reserved — The number of 512 byte blocks currently allocated but not yet marked as private pages (the space has been claimed, but is not yet being used.)

blocks used — The number of 512 byte blocks either allocated or reserved (the total number of allocated and reserved blocks)

blocks available — The number of 512 byte blocks available for future reservation and allocation (the total swap shown by the *swap -l* command less the number of blocks used)

Given the following sample *swap -s* output:

```
total: 0 allocated + 34200 reserved = 34200 blocks used,
47448 blocks available
```

It can be seen that 0 swap blocks are in use, 34200 have been reserved, but not used, which leaves 47448 blocks available for reservation. So, at this point in time, the system above is not swapping, but the programs running on the system have requested approximately 17 Megabytes of swap space, just in case they will need to grow.

Note: 10000 blocks is approximately equal to 5 Megabytes

Many applications reserve what is known as virtual swap space. That is, they request more memory than they will ever need to grow into. The actual size of the application is the amount of physical system resources that the application is utilizing. The virtual size of the application is the amount of system resources it is utilizing plus the amount of extra resources requested but not in use. This is the case in the above example; space has been reserved, but is not in use.

Negative swap space

Let's look at another example of *swap -s* output:

```
total: 41920 allocated + 58736 reserved = 100656 blocks
used, -19400 blocks available
```

It may seem worrisome that the swap space available is a negative number. What this means, though, is that some of the allocated/in use pages are located in main memory (RAM). The *swap -s* output does not take main

memory into account. The data that is shown in the negative is actually data that is contained in system memory.

It appears that approximately 20 Megabytes of physical swap space is in use, as shown by the amount of allocated space. Therefore, the system is not out of physical swap space. If there was no more physical swap space, the number of allocated blocks would be very close/the same as to the number of blocks reported by the *swap -l* command. Approximately 30 additional Megabytes of swap space has been requested, shown by the requested field, giving a total of 50 Megabytes requested and/or in use. This appears to leave us with an overrun of 10 Megabytes.

Another way to think of that negative number is that the negative number is the amount of physical swap space minus the number of blocks used (allocated + requested). So, as long as this total is less negative than approximately the amount of physical memory (obtained from the *hinv* command) that you have, you have not overrun your system.

The following example shows *swap -s* output of a system that has most likely come to its swapping limit:

```
total: 76920 allocated + 23736 reserved = 100656 blocks
used, -19400 blocks available
```

Notice that the total numbers are the same, but the number of allocated blocks is much higher. If the *swap -l* in this example were to report 81000 blocks of physical swap space on the system, it is easy to see that there are only 4000 physical blocks that are not in use.

If *swap -s* reports a negative number, increase virtual swap when your system is not near its physical limits. This will allow your system to allocate space to those applications that grab more space than they actually need. To do this you can turn on virtual swapping by executing the following commands:

```
su
chkconfig vswap on
/etc/init.d/swap start
```

This will allocate more swap space, or space that can be reserved, but not allocated. Please see the */etc/init.d/swap* file and the *swap(1M)* reference page for more information.

If virtual swapping is already *chkconfig*'d on or if the number of allocated blocks is approaching the number of blocks reported by the *swap -l* command, the only way to remedy the situation would be to add more physical memory or swap space. Please see the *swap(1M)* reference page for more information regarding adding swap space (whether through another disk partition or a swap file).

Increasing Swap Space on a One-Disk System

Suppose you don't have the luxury of a multiple-disk system. This section explains how to increase the size of the swap partition on a single disk. You can increase your available swap space by repartitioning your disk, as described earlier in this chapter, or you can add space with the *swap*(1M) command.

The *swap*(1M) command allows you to designate a portion of any disk partition as additional swap space. You can add swap space at any time and delete the new swap space when you no longer need it. There are several options available with this command, and the command is described completely in the *swap*(1M) reference page, but the most convenient method to use is to specify a normal system file as additional swap space.

To specify a file as additional swap space, you first create an empty file of appropriate size with the *mkfile*(1M) command. For example, if you wish to add 10 megabytes of swap space to your system, and you wish that space to be taken from the */usr* file system, use the following *mkfile* command:

```
mkfile -v 10m /usr/tmp.O/moreswap
```

In this command, the **-v** option directs *mkfile* to be verbose in its output to you, which means that you see the following message as a confirmation that the file has been created:

```
/usr/tmp.O/moreswap 10485760 bytes
```

If you do not specify the **-v** option, *mkfile* does its work silently. The second field in the *mkfile* command is the size of the file. In this case, **10m** specifies a file that is 10 megabytes in size. You can use **b**, **k**, or **m** as a suffix to the **size** argument to indicate that the size number is in bytes, kilobytes, or megabytes, respectively. For example, the following commands all produce files of 10 megabytes:

```
mkfile -v 10485760b /usr/tmp.O/moreswap
mkfile -v 10240k /usr/tmp.O/moreswap
mkfile -v 10m /usr/tmp.O/moreswap
```

Once your file is created, you can use the *swap* command to add it as additional swap space on the system. When you make your file, be certain that the file resides in the file system from which you wish to take the space. The */usr/tmp*.O directory is a good place to use if you wish to use space from the */usr* file system. Typically */usr* will have more available space than the *root* file system (/). Note, however, that you can also use file systems mounted remotely via NFS. Complete information on using remote mounted file systems for swap space is available in the *swap*(1M) reference page.

To begin using your new file as swap space, give the following command:

```
/sbin/swap -a /usr/tmp.O/moreswap
```

The **-a** option indicates that the named file is to be added as swap space immediately. To check your new swap space, use the command:

```
swap -l
```

This command lists all current swap spaces and their status.

To make your new swap file permanent (automatically added at boot time), add the following line to your */etc/fstab* file:

```
/usr/tmp.O/moreswap    swap    swap   pri=3  0  0
```

Note that if you create a swap file in the */tmp* directory of your *root* file system, the file is removed when the system is booted. The */usr/tmp*.O directory of the */usr* file system is not cleaned at boot time, and is therefore a better choice for the location of swap files. If you wish to create your swap files in the *root* file system, first create a */swap* directory, and then create your swap files within that directory.

Increasing Swap Space on a Multidisk System

Adding more swap space to a multidisk system can be done just as if you were adding space on a single disk system. You can always use the *mkfile*(1)

and *swap*(1M) commands to add a swap file to your system. However, if you wish to add dedicated swap space in a new disk partition, follow the instructions below.

To double the default amount of swap space, you can use another disk drive as follows:

```
Partition/slice
    0        Temporary space (mount as /tmp)
    1        Swap space
    6        usr2
```

Note that the operating system continually writes onto the partition that is used as swap space, completely destroying any data that might exist there. Be sure that the swap partition does not overlap any user file system partitions. Verify the size of the swap partition in blocks.

Once you choose a partition, create the file */etc/init.d/addswap* to add this partition permanently as a swap partition. Place a line of the following form in the file:

```
swap -a /dev/dsk/devicename 0 length
```

The argument *devicename* is the device name where the swap partition is located (such as *ips0d1s1*), and *length* is on blocks. Once you create this file, use the *chmod*(1) command to enable execute permission on the file. The command is:

```
chmod +x addswap
```

Next, create a symbolic link to the new file with the command:

```
ln -s ./addswap ../rc2.d/S59addswap
```

The */etc/rc2.d* directory controls the system activities that take place when the system boots into multiuser mode (run level 2). The "S" at the beginning of the symbolic linkfile that you created indicates that the commands in the file should be *started* when the system initiates this run level. Symbolic link files that begin with the letter "K" indicate that the commands described in the file should be *killed*. The number following the S or K at the beginning of the linkfile name indicates the sequence in which the commands are executed.

You can also modify the file */etc/fstab* to document (in the form of a comment) that the chosen partition is being used as a swap partition.

Logical Volumes and Disk Striping

The previous section discussed the use of partitions, which cause one disk drive to behave as if it were several smaller drives. Several partitions may be built up into one logical disk drive through the use of *logical volumes.* A logical volume might include partitions from several physical disk drives and, thus, be larger than any of your physical disks.

These logical volumes behave like regular disk partitions. File systems can be created on them just as on a regular disk, and then mounted and used in the normal way. The only difference is that they can be built from several partitions and across more than one physical disk device. A logical volume disk driver, referred to as *lv*, maps requests on logical volume devices onto the underlying physical devices, in a manner transparent to the user.

The drawback to logical volumes is that all disks must function correctly at all times. If you have a logical volume set up over three disks and one disk goes bad, the information on the other two disks will be unavailable, and must be restored from backups.

The exception is that the root partition must never be a logical volume, since the utilities required for logical volume initialization must reside on it. Also, swap space cannot be configured as a logical volume, since the disk is used as raw space with no file system. Use the *swap -a* command to increase your swap space on a second disk. This procedure is documented above in "Increasing Swap Space on a Multidisk System" on page 242.

You can use logical volumes to extend an existing file system onto a new disk drive, using the *growfs*(1M) command. The *growfs* command is also discussed in "Changing File System Size" on page 302.

IRIX allows for the creation of *striped* logical volumes. Disk storage on a striped volume is allocated alternately among partitions. The *granularity*, or width, of the stripes may optionally be set.

The concept of logical volumes thus adds a layer of abstraction to the concept of the physical disk drive and allows more efficient disk use.

Logical volumes are administered by means of a file defining the volumes known to the system (*/etc/lvtab*) and three utilities: *mklv*, *lvinit*, and *lvck*. These are described in more detail below.

You can create a logical volume by adding an entry to */etc/lvtab* and then running *mklv*. Existing logical volumes are "connected" to the system at boot time by *lvinit*; it is not necessary to run *lvinit* explicitly.

The *lvck* program is a diagnostic tool. You need to run this only if there are problems.

The /etc/lvtab File

The file */etc/lvtab* contains a table of the logical volumes used by IRIX. It is read by the utilities that create, install, and check the consistency of logical volumes. You can modify it with a text editor to add new logical volumes or to change existing ones.

The entries have the form:

volume_device_name:[*volume_name*]:[*options*]:*devs*=*device_pathnames*

The *volume_device_name* is of the form `lv`*n*, where *n* is an integer by default between 0 and 9. The logical volume is accessed through the device special files */dev/dsk/lvn* and */dev/rdsk/lvn*.

The *volume_name* is an arbitrary identifying name for the logical volume. This name is included in the logical volume labels on each of the partitions making up the logical volume. It is then used by utilities to verify that the logical volume on the disks is actually the volume expected by */etc/lvtab*. Any name of up to 80 characters can be used; you should probably choose something that other users can identify. You can leave this field blank, but this is not recommended.

The options are specified with the syntax *option_name=number*, with no spaces surrounding the equal sign. You may specify more than one option, separated by colons. Currently recognized options are **stripes** and **step**.

The **stripes** option creates a logical volume that is striped across its constituent devices. The number of device pathnames must be a multiple of this parameter. This specifies the number of ways the volume storage is striped across its constituent devices. For example, suppose you have a logical volume with 6 constituent devices and a **stripes** parameter of 3. The

logical volume is set up to stripe across the first three devices until they are filled, then to stripe across the second three.

The **step** option further specifies the granularity, or step size, with which the storage is distributed across the components of a striped volume. Granularity is measured in disk blocks. The default step is the device tracksize, which is generally a good value to use.

The device pathnames are listed following any options. They are the block special file pathnames of the devices constituting the logical volume. Device pathnames must be separated by commas. The partitions named must be legal for use as normal data storage, and not as dedicated partitions, such as *swap*.

The mklv(1M) Command

The *mklv*(1M) utility constructs the logical volumes by writing labels for the devices that will make up the volume. The general format of the *mklv*(1M) command is:

`/etc/mklv [-f]` *volume_device_name*

mklv reads the entry in */etc/lvtab* identified by *volume_device_name* and constructs the logical volume described. It labels the devices appropriately, then initializes the logical volume device for use.

An existing logical volume can be extended by adding device pathnames to the */etc/lvtab* entry, then running *mklv* on that entry.

Normally, *mklv* checks all the named devices to see if they are already part of a logical volume or contain a file system. The option **-f** forces *mklv* to skip those checks.

Various errors can arise when trying to create a logical volume. For example, one of the specified disks might be missing, or the new *lvtab* entry might have a typographical error. The man page for *mklv* describes the possible error messages and their meanings.

The lvinit(1M) Command

/etc/lvinit(1M) initializes the logical volume device driver that allows access to disk storage as logical volumes. It works from entries in */etc/lvtab*, as does *mklv.* Its form is:

lvinit [*volume_device_names*]

Without arguments, *lvinit* initializes every logical volume with an entry in */etc/lvtab*. With arguments, it initializes only those named in the argument list. The component devices of the logical volumes to be initialized must have been previously labeled as members of the volume by *mklv.*

lvinit is run automatically at system boot time. It is not normally necessary to invoke it explicitly. It performs sanity checks when initializing the volumes and prints messages describing any error conditions.

If error messages from *lvinit* are seen during system boot, you will want to run *lvck* to obtain more information about the problem.

The lvck(1M) Command

The *lvck*(1M) command checks the consistency of logical volumes by examining the logical volume labels of devices constituting the volumes. Possible errors and the messages from *lvck* that describe them are detailed in the *lvck* man page.

Invoked without parameters, *lvck* checks every logical volume for which there is an entry in */etc/lvtab.*

Invoked with the name of a logical volume device, for example, *lv0*, *lvck* checks only that entry in */etc/lvtab.*

Invoked with the **-d** flag, *lvck* ignores */etc/lvtab* and searches through all disks connected to the system in order to locate all logical volumes present. *lvck* prints a description of each logical volume found in a form resembling an *lvtab* entry. This option facilitates recreation of an *lvtab* for the system, if necessary. You might use this option if, for example, */etc/lvtab* became corrupted or if you somehow lost track of which disks were connected during a system reconfiguration.

Invoked with the device block special file name of a disk device, *lvck* prints any logical volume label that exists for that device, again in a form resembling an *lvtab* entry. Note that this mode of *lvck* is purely informational; no checks are made of any other devices mentioned in the label.

lvck has some repair capabilities. If it determines that the only inconsistency in a logical volume is that a minority of devices have missing or corrupt labels, it is able to restore a consistent logical volume by rewriting good labels. *lvck* queries the user before attempting any repairs on a volume.

Examples of Logical Volumes

Logical volumes can be used to:

- provide storage for a new file system on newly added disks
- allow an existing file system to grow onto newly added disks

The following sections give examples of the above listed uses.

Creating a New File System on Newly Added Disks

Suppose that new disks are added to your system in order to provide storage for a new file system. You can assume that they are connected and initialized correctly.

Decide which partitions of these new disks you want to use for the new file system. (Normally, when adding a new file system like this, you will want to use the whole of all the disks, that is, partition 7 of each disk.)

Decide if you want to make a striped volume. The benefit of striping is improved throughput; however, it does impose some minor restrictions. If you want to stripe, all the drives (or to be exact, the partitions on them that you are using) must be exactly the same size. If you later want to add more drives to the volume, you must add them in units of the striping. That is, if you want to add disks to a 3-way striped volume, you must add them three at a time.

Also, to obtain the best performance benefits of striping, you should arrange to connect the disks you are striping across on different controllers. In this arrangement, there are independent data paths between each disk and the

system. However, a performance improvement of up to 20% can be obtained using SCSI disks striped on the same controller.

Add an entry to */etc/lvtab* containing the pathnames of the new disks that are to be part of the new volume. (See the *lvtab* man page for details of the syntax of *lvtab* entries.) For example:

```
lv0:proj:stripes=2:devs=/dev/dsk/dks0d2s7,/dev/dsk/dks1d0s7
```

In this example, the logical volume named *proj* consists of two partitions from two separate disks. Storage is striped across the two disks. (Note that it is not normally necessary to specify the **step** parameter. An appropriate value will be used automatically.)

Run *mklv* to place the labels on the new disks to identify them as parts of a logical volume:

```
mklv lv0
```

Device names */dev/dsk/lv0* and */dev/rdsk/lv0* have been created. These can now be accessed exactly like any regular disk partition.

To create the new file system, run *mkfs* on */dev/rdsk/lv0* as you would run it on a regular disk:

```
mkfs /dev/rdsk/lv0
```

Then, you can mount */dev/dsk/lv0* exactly as you would mount a file system on a regular disk. You may want to add an entry to */etc/fstab* to mount this file system automatically, for example:

```
/dev/dsk/lv0 /some_directory efs rw,raw=/dev/rdsk/lv0 0 0
```

When creating a striped volume, all the partitions must be the same size. Unfortunately, the default partitioning for drives of similar sizes but from different manufacturers can be slightly different. If this is the case, you will see an error message similar to this from *mklv*:

```
lv0:proj:stripes=2:devs= \
        /dev/dsk/dks0d2s7,  \
        /dev/dsk/dks1d0s7 <INCORRECT PARTITION SIZE>
```

In this case, you need to adjust the partition sizes; see “Changing Hard Disk Partitions” on page 231 for instructions.

Extending an Existing File System

Suppose you have a file system (on a regular disk) that is full, and you would like to have more space available in it.

You need a new disk (or at least a partition on a disk) that you can dedicate to providing this extra space. This new partition must not, of course, be currently used for anything or contain any valuable data. Perform the following steps:

1. Make a backup of the file system you are going to extend.
2. Add an entry to */etc/lvtab* for a new logical volume. The first device in this volume should be the device on which the existing file system resides. This is followed in the description by the new device you are adding. For example:

   ```
   lv2:new development volume:devs=/dev/dsk/ips0d1s7,   /dev/
   dsk/ips0d2s7
   ```

3. In this example, */dev/dsk/ips0d1s7* is the disk on which the existing file system resides, and */dev/dsk/ips0d2s7* is the added disk. When using a logical volume to extend an existing file system, the logical volume may *not* be striped. Before proceeding further, you must unmount the file system.
4. Run *mklv* to place the labels on the disks that identify them as parts of a logical volume:

   ```
   mklv lv2
   ```

5. *mklv* mentions that one of the devices contains a file system and asks if you want to proceed. This is a safety measure to help avoid inadvertently performing logical volume operations on disks that do contain important information: answer **y**

   You find that device names */dev/dsk/lv2* and */dev/rdsk/lv2* have been created.

6. Extend the file system to take advantage of the newly added space. This is done with *growfs*. For this example, type:

   ```
   growfs /dev/rdsk/lv2
   ```

7. This expands the file system into the new space while leaving its existing contents intact. From now on, you can access this file system using the logical volume device rather than the original disk device. So you might want to run *fsck* on */dev/rdsk/lv2* (in the example) to verify that your file system is valid (and has indeed increased in size).
8. Mount the logical volume device (*/dev/dsk/lv2*) rather than the original disk device; you will need to update your */etc/fstab* to reflect this. Don't forget to change *raw* also.

It is worth noting that you can repeat this expansion process indefinitely. You can always add a new disk, add its name to the *lvtab* entry, and then rerun *mklv* and *growfs* to further expand the file system.

The Bad-Block Handling Feature

Through the bad-block handling feature of the *fx*(1M) command, IRIX provides mechanisms for:

- detecting and remembering blocks that are no longer usable
- reminding you that you need to "fix" some remembered bad blocks
- restoring the usability of the disk in spite of the bad blocks that exist

Note that actual bad-block handling is done by the disk controller. The IRIX bad block feature provides a way to tell the controller which blocks are bad and to tell the controller to make use of its bad-block handling capabilities. There is no comparable feature for tapes.

Virtually every disk has bad blocks, many of them detected by the manufacturer with very sensitive analog equipment. These bad blocks are known as the manufacturers bad block list and are also recorded on a sheet of paper attached to the disk or, on SCSI drives, are saved in PROM on the disk. The bad blocks mapped by the manufacturer are not the only bad blocks that will appear on the disk. It is not uncommon for a new defect to appear once a disk is in the field. These new defects are known as *grown defects*.

As long as you treat a disk drive with care, and do not subject it to excessive vibration and shock, blocks on the drive should seldom go bad. However, if

a new bad block does occur, the data stored in the bad block may be lost, and the disk may be unusable in its current state.

The bad-block handling feature lets you work around the bad block and continue to use the good sections of the disk. However, the only way to recover information that is lost if a block goes bad is to retrieve it from a backup.

When Is a Block Bad?

A block is termed " bad" when it cannot reliably store data. This is discovered typically when an attempt is made to read data from that block, and the read fails. Note, though, that a read fail does not always indicate a bad block. A read fail might also mean problems in the format of the disk or a failure in the controller or the hardware. One way to make an initial determination is if the failures are frequent and involve large numbers of files or directories, or if the system only occasionally experiences such problems. A major failure of the controller or hardware generally results in immediate and massive errors, while a bad disk block does not affect the operation of other parts of the disk.

Disk write failures are far less common than read failures. A write failure generally signals a problem with the format of the disk or a more basic failure in the disk or disk controller hardware. While all failures are reported to the system console, the bad-block handling feature cannot distinguish actual bad blocks from format problems or hardware problems. To fix format problems or hardware errors, you need to reformat the disk or get the hardware repaired. In either case, you should contact your service representative. Bad blocks, though, can be mapped out of use by *fx*(1M) without difficulty. Note that you should never need to reformat a SCSI disk. Simply map the bad tracks out using the *fx*(1M) command.

How to Recognize a Bad Block

Certain error messages indicate bad blocks. The following are examples of some of the error messages you may see in the console window or in the file */usr/adm/SYSLOG*:

- Unrecovered media error 101/5/6 (or block) bn 13495
- error dks0d1s0 media error no address mark found in id field

- dks0d1s0 media error;id crc ECC error block #13951
- disk error: ip0d0 error: csr unrecovered 0x29 <err=sector not found
- Bad header
- Hard error {unrecoverable data error}
- Soft error {Soft errors are recoverable data errors a single fx exercise may not locate.}

Bad blocks may also be the cause of file system corruption. If you have a corrupt file system, you may want to run an *fx* exercise on the disk to see if this file system problem was caused by a problem with the disk media. A corrupt file may (rarely) crash the system with a SEGV (segmentation violation).

A segment of your disk is used as virtual memory or swap space. To the CPU, this section of the disk is seen as part of real memory. A disk error in this partition may also crash the system. The following examples are some of the error messages you may see in the console window or in the file */usr/adm/SYSLOG*:

- err msg: ips0d0s6 uncorrectable error 023 chs=273/13/49 process killed due to bad pageread
- panic get pages - io error in swap
- Read error in swap
- swapseg -i/o error in swap

What Makes a Block Unreliable?

A hard disk uses the magnetic properties of the film coating on disk surfaces to record information. The information is recorded by applying a magnetic "charge" to each point on the disk. This charge is similar to the physical changes in the groove of a phonograph record. Just as the variations in the groove of a record cause the phonograph needle to vibrate and reproduce sound, so the magnetic fluctuations on the disk surface cause identical fluctuations in the magnetic field of the disk head. This fluctuation is read by the disk hardware and translated into data. The process of writing on a hard disk is similar, but in reverse. The disk head generates a magnetic field as it passes over the disk, applying that magnetic force to the coating on the disk.

The information on your disk is recorded with a very high density. Because of the density, the significance of small variations in the magnetic properties of the disk coating is magnified. The variations mean that a given portion of the coating may store some magnetic patterns better than other magnetic patterns. Normally, these variations are insignificant compared to the amount of data being recorded or retrieved. When the variations get to the point where the internal error checking can no longer guarantee that what is written will match what will be read, the area of disk is unreliable. If the data pattern that has been written in a particular area of disk matches the "preferences" of that area of the disk coating, the bad block may escape recognition for a while. If the disk is active, however, the bad block will eventually be discovered.

How Are Bad Blocks Fixed?

A small portion of the disk is set aside from the normally accessible portion of the disk. This portion, called the alternate-track area, cannot be reached by normal IRIX system commands and system calls and is used to provide replacement storage for bad blocks.

Most disks come with a few manufacturing defects. Bad blocks detected in the manufacturer's quality control checks are identified on a label when the unit is delivered. The bad-block handling feature provides special software for remembering bad blocks that have been found and for mapping any additional ones that are found. If a substitute block becomes bad, the software even remaps the original bad block to a new substitute block.

Bad Blocks: Questions and Answers

Bad blocks are detected when input/output disk operations fail for several successive attempts. This means that data being input or output are lost, but the system can restore use of the disk by mapping bad blocks to surrogate blocks that are readable.

The following are some often-asked questions about bad blocks:

- **Why doesn't the system try to discover that a given block is bad while the system still has the data in memory?**

Besides the undesirable increase in system size and complexity, severe performance degradation would result. Also, a block can become a bad block after the copy in memory no longer exists.

- **Why doesn't the system periodically test the disk for bad blocks?**

  Reading blocks with their current contents may not show a bad block to be bad. A thorough bit pattern test would take so long that you would never run it, even assuming a thorough test could be devised using ordinary write/read operations. The disk manufacturer already has tested the disk using extensive bit pattern tests and special hardware, so manufacturing defects have been dealt with.

  Also, many blocks go bad only after information is written to them. Therefore, in order to reliably test for bad blocks, the system would have to write over valid data.

- **Why are disks with manufacturing defects used?**

  Almost all disks have some manufacturing defects that lead to bad blocks. Allowing the disks to contain a modest number of manufacturing defects greatly increases the yield, thereby considerably reducing the cost. Many systems, including this one, take advantage of this cost reduction to provide a more powerful system at lower cost.

Mapping Out Bad Blocks With fx(1M)

Fixing bad blocks with *fx*(1M) is a fairly simple matter. To begin, prepare your system for complete reinstallation and recovery. Make a complete backup of all your system and user files with your preferred backup tools. Then follow these steps:

1. Become superuser and invoke *fx*(1M). You can either bring the system down and boot *fx* in standalone mode or you can invoke *fx* from the command line.
2. Once in *fx*, select the disk you wish to scan as you are prompted. You see the following prompts:

```
fx: ''device-name'' = (dksc)
fx: ctlr# = (0)
fx: drive# = (1)
```

If you press **<Return>** at each prompt, you are selecting the default value (displayed in the parentheses) for that prompt. The default values select your root disk. If you wish to select a different disk, for example **dksc(0,2)** type the correct disk information at the prompt before you press **<Return>**. For the purposes of this example, we will use the root disk. You see:

```
...opening dksc(0,1,)
Warning: This disk appears to have mounted file systems
         Don't do anything destructive, unless you are
         sure nothing is really mounted on this disk.
...controller test...OK
Scsi drive type == CDC    94171-9       0104
---please choose one (? for help  .. to quit this menu)---
[exi]t          [d]ebug/        [l]abel/
[b]adblock/     [exe]rcise/     [r]epartition/

fx>
```

3. Select the **exercise** option from this menu. The **badblock** option is used only to display the current bad block table and to add faulty blocks that are not automatically added by the **exercise** option.

   The **exercise** menu gives access to functions intended for surface analysis of the disk to find bad blocks. Only read-only tests are possible in normal (non-expert) mode. Destructive read-write tests are allowed in expert mode. For complete information on expert mode, see the *fx*(1M) reference page; this example is done on normal mode.

   You see the following menu:

   ```
   ---please choose one (? for help  .. to quit this menu)---
   [b]utterfly      [r]andom      [st]op_on_error
   [e]rrlog         [se]quential
   fx/exercise>
   ```

4. Now you choose how your disk will be scanned for bad blocks. For all choices except **random**, the scan is done one cylinder at a time, unless an error is found. If an error is found, the scan is repeated one sector at a time to find the actual block that is bad, except for drive types that require the entire disk track to be mapped (such as ESDI drives).

   For each unrecoverable error that is found, the failing block is added to the bad block list automatically. The number of retries performed by *fx* itself defaults to 3. You can set the number of retries to any number, including 0, using the **-r** option when you invoke *fx*. Most disk drivers,

and some drives themselves, do retries on their own before reporting an error. For most SCSI drives, the number of retries performed may be set by using the **/label/set/param** menu of *fx*.

The **stop_on_error** menu item does not begin a scan. Selecting this menu option tells *fx* that while the scan is happening, *fx* should stop each time it detects an error and ask you if you want to map the bad block. Whether you answer **yes** or **no**, you are then asked if you want to continue exercising. This can be useful when trying to determine how many errors a disk has before you commit yourself to mapping the bad blocks.

The **butterfly** option invokes a test pattern in which successive transfers cause seeks to widely separated areas of the disk. This is intended to stress the head positioning system of the drive, and will sometimes find errors that do not show up in a sequential test. It prompts for the range of disk blocks to exercise, number of scans to do, and a test modifier. Each of the available test patterns may be executed in a number of different modes (**read-only**, **read-cmp**, etc) that are described below.

The **errlog** option prints the total number of read and write errors that have been detected during a preceding exercise, showing both soft and hard errors. If the **-l** option is used, the blocks on which errors occurred are also reported. Soft errors are those errors for which a driver reported an error, but *fx* was able to successfully complete the scan on a retry. Blocks with soft errors are not forwarded to the bad track table automatically, but since a bad block may read successfully and still generate a driver error, it is often useful to compare the soft errors with the hard errors if troubles persist.

The **random** option invokes a test pattern in which the disk location of successive transfers is selected randomly. It is intended to simulate a multiuser load. Like the **butterfly** test, it prompts for range of blocks to exercise, the number of scans, and any modifier you may want. This does random sized scans (from 1 block to the total cylinder size) as well as seeking to random locations on the disk. It is useful for finding problems on drives with seek problems, and for finding errors in the caching logic or hardware.

The **sequential** option invokes a test pattern in which the disk surface is scanned sequentially. As with the butterfly test, it prompts for the range of blocks to exercise, the number of scans, and any modifier you may want.

The **stop_on_error** option toggles; it determines whether *fx* proceeds automatically when errors are detected. The default is to proceed automatically. If **stop_on_error** is set, then you are prompted at each error whether you want to continue or not. If you continue, you are then asked if you want to add the failing block to the bad block list. This can be useful if you want to find all the failing bad blocks but not actually add them to the bad block list.

The **butterfly**, **random**, and **sequential** tests prompt for a modifier that determines the type of transfer that will occur during the test patterns. Possible modifiers are:

`rd-only`

Performs reads only. The value of read data is ignored. The test detects only the success or failure of the read operation.

`rd-cmp`

Causes two reads at each location in the test pattern. The data obtained in the two reads is compared. If there is a difference, the block(s) that differ are considered bad.

`seek`

Causes each block in the test pattern to be read separately. It is used to verify individual sector addressability. (This operation takes a great deal of time.)

Select the operation you wish to use and press **`<Return>`**. Most bad block scans use the **sequential** option to scan the entire disk automatically, and do not use the **stop_on_error** option.

5. As the scan progresses, *fx* displays some numbers and dots on your screen. These can be safely ignored unless you have chosen the **stop_on_error** option, in which case you may need to respond to prompts and make decisions about logging bad blocks. When the scan is completed, you may exit *fx* and resume normal system operation.

For complete technical information on the normal and extended modes of *fx*(1M), see the *fx* reference page.

Using Floppy Disks Under IRIX

There are a number of SCSI floppy disk drives available for use with your system. To install a floppy disk drive on an IRIX system, follow the hardware documentation that is furnished with your floppy drive to connect it to the computer.

If you are adding a floppy drive to a system that does not have one, the software configuration is taken care of automatically when the system boots. When the system boots, if *hinv* indicates that a floppy drive is installed, but there is no link to it through the */dev* special device files, the MAKEDEV program is automatically invoked to add the proper device files.

If you are installing a floppy disk drive after your initial system installation, perform the following steps:

1. Install the hardware.
2. Log in as root (the Superuser) and enter these commands:

   ```
   cd /dev
   ./MAKEDEV floppy
   ```

3. The *MAKEDEV* program creates the appropriate device nodes.

If you have removed a floppy drive and are installing one of a different type, follow these steps:

1. Install the hardware.
2. Log in as the superuser and enter these commands:

   ```
   cd /dev/rdsk
   rm fds*
   ./MAKEDEV floppy
   ```

3. The *MAKEDEV* program creates the appropriate device nodes according to the SCSI controller and drive number of the floppy drive. For example, a 3.5 inch drive configured as drive 2 on SCSI controller 0 would have the device node:

   ```
   /dev/rdsk/fds0d2.3.5
   ```

There are various options for the various different kinds of floppy devices supported. For example, your device node could have any of the following names, depending on which name suits the hardware you are installing:

```
3.5     (720Kb 3.5" Floppy)
3.5hi   (1.44Mb 3.5" Floppy)
3.5.20m (20.1Mb Floptical)
48      (360Kb 5.25" Floppy)
96      (720Kb 5.25" Floppy)
96hi    (1.2Mb 5.25" Floppy
```

4. Use the following command to link your floppy disk device node with a convenient filename for access, typically */dev/floppy*. Substitute the device node information for your floppy installation for the node name used here:

   ```
   ln -s /dev/rdsk/fds0d2.3.5 /dev/floppy
   ```

Using a Floppy Drive With DOS and Macintosh Floppies

The *mediad* daemon automatically determines the format of a floppy disk inserted in your drive and, if it is a DOS or Macintosh floppy, automatically mounts the file system on your default mount directory. Once mounted, you can use typical IRIX commands such as *cd*, *ls*, and *pwd* with the file system. See the *mediad*(1M) reference page for complete information.

Using a Floppy Drive For IRIX File Transfer

You can use a floppy disk like a tape drive for IRIX file transfer. You can use the standard tape archive commands to write files to the floppy disk if it is in DOS format. Use the *mkfp*(1M) command to create the DOS file system on the disk. See the *mkfp* reference page for complete information. You can also use the command version of *fx*(1M) to format your floppy for file transfer use.

When you place files on a floppy disk, it is a good idea to write the format used, or the exact command used, to place the files on the disk. This makes it much easier for you (and others) to retrieve the files from the disk. Also, whenever possible, change directories to the directory that contains the file and place the file on the floppy using a relative pathname, rather than specifying the filename completely.

Also, be aware that using a floppy to transfer files to systems made by other manufacturers may mean that the same tools are not available on the receiving system. The *tar*, *cpio*, and *dd* tools are usually available on all UNIX systems, but it is a good idea to check beforehand.

In the following examples, the floppy device is given as */dev/rdsk/fds0d3.3.5*. Your actual disk device name may be different.

Floppy File Transfer With tar

To place a copy of the file *transfer.file* on a floppy disk with the *tar* command, use the syntax:

```
tar cvf /dev/rdsk/fds0d3.3.5 transfer.file
```

To retrieve the file, use the command:

```
tar xvf /dev/rdsk/fds0d3.3.5 transfer.file
```

To retrieve all files from a *tar* floppy, use the command:

```
tar xvf /dev/rdsk/fds0d3.3.5
```

or for high density disks:

```
tar xvf /dev/rdsk/fds0d3.3.5hi
```

For complete information on *tar* and its options, see the *tar*(1) reference page.

Floppy File Transfer With cpio

To copy files with *cpio*, use the command:

```
ls transfer.file | cpio -oc > /dev/rdsk/fds0d3.3.5
```

To retrieve the file again, use the command:

```
cat /dev/rdsk/fds0d3.3.5 | cpio -i
```

For complete information on *cpio* and its options, see the *cpio*(1) reference page.

Floppy File Transfer With dd

This *dd* command copies a file to the default floppy device:

```
dd if=transfer.file of=/dev/rdsk/fds0d3.3.5 conv=sync
```

The following command extracts the same file:

```
dd if=/dev/rdsk/fds0d3.3.5 of=transfer.file conv=sync
```

Note that *dd* works only with single files. You can use *tar* or *cpio* to create an archive file, though, and then use *dd* to transfer that archive. If you attempt to extract the file on another brand of workstation and you experience an error, try adding the *conv=swab* statement to your extraction command line. For complete information on *dd*(1), see the *dd* reference page.

Tape Devices

Almost all workstations are configured with some sort of tape device for making backup copies of your files. Whether you maintain one system or a network of hundreds of workstations, you will eventually have to use and maintain some form of tape drive.

Adding a Tape Drive

To install a tape drive on an IRIX system, follow the hardware documentation that is furnished with your tape drive. Make sure you carefully follow any instructions regarding drive terminators.

If you are adding a tape drive to a system that does not have one, the software configuration is taken care of automatically when the system boots. When the system boots, if *hinv* indicates that a tape drive is installed, but there is no link to it through the */dev/tape* file, the *MAKEDEV* program is automatically invoked to add the proper device nodes.

If you are installing a tape drive after your initial system installation, perform the following steps:

1. Install the hardware.
2. Log in as the superuser and enter these commands:

```
cd /dev
./MAKEDEV tape
```

The *MAKEDEV* program creates the appropriate device nodes.

If you have removed a tape drive and are installing one of a different type, follow these steps:

1. Install the hardware.
2. Log in as the superuser and enter these commands:

```
cd /dev
rm *tape
./MAKEDEV tape tapelinks
```

The MAKEDEV program creates the appropriate device nodes and links the correct node for the drive to /dev/tape.

MAKEDEV Commands For Tape Drives

The *MAKEDEV* program supports these options for tape drives:

tape	Creates all the *tps* and *xmt* tape devices, then makes links to *tape*, *nrtape*, *tapens*, and *nrtapens* for the first tape drive found, if one exists. It first checks for *xmt*, then for SCSI in reverse target ID order.
qictape	Creates special files for 1/4-inch cartridge tape drives connected to an ISI QIC-O2 tape controller. See *ts*(7M) for details.
magtape	Creates special files for 1/2-inch tape drives connected to a Xylogics Model 772 tape controller. See *xmt*(7M) for details.
links	Creates both disk and tape special files.
tps	Creates special files for SCSI tape drives. See *tps*(7M) for details.
tapelinks	Makes only links to *tape*, *nrtape*, *tapens*, and *nrtapens*. Examine the target *tapelinks* in the script */dev/MAKEDEV* for more information.

Tape Capacities

Tables 7-3 and 7-4 list the maximum tape capacities in megabytes (MB) for the tape formats IRIX supports. Note that these are maximum, not average capacities.

Table 7-3 Cartridge Tape and DAT Capacities

Format	Capacity (max.)
QIC24	60MB (only reads/writes QIC24)
QIC150	150MB with 600XTD and 6150 tapes (reads QIC24, writes QIC120 and QIC150), 120MB with 600A tapes (writes in QIC120 format) and 250MB with 6250 tapes
DAT	1300MB with 60 meter (1 hour) cartridge, 2000MB with 90 meter (1.5 hour) cartridge Uses the DDS (not DataDAT) format
8mm	2093MB with 112 meter (120 min.) P6 (US) cart 2279MB with 122 meter (90 min.) P5 (European) cart.

Note: Almost all DAT drives use DDS format. 8mm tapes are also available in P6 lengths of 15, 30, 60, and 90 minutes for the U.S., and lengths of 15, 30, and 60 minutes for Europe; the P6 cartridge is for NTSC, and the P5 is for PAL. The drive must be jumpered to match the cartridge type.

Table 7-4 shows maximum capacities for 9-track tapes. Note that 9-track tape capacities vary more than other types because of block size and tape length issues.

Table 7-4 9-track Tape Capacities

Length:	Reel size:	200 ft. 6”	600 ft. 7”	2400 ft. 10.5”	3600 ft. 10.5”
BPI	BLKSZ				
800	512	1	3	10	15
		8192	1.8	5.5	21
		64K	2	6	23

Table 7-4 9-track Tape Capacities

Length:	Reel size:	200 ft. 6”	600 ft. 7”	2400 ft. 10.5”	3600 ft. 10.5”
1600	512	1.3	4	15	22
		8192	3.5	11	41
		64K	4	12	45
6250	512	3.2	10	37	56
		8192	12	37	145
		64K	15	44	175

Note: 3600-foot tapes use thin tape (1.3 mm). BLKSZ indicates block size in bytes.

Making Tape Drive Links

For more information on making tape drive links, see the *ln*(1) and *mknod*(1M) reference pages.

If you suspect that the tape device has not been properly created or that the links between the low level device name (for example, */dev/mt/tps0d3*) and the symbolic name (for example, */dev/tape*) are not correct, then you may want to run the *MAKEDEV* script with the following command sequence:

```
login root
cd /dev
rm *tape*
./MAKEDEV [links or device-type]
```

Device types can be:

- tape links (to recreate all default tape links)
- tape (for all tape devices)
- fds (for floppy device)

- qictape (for the older QIC-02 tape)
- tps (for kennedy SCSI 1/2″ tape)
- magtape (for Xylogics 1/2″ tape)

Normally, the *./MAKEDEV tapelinks* command is all that is needed and will create links for the following default device names: *nrtape*, *nrtapens*, *tape*, *tapens*.

dump(1M) Update for DAT Tapes

The *dump* command is used to *back up* all files in a file system, or files changed after a certain date to magnetic tape or files.

If you are using the *dump* command to perform an incremental file system dump with a DAT tape drive, use the following recommendation to specify the capacity in kilobytes or megabytes of the DAT tape.

Reduce the 4mm tape-length parameter by 40% for the 60-meter tape, and leave as is for the 90-meter tape (2.0 gigabytes). You may want to trim an additional 5% or 10%, if you want to be conservative.

Troubleshooting Inaccessible Tape Drives

Note: This section does not allow for customized installations and does not address complex multiple tape drive issues. You should take care not to violate your maintenance agreements.

Error Indications

The following are some examples of commands and error messages. This is not an exhaustive list.

- **`tar tvf /dev/nrtape`**

  ```
  tar: /dev/nrtape: No such device
  ```

- **`cpio -itvI /dev/nrtape`**

```
cpio: ERROR: Cannot open </dev/nrtape> for input. No such
device
```

- **`tar t`**

  ```
  tar: archive file /dev/tape does not exist or is a
  regular file
  ```

- **`/usr/etc/restore t`**

  ```
  /dev/tape: No such file or directory
  ```

Checking the Hardware

Use the *hinv*(1) command to see if the operating system recognized the tape drive at boot time. This is one of the most basic and critical tests to check hardware. (An output similar to the following is returned with the *hinv* command):

```
Iris Audio Processor: version A2 revision 4.1.0
1 100 MHZ IP22 Processor
FPU: MIPS R4010 Floating Point Chip Revision: 0.0
CPU: MIPS R4000 Processor Chip Revision: 3.0
On-board serial ports: 2
On-board bi-directional parallel port
Data cache size: 8 Kbytes
Instruction cache size: 8 Kbytes
Secondary unified instruction/data cache size: 1 Mbyte
Main memory size: 64 Mbytes
Integral Ethernet: ec0, version 1
Integral SCSI controller 0: Version WD33C93B, revision D
CDROM: unit 4 on SCSI controller 0
Disk drive: unit 1 on SCSI controller 0
Graphics board: Indy 24-bit
Vino video: unit 0, revision 0, Indycam connected
```

If *hinv* does not report an attached tape drive, then your operating system will not be able to use it. You need to check the installation of the hardware. What you can do at this time depends on what you can do with your computer within the confines of your personal ability and your maintenance support agreements.

Simple hardware checks would be:

- If the tape drive is an external unit, does it have power? Simply powering it up will not cause it to be seen by the computer. The system must be shutdown, power cycled, then rebooted.
- During boot up, do you see the access light on the tape drive light up at all? If it doesn't flash at all, chances are the operating system is still not seeing the drive.
- Is the SCSI cabling and termination correct? If visual inspection shows nothing obvious, try resetting the connectors. Any movement of hardware or cabling must be done with the machine powered off.

If nothing that you do here causes *hinv*(1M) to report the tape drive, then the most likely problem is faulty hardware. Contact your support provider.

Checking the Software

If you are reasonably sure the tape drive is correctly installed on the computer, but your software does not seem to be able to use it, a possible cause for the problem is that the tape device's scsi address changes when other scsi devices are added to your system.

The system assumes that if */dev/nrtape* exists and appears to be a tape drive of some kind, then it doesn't need to remake the default tape drive links of */dev/tape*, */dev/nrtape*, etc. It also assumes that the first tape drive that it finds will be the main tape drive. It searches for devices starting at the highest SCSI id numbers, so the tape device on SCSI ID 7 will get the default links before a tape device on SCSI ID 3.

The default tape drive for most commands is */dev/tape*. If the tape drive installation proceeded correctly, you should have at least */dev/tape* and */dev/nrtape* special device files. You may have several others, depending on the type of tape drive.

A *mt* command can be used to confirm that */dev/tape* exists and that the tape drive is responding. An output similar to the following from the *mt status* command confirms that:

```
Controller: SCSI
Device: ARCHIVE: Python 25601-XXX2.63
Status: 0x20262
Drive type: DAT
Media : READY, writable, at BOT
```

The following output means that you have another process accessing the drive right now:

```
/dev/nrtape: Device or resource busy
```

The following output appears when a special device file does not exist:

```
/dev/nrtape: No such file or directory
```

The output when a device file exists, but no hardware is responding at that address, is:

```
/dev/nrtape: No such device
```

If the hardware appears to be present, but */dev/tape* does not appear to be valid, you should confirm the file links. Take the device unit number from *hinv* output:

```
Tape drive: unit 3 on SCSI controller 0: DAT
```

In this example the device unit number is 3 (This is likely to be different on your machine). Use the following *ls* command to confirm that */dev/tape* is linked to the correct device (change the numeral 3 to the correct numeral for your drive):

```
ls -l /dev/tape /dev/mt/tps0d3*
crw-rw-rw- 2 root sys 23, 96 Sep 21 11:11 /dev/mt/tps0d3
crw-rw-rw- 2 root sys 23, 97 Jun 20 05:55 /dev/mt/tps0d3nr
crw-rw-rw- 2 root sys 23, 99 Jul 8 09:57 /dev/mt/tps0d3nrns
crw-rw-rw- 2 root sys 23,103 Jun 20 05:55 /dev/mt/tps0d3nrnsv
crw-rw-rw- 2 root sys 23,101 Jun 20 05:55 /dev/mt/tps0d3nrv
crw-rw-rw- 2 root sys 23, 98 Jun 20 05:55 /dev/mt/tps0d3ns
crw-rw-rw- 2 root sys 23,102 Jun 20 05:55 /dev/mt/tps0d3nsv
crw-rw-rw- 2 root sys 23,100 Jun 20 05:55 /dev/mt/tps0d3v
crw-rw-rw- 1 root sys 23,102 Jun 23 09:19 /dev/tape
```

The major and minor device numbers are the key, here. They are the 2 numbers separated by a comma:

```
crw-rw-rw- 1 root sys 23,102 Jun 23 09:19 /dev/tape
```

Match these numbers with one of the lines from */dev/mt*. In our example, it should match to:

```
crw-rw-rw- 2 root sys 23,102 Jun 20 05:55 /dev/mt/tps0d3nsv
```

Compare the major and minor device numbers that are reported with */dev/tape* and the ones reported for */dev/mt/tps0dX**. Is there a match? If not, remove */dev/tape* and */dev/nrtape* and run *MAKEDEV* as **root** from the */dev* directory. Give the command:

```
./MAKEDEV tapelinks
```

The *MAKEDEV* command can be very verbose in describing what it is doing. Your output may differ in the number of devices made and the unit number. Once it is complete, go through these same checks again to be sure of success.

The *MAKEDEV* command does not let you choose which tape device to link to. You will need to make the links by hand if MAKEDEV does not default to the drive that you wish to use.

This covers the basic problems that administrators experience regarding missing tape drives. See the following reference pages for more information on the commands used in this section: *mt*(1), *ls*(1), *hinv*(1M), and for more technical information about tapes, see *mtio*(7), *tps*(7M), or *xmt*(7).

Troubleshooting Tape Read Errors

Often there is a quick and simple fix for an error message that originates either due to a tape drive unit malfunction or the tape itself. Both recoverable and unrecoverable errors can be caused by something as basic as a dirty read/write head, a poorly tensioned tape, or a dropout even which is a physical bad spot on the tape. An EOT message can also mean that there is no data on the tape.

The following information covers some of the basic tape maintenance/performance functions that should be considered as factors that could either prevent future error conditions from occurring or to aid you in recovering from an existing error message:

- Be sure your read/write head is clean.
- Use the *hinv* command to determine which tape drive type is connected to your system.
- Use the *mt stat* command to verify the status of the tape drive and the media.

- Use the *mt ret* command before read or write operation.

1/2-inch Tape Drives

The following sections offer information on the two most popular 1/2 inch tape drives used with Silicon Graphics systems.

Switch Settings for the Kennedy 1/2-inch SCSI Tape Drive

There are two DIP switch banks located on the rearmost board in the small card cage at the rear of the Kennedy drive.

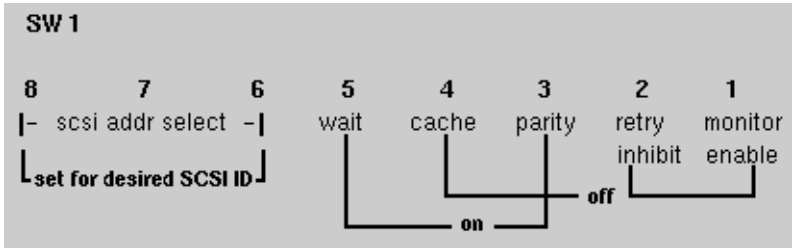


Figure 7-1 Kennedy Dipswitch Bank 1

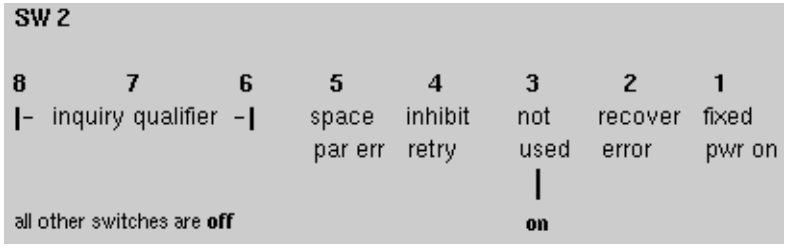


Figure 7-2 Kennedy Dipswitch Bank 2

Note: This applies only to 2 of the 4 SCSI controller boards. The other 2, including the current versions, are configured from the front panel.

1/2-inch Reel-To-Reel Tape Drive Cleaning Process

The purpose of this section is to provide you with a recommended process for cleaning your tape drive and information on how often the drive should be cleaned. The information in this article concerns 1/2-inch reel-to-reel tape drives.

Normal tape drive usage is for system backup and restoration purposes, and the transportation of data using the tape drive device as a common medium. To ensure data integrity, it is important to clean your tape drive on a regular basis. The process of removing the accumulation of oxide and/or dirt from the Erase/Write/Read head surface and transport system will help to provide you with continued trouble-free operation of your equipment.

1/2-inch Tape Drive Types:

Cipher 880/890; Kennedy 9660, 9610

Tools Needed:

Lint-free, nonabrasive cloths or cotton swabs, 90% or higher isopropyl alcohol or Freon-TF, and mild soapy water.

Note: Never clean any plastic or rubber component (for example, the tape guide) in the tape path with 90% or higher isopropyl alcohol. Doing so will degrade the composition of the component.

If Freon-TF has been banned from use in your company, you can use the 90% isopropyl alcohol. However, you must wipe the alcohol residue off with a swab and water.

A cleaning kit for the Cipher 880/890 can be purchased. See your local sales representative for more information.

Cipher Tape Drive Cleaning Process

The drive should be cleaned after every 4 hours of tape movement operation.

The components of the Cipher Tape Drive can be cleaned by placing the drive in the service access position. This is done by extending the unit fully on the mounting slides and opening the top plate casting. To assure safety in the open position, a cover stay pin on the side of the top cover is provided to be inserted in the hole in the chassis.

Warning: If the tape drive is located in a rack, be sure to extend the antitip legs at the base of the rack.

Follow the cleaning method for each part listed:

Tachometer roller

Use a swab moistened with Freon-TF. Gently wipe the entire roller surface. The roller can be rotated by manually turning the take-up hub slowly.

Take-up hub — Use a swab moistened with Freon-TF. Rotate the hub manually while gently wiping the tape wrapping surface.

Roller guides — Use a swab moistened with Freon-TF. Rotate each roller and gently wipe the tape contact surface and flanges or washers.

Reel hub pads — Use a swab moistened with Freon-TF. Wipe the contact surface of each pad and remove any debris around the pad.

Head — Use a swab moistened with Freon-TF. Wipe the entire face of the head, paying particular attention to the recessed areas.

Caution: Rough or abrasive materials can scratch sensitive surfaces of the head resulting in permanent damage. Other cleaners, such as alcohol-based types, can cause read/write errors.

Tape cleaner — Use a swab moistened with Freon-TF. Wipe each blade along its length. Remove accumulated oxides from the recessed area between the blades.

Front panel/door

Use a cloth moistened with mild soapy water.

Top plate casting

Use a cloth moistened with mild soapy water. Wipe away the oxide dust in the tape path area. Be careful not to get dirt on the head, rollers, and other similar parts.

Filter — Locate and remove the filter from inside the air duct opening at the lower left of the front panel. Clean the filter with low-pressure compressed air, or vacuum, in the opposite direction of airflow and reinstall.

Kennedy Tape Drive Cleaning Process

The drive should be cleaned after every 4 hours of operation.

Components of the Kennedy Tape Drive can be cleaned by placing the drive in the service position. This is done by extending the unit fully on the mounting slides and opening the dust cover. To open the dust cover, turn the two holding screws one-fourth of a turn counterclockwise. To secure the dust cover in the open position, position the hole in the autolocking support bar onto the pin located on the side of the chassis. To expose the tape path components, lift the tape path dust cover next to the vacuum hub assembly.

Note: If the tape drive is located in a rack, be sure to extend the anti-tip legs at the base of the rack.

Erase/write/read head — Use a clean lint-free cloth or cotton swab dampened with 90% isopropyl alcohol or Freon-TF. Wipe the head with firm but gentle vertical strokes. Pay particular attention to the recessed areas of the head.

Note: Using non-recommended cleaning fluids or excessive amounts of the recommended cleaning fluid can damage the tape drive. Cleaning fluids allowed to run into the bearings will break down the lubricant.

Tape path cleaning

Clean tape guides, rollers (except capstan roller), and the tape cleaners with a cotton swab dampened with 90% isopropyl alcohol. Clean the capstan roller with a cotton swab dampened with water or, if it is excessively dirty, dampen it with mild soapy water. Dry thoroughly with a clean lint-free cloth.

Warning: Never clean any plastic or rubber component (for example, the tape guide) in the tape path with 90% isopropyl alcohol. Doing so will degrade the composition of the component.

Reel locking fingers

With no tape reel on the supply hub, clean the rubber pads on the fingers with a clean cotton swab or cloth dampened with water or if it is excessively dirty, with mild soapy water.

Caution: Do not lubricate the bearings.

8mm and 4mm Tape Drives

The following section provides useful information for administrators of 8 and 4 millimeter tape drives.

Exabyte 8mm Cartridge Tape Media Specifications

The following table lists the various cartridge sizes and tape lengths available for the Exabyte 8mm tape drive.

Table 7-5 Exabyte 8mm cartridge tape media specifications

Cartridge Size	Tape Length	Formatted Capacity
256	15m	291 MB
512	28m	583 MB
1024	54m	1166 MB
1536	80m	1750 MB
2048	106m	2332 MB

Manufacturers: Exabyte; Sony Metal MP 120 P6-120MP

Tape Availability: Exabyte, (303)442-4333; Silicon Graphics

8mm and 4mm Tape Drive Cleaning Process

The purpose of this section is to provide you with a recommended process for cleaning your tape drive, including the frequency in which the drive should be cleaned.

Normal tape drive usage is for system backup, restore purposes, and the transportation of data using the tape drive device as a common medium. In order to ensure reliable data integrity when using your tape drive, it is important to clean your drive on a regular basis. The process of removing the accumulation of oxide and/or dirt from the erase/write/read head surface

and transport system will help ensure continued trouble-free operation of your equipment.

Tape Drives:

- 8mm Exabyte Tape Drive
- 4mm Archive DAT Tape Drive

A cleaning kit for these tape drives should be purchased. See your local sales representative for more information.

The unit should be cleaned when enough data has been written to fill four tapes.

Using the cleaning kit, perform the following steps:

1. Prepare the tape drive to be cleaned by applying power to the unit. When the power-up cycle is complete, open the door and remove any data cartridges in the unit. Leave the door open.
2. Place the cleaning cartridge into the drive and close the door.

   The remainder of the cleaning process is automatically performed by the tape drive. When the cleaning process is complete, the cleaning cartridge is automatically unloaded and ejected from the drive. The average cleaning cycle is 15 seconds.
3. On the cartridge label, record the date the cleaning was performed, then store the cleaning cartridge for future use.

   Do not store the cleaning cartridge if the cleaning dates have filled the cartridge label. Discard the cleaning cartridge and store a new cleaning cartridge for future use.

   If the cleaning cartridge is ejected from the tape drive without performing a cleaning cycle (before 15 seconds), the cleaning cartridge has reached the end of its useful life and should be discarded.

   Do not rewind and reuse the cleaning cartridge.

QIC Tape Drives

The following section provides information useful to the administrators of systems with Quarter-Inch Cartridge (QIC) tape drives.

The following terms are defined for QIC tapes:

QIC	Quarter Inch Cartridge
QIC-02	Host interface standard. (ts(7) driver)
QIC-11	Recording format. Used on Sun Workstations with 60Mb drives. Cannot be read on Silicon Graphics systems.
QIC-24	Recording format: 9 tracks with a typical track width of .0135 inch. Density is 8000 bpi. Typical capacity is 60 megabytes with 6.6 megabytes per track.
QIC-120	Recording format: 15 tracks with a typical track width of .0065 inch. Density is 10,000 bpi (NRZI Recording Mode). Typical capacity is 120 megabytes, with approximately 8+ megabytes per track.
QIC-150	Recording format used on current SGI drives. Uses 18 tracks.

Note: It is important to use actual QIC designations here, since many low density drives can write (and read) in both QIC 24 and QIC11. Typically, none of the QIC150 drives can read QIC11.

QIC150 drives can write in both QIC150 (using DC6150 or DC600XTD; the name changed to the first recently), *or* in QIC120, if the tape is a 600A-style tape. Typically, QIC150 drives cannot write to QIC24.

Also note that the word *format* is misleading; there is no formatting on QIC tapes (some variants do require formatting, but Silicon Graphics does not support them). Format actually refers to the pattern of data blocks. Tapes have a cartridge type, and they are written in the format correct for that type. The type is determined by the hole pattern in the tape (preceding Beginning Of Tape, or BOT). To confuse things further, tapes written on QIC150 drives have a reference burst (magnetic pattern) written at the beginning of the drive.

The noise you often hear when you first try to read (on a QIC150 drive) a tape written on a QIC24 drive is the drive trying to figure out how the tape was written, by switching modes and retrying all the possibilities, if it doesn't see BOTH a QIC150 cartridge and the reference burst. Physically, the noise you hear is the serve motor stepping the read/write head over each track.

The difference between a QIC150 and QIC120 (600A) cartridge is in its mechanical tolerances. The QIC150 has tighter tolerances. About the only visible difference is in the pinch roller (next to the rubber drive roller). The QIC150 has a guide slot milled into it, and the 600A does not.

Table 7-6 Low-density QIC Tape Drive Compatibility

Tapes	Read	Write
LD Tapes formatted in LDF	Yes	Yes
LD Tapes formatted in HDF	Process not recommended.	Process not recommended.
HD Tapes formatted in LDF	Yes	Yes
HD Tapes formatted in LDF assuming LDF is QIC 24	Yes	Yes
HD Tapes formatted in HDF	No	Yes (rewrite to LDF)

Table 7-7 High-density QIC Tape Drive Compatibility

Tapes	Read	Write
HD Tapes formatted in LDF	Yes	No
LD Tapes formatted in HDF	Process not recommended.	Process not recommended.

Table 7-7 (continued) High-density QIC Tape Drive Compatibility

Tapes	Read	Write
HD Tapes formatted in LDF	Yes	Yes
HD Tapes formatted in LDF assuming LDF is QIC 24	Yes	No
HD Tapes formatted in HDF	Yes	Yes

Regarding read/write activity for a low density tape formatted in high density, it is not only not recommended, it isn't even possible if the tape is a QIC24 (DC300XL or DC450XL) tape. If the tape is a QIC120 (DC600A) it will work correctly, and there is no reason to recommend against it.

Chapter 8

File System Administration

Chapter 8 describes the IRIX file system environment. In this chapter, you learn how IRIX approaches the complex task of keeping track of the thousands of files that exist on the average system. The specific tasks that you perform that are covered here are:

- *Using fsck(1M) to maintain filesystem integrity*
- *Steps to create, remove, name, and otherwise manipulate file systems.*
- *Information about creating and using logical volumes to create file systems that span disk partitions and physical disk drives.*

Chapter 8

File System Administration

The *file system* is the structure by which files and directories are organized in the IRIX system. The file system is also the logical layout that is placed on the hard disk to allow the CPU easier access to data. It is extremely important to maintain file systems properly, in addition to backing up the data they contain. Failure to do so might result in loss of valuable system and user information.

This chapter contains:

- An overview of the IRIX Extent File System (EFS). See "IRIX File System Overview" on page 284.
- An overview of other kinds of file systems used with IRIX, See "Kinds of File Systems" on page 284.
- Steps to administer and maintain the file system. See "Maintaining File Systems" on page 289.
- Steps to create, remove, name, and otherwise manipulate file systems. See "Making New File Systems" on page 299, "Changing File System Size" on page 302, and "Naming a File System" on page 305.
- Information about *logical volumes* and directions for using them to create file systems that span disk partitions and physical disk drives. See "Changing File System Size" on page 302.
- Information about the file system as a functional data structure. See "Basic File System Parameters" on page 284.

Even if you are familiar with the basic concepts of the UNIX file system, you should read through the overview in the next section. The IRIX Extent File System is slightly different internally from other UNIX file systems.

IRIX File System Overview

The basic IRIX file system contains an enhancement to the standard UNIX file system called *extents*, and thus is called the IRIX Extent File System (EFS). Extents, and the various file systems available with IRIX, are described in the next sections.

Basic File System Parameters

The following are some basic parameters of the Extent File System, and a list of what you can and can't do with a file system:

- The maximum size of an IRIX file system is about 8 GB (gigabytes)
- The maximum size of a single file is about 2 GB
- You can increase the size of an existing file system (up to the maximum size of 8 GB) and increase the number of *index nodes*, more commonly known as *inodes*.
- Files and directories cannot span file systems.
- You cannot use *hard links* across file systems.
- You can use *symbolic links* across file systems.
- Only one file system can reside on a disk partition.
- You can join two or more disk partitions to create a *logical volume*.
- Only one file system can reside on a logical volume.

Kinds of File Systems

Several types of file systems are available with the IRIX system:

- Extent File System (EFS). This is the standard IRIX file system and is described in the next section.
- Network File System (NFS). NFS file systems are available if you are using the optional NFS software. NFS file systems are exported from one host and are mounted on other hosts across the network.

On the host where the file systems reside, they are treated just like any other file system. The only special feature of these file systems is that they are exported for mounting on other workstations. Exporting NFS file systems is done with the *exportfs*(1M) command. This type of file system is described in the *NFS and NIS Administration Guide*, which is shipped with the NFS product.

- Debug file system (dbg). The debug file system provides an interface to running IRIX processes for use by debuggers, such as *dbx*(1). The debug file system is usually mounted on */proc* with a link to */debug*. Note that for convenience, */proc* is not displayed when you list free space with the *df*(1) command.

  Note that the */proc* file system does not consume any disk space and its files cannot be removed. Files residing in */proc* are merely convenient handles for debugging processes. See *dbg*(4) for more information on the debug file system.

- Floppy disk file systems and CD file systems are available through the *mediad*(1M) daemon.

  IRIX supports 720k and 1.44MB FAT (DOS) and 1.44MB HFS (Macintosh) file systems on floppy disks. 800K HFS floppies are not supported. HFS file systems on CD-ROM drives are supported. The ISO9660 standard is supported, and this includes photo CD support. The *mediad* daemon also supports industry-standard music compact disks. For complete information on these file system types, see the *mediad*(1M) reference page.

Extent File System (EFS)

This section describes the IRIX EFS:

- The first block of the file system is not used by IRIX. (An IRIX file system block is 512 bytes.) It is normally a *boot block*, reserved for booting procedures, but IRIX does not use the boot block. This block is unavailable for any use.
- Information about the file system is stored in the second block of the file system (block 1), called the *super-block*. This information includes:
  - the size of the file system, in both physical and logical blocks
  - the read-only flag; if set, the file system is read only

  - the super-block-modified flag; if set, the super-block has been modified
  - the date and time of the last update
  - the file system label (optional)
  - the total number of index nodes (*inodes*) allocated
  - the total number of inodes free
  - the total number of free blocks
  - the starting block number of the free block bitmap

- After the super-block bitmap is a series of *cylinder groups*. Each cylinder group contains both inodes and data blocks.

  An inode is a 128-byte data structure that stores all information about a file, except for its name. The file name is kept in a directory. When a file is created, an inode is allocated to that file.

  There is one inode per file in the file system. The maximum number of files in a file system is limited by the number of inodes in that file system.

  An inode contains:

  - the type and mode of the file; types are *regular file*, *directory, block*, *character*, *FIFO* (also known as a *named pipe*), *symbolic link*, and *UNIX domain socket*; the mode defines the access permissions *read*, *write*, and *execute*.
  - the number of hard links to the file
  - who owns the file (the owner's user-ID number)
  - the group to which the file belongs (group-ID number)
  - the number of bytes in the file
  - an array of up to 12 structures, called *extent descriptors*, that contain pointers to the file data
  - the date and time the file was last accessed
  - the date and time the file was last modified
  - the date and time the file was created

- Extents are the data blocks that make up a file. There are 12 extent addresses in an inode. Extents are of variable length, anywhere from 1 to 248 contiguous blocks.

  An inode contains addresses for 12 extents, which can hold a combined 2976 blocks, or 1,523,712 bytes. If a file is so large that it cannot fit in the 12 extents, each extent is then loaded with the address of up to 248 *indirect* extents. The indirect extents then contain the actual data that makes up the file. Because IRIX uses indirect extents, you can create files up to 2 GB, assuming you have that much disk space available in your file system.
- The last block of the file system is an exact duplicate of the file system super-block. This is a safety precaution that provides a backup of the critical information stored in the super-block.

Floppy and CD File Systems

IRIX allows you to mount and use file systems on floppy disks and on CD-ROM drives. You can use these file systems on your own system, or you can export them via NFS for use on other systems (if you have NFS installed). See the *NFS Administration Guide* for more information on exporting file systems.

The operating instructions for these kinds of file systems are very similar and are covered in detail in the *mediad*(1M) reference page.

IRIX supports the following CD and floppy disk file system formats:

- FAT (MS-DOS)
- HFS (Macintosh)
- ISO 9660
- Photo CD
- High Sierra
- EFS (IRIX File System)
- Music CD format

Floppy Disk File Systems

File systems on floppy disks are controlled by the *mediad*(1M) daemon. *mediad* monitors a given floppy drive, waiting for a disk to be inserted. If your system is running the *objectserver*, floppy disks are mounted on */floppy* if the disk is in FAT (MS-DOS) or HFS (Macintosh) format. If you have more than one floppy drive, additional disks in additional drives are automatically mounted on /floppy2, /floppy3, and so on. If your system is not running the *objectserver*, you must provide a location for the mount point. See the *mediad*(1M) reference page for complete information. When you are through using the floppy file system, issue the *eject*(1) command, and *mediad* will attempt to unmount the file system. If the unmount is successful, it ejects the floppy immediately. Note that only one instance of *mediad* is allowed per system. Two invocations of *mediad* with the same floppy parameter generate an error.

To specify a particular floppy drive, use the appropriate device special file in */dev/rdsk*. High density diskettes can be accessed by using floppy devices with the *hi* suffix on the device special file name.

If you are not running the *objectserver* and you wish to start *mediad* for a high density floppy drive with SCSI identifier 7 and the mount point */floppy*, use this command:

```
mediad -ip /dev/rdsk/fds0d7.3.5hi /floppy
```

You must give instructions for the floppy device to be monitored and a location to mount the file system. You must also have created the mount point and ensured that the directory permissions are set appropriately (777 is usually required for read-write file systems).

CD-ROM File Systems

mediad(1M) also monitors CD-ROM drives, waiting for a disk to be inserted. When a disk is inserted, the file system it contains is mounted if the file system is in EFS, HFS, ISO 9660, or High Sierra format. If your system is running the *objectserver*, the CD-ROM drives are monitored automatically and when a CD containing a valid file system is inserted, it is automatically mounted on */CDROM* (for the first CD-ROM drive), and */CDROM2*, */CDROM3*, and so on for additional drives.

If you are not running the *objectserver* when you invoke *mediad*, you must give instructions for the SCSI device to be monitored and a location to mount the CD-ROM file system. You must also have created the mount point and ensured that the directory permissions are set appropriately (755 is usually adequate for read-only file systems). If you are not running the *objectserver* and you wish to start *mediad* for a CD-ROM drive with SCSI identifier 4 and the mount point */cdrom* with the mount option **ro** (for read-only), issue this command:

```
mediad -o ro -ip /dev/scsi/sc0d410 /cdrom
```

Note that CD-ROM file systems are always read-only. When you are finished using the file system, issue the *eject* command, and *mediad* will attempt to unmount the file system. If the unmount is successful, it ejects the CD. When *mediad* is running, however, any user can unmount and eject a CD with the *eject* command. Only one instance of *mediad* is allowed per system.

Maintaining File Systems

To administer file systems, you need to do the following:

- Monitor the amount of free space and free inodes available.
- If a file system is chronically short of free space, take steps to alleviate the problem, such as removing old files and imposing disk usage quotas.
- Periodically check file systems for data integrity using *fsck*.
- Back up file systems.

The first three tasks are described in this chapter. Information about backing up file systems is found in Chapter 6, “Backing Up and Restoring Files.”

Shell Scripts for File System Administration

Many routine administration jobs can be performed by shell scripts. Here are a few ideas:

- Use a shell script to investigate free blocks and free inodes, and report on file systems whose free space dips below a given threshold.

- Use a shell script to automatically "clean up" files that grow (such as log files).
- Use a shell script to highlight cases of excessive disk use.

All of these scripts can be run automatically by *cron*(1M) and the output sent to you using electronic mail. Typically, these scripts use some combination of *find*(1), *du*(1M), *Mail*(1), and shell commands.

The process accounting system performs many similar functions. If the process accounting system does not meet your needs, examine the scripts in */usr/lib/acct*, such as *ckpacct* and *remove*, for ideas about how to build your own administration scripts.

Checking Free Space and Free Inodes

You can quickly check the amount of free space and free inodes with the *df*(1) command. For example, the command:

```
df
```

produces the following output:

```
File              Type blocks  use     avail %use  Mounted on
/dev/root         efs  31464   25238   6226  80%   /
/dev/usr          efs  491832  452902  38930 92%   /usr
ralph.cbs:/ralph  nfs  463360  409088  54272 88%   /usr/ralph
```

To determine the number of free inodes, use this command:

```
df -i
```

You see a listing similar to the one above, except that it also lists the number of inodes in use, the number of inodes that are free (available), and the percentage of inodes in use.

When a file system is more than about 90-95% full, system performance may degrade, depending on the size of the disk. Therefore, you should monitor the amount of available space and take steps to keep an adequate amount available. The percentage of disk use is non-linear, which means that a larger disk that is 97% full is not burdened as heavily as a smaller disk that is also 97% full.

If it is not possible to significantly reduce the amount of disk space used, and more space is needed for a particular file system, you can change the size of the file system. If you cannot add an additional disk drive and you need to adjust the size of your file systems, you must back them up, remove them, and then remake them using *mkfs*(1M). If you can add another hard disk to the system, you can grow existing file systems onto the new disk using a logical volume and *growfs*(1M). For more information on logical volumes, see "Logical Volumes and Disk Striping" on page 244.

Why Free Space Decreases

The amount of free space on a file system decreases over time for the following reasons:

- People tend to forget about files they no longer use. Outdated files often stay on the system much longer than necessary.
- Some files, particularly log files like */var/adm/SYSLOG*, grow as a result of normal system operations. Normally, *cron* rotates this file once per week so that it does not grow excessively large. (See */var/spool/cron/crontabs/root*.) However, you should check this file periodically just to make sure it is being rotated properly, or when the amount of free disk space has grown small.
- The lost+found directory at the root of each file system may be full. If you log in as *root*, you can check this directory and determine if the files there can be removed.
- Some directories, notably */tmp*, */usr/tmp.O*, and */var/tmp*, accumulate files. These are often copies of files being manipulated by text editors and other programs. Sometimes these temporary files are not removed by the programs that created them.
- The directories */usr/tmp.O*, */var/tmp*, and */var/spool/uucppublic* are public directories; people often use them to store temporary copies of files they are transferring to and from other systems and sites. Unlike */tmp*, they are not cleaned out when the system is rebooted. The site administrator should be even more conscientious about monitoring disk use in these directories.
- WorkSpace users move old files to the dumpster without realizing that such files are not fully deleted from the system.

- *vm* and IRIX *core* files in */var/adm/crash* are accumulating without being removed.
- binary core dumps from crashed application programs are not being removed.

Monitoring Key Files and Directories

Almost any system that is used daily has several key files and directories that grow through normal use. Some examples are shown in Table 8-1.

Table 8-1 Files and Directories That Tend to Grow

File	Use
/etc/wtmp	history of system logins
/var/adm/sulog	history of *su* commands
/var/cron/log	history of actions of *cron*
/tmp	directory for temporary files (*root* file system)
/var/tmp	directory for temporary files
/usr/tmp.O	directory for temporary files

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 the *tail*(1) and *mv*(1) commands:

```
tail -50 /var/adm/sulog > /var/tmp/sulog
mv /var/tmp/sulog /var/adm/sulog
```

This sequence puts the last 50 lines of */var/adm/sulog* into a temporary file, then moves the temporary file to */var/adm/sulog*. This reduces the file to the 50 most recent entries. It is often useful to have these commands performed automatically every week using *cron*(1). For more information on using *cron* to automate your regular tasks, see "Automating Tasks with at(1), batch(1), and cron(1M)" on page 29.

Cleaning Out Temporary Directories

The directory */tmp* and all of its subdirectories are automatically cleaned out every time the system is rebooted. You can control whether or not this happens with the *chkconfig* option **nocleantmp**. By default, **nocleantmp** is off, and thus */tmp* is cleaned.

The directory */var/tmp* is not automatically cleaned out when the system is rebooted. This is a fairly standard practice on IRIX systems. If you wish, you can configure IRIX to automatically clean out */var/tmp* whenever the system is rebooted. Changing this standard policy is a fairly extreme measure, and many people expect that files left in */var/tmp* are not removed when the system is rebooted. The same rules apply to */usr/tmp.O*.

If you must change the policy, this is how to do it:

1. Notify everyone who uses the system that you are changing the standard policy regarding */var/tmp*, and that all files left in */var/tmp* will be removed when the system is rebooted. Send electronic mail and post a message in the */etc/motd* file.

   Give the users at least one week's notice, and longer if possible.

2. Edit the file */etc/init.d/RMTMPFILES*.

3. Find a block of commands in the file that looks something like this:

```
# make /var/tmp exist
if [ ! -d /var/tmp ]
then
        rm -f /var/tmp # remove the directory
        mkdir /var/tmp
fi
```

4. Remove the `fi` statement, then add the following lines:

```
else
        # clean out /var/tmp
        rm -f /var/tmp/*
fi
```

   The complete block of commands should look something like this:

```
# make /var/tmp exist
if [ ! -d /var/tmp ]
then
        rm -f /var/tmp # remove the directory
```

```
        mkdir /var/tmp
else
        # clean out /var/tmp
        rm -f /var/tmp/*
fi
```

5. Save and exit the file.

Do not make this change without warning users well in advance. You can also automate this task by using the *find*(1) command to find files over 7 days old in the temporary directories and remove them. Use the following commands:

```
find /var/tmp -atime 7 -exec rm {} \;
find /tmp -atime 7 -exec rm {} \;
```

See "cron(1M) Command" on page 30 for information on using the *cron* command to automate the process.

Tracking Disk Use

Part of the job of cleaning up file systems is locating and removing files that have not been used recently. The *find*(1) command can locate files that have not been accessed recently.

The *find* program searches for files, starting at a directory named on the command line. It looks for files that match whatever criteria you wish, for example all regular files, all files that end in ".trash," or any file older than a particular date. When it finds a file that matches the criteria, it performs whatever task you specify, such as removing the file, printing the name of the file, changing the file's permissions, and so forth.

For example:

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

In the above example:

/usr	specifies the pathname where *find* is to start.
-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.

`> /usr/tmp/deadfiles &`
directs the output to the temporary file */tmp/deadfiles* and runs in the background. Redirecting the results of the search in a file is a good idea if you expect a large amount of output.

Identifying Large Space Users

Four commands are useful for tracking down accounts that use large amounts of space: *du*(1), *find*(1), *quot*(1M), and *diskusg*(1M).

du displays disk use, in blocks, for files and directories. For example:

```
du /usr
```

This displays the block count for all directories in the *usr* file system.

The *du* command displays disk use in 512-byte blocks. To display disk use in 1024-byte blocks, use the **-k** option. For example:

```
du -k /usr/people/ralph
```

The **-s** option produces a summary of the disk use in a particular directory. For example:

```
du -s /usr/people/alice
```

For a complete description of *du* and its options, see the *du*(1M) reference page.

Use *find* to locate specific files that exceed a given size limit. For example:

```
find /usr -size +10000 -print
```

This example produces a display of the pathnames of all files (and directories) in the *usr* file system that are larger than ten 512-byte blocks.

quot reports the amount of disk usage per user on the file system. You can use the output of this command to inform your users of their disk space usage.

diskusg(1M) is part of the process accounting subsystem that serves the same purpose as *quot*. *diskusg*, though, is typically used as part of general system accounting. This utility generates disk usage information on a per-user basis. *diskusg* prints one line for each user identified in the */etc/passwd* file. Each line contains the user's UID number and login name, and the total number of 512-byte blocks of disk space currently being used by the account. The command:

```
/usr/lib/acct/diskusg /dev/usr
```

produces output in the following format:

UID login_name number_of_blocks

The output of *diskusg* is normally the input to *acctdisk* (see the *acct*(1M) reference page), which generates total disk accounting records that can be merged with other accounting records. For more information on the accounting subsystem, consult the *acct*(4) reference page or Chapter 2, "Operating the System," in this Guide.

Imposing Disk Quotas

If your system is constantly short of disk space and you cannot increase the amount of available space, you may be forced to implement disk quotas. IRIX provides the *quotas* subsystem to automate this process. A limit can be set on the amount of space a user can occupy, and there may be a limit on the number of files (inodes) he can own.This subsystem is described completely in the *quotas*(4) reference page. You can use this system to implement specific disk usage quotas for each user on your system. You may also choose to implement "hard" or "soft" quotas. (Hard quotas are enforced by the system, soft quotas merely remind the user to trim disk usage.)

With soft limits, whenever a user logs in with a usage greater than his soft limit, he or she will be warned (via */bin/login*(1)). When the user exceeds the soft limit, the timer is enabled. Any time the quota drops below the soft limits, the timer is disabled. If the timer is enabled longer than a time period set by the administrators, the particular limit that has been exceeded will be

treated as if the hard limit has been reached, and no more resources will be allocated to the user. The only way to reset this condition is to reduce usage below the quota. Only *root* may set the time limits and this is done on a per file system basis.

Several options are available with the *quotas* subsystem. You can impose limits on some users and not others, some file systems and not others, and on total disk usage per user, total number of files, or size of files. The system is completely configurable. You can also keep track of disk usage through the process accounting system provided under IRIX.

The importance of managing disk quotas carefully cannot be over-emphasized. It is strongly recommended that if disk quotas are imposed, they should be soft quotas, and every attempt should be made to otherwise rectify the situation before removing someone's files. Before using the *quotas*(4) subsystem to enforce disk usage, carefully read the material on disk quotas in Chapter 3, "User Services," in this Guide.

The following steps impose soft disk quotas:

1. Log in as **root**.
2. To enable the quotas subsystem, give the commands:

   ```
   chkconfig quotas on
   chkconfig quotacheck on
   ```

3. Next, a file named *quotas* should be created in the root directory of each file system that is to have a disk quota. This file should be zero length and should be writable only by *root*. The command

   ```
   touch quotas
   ```

   issued as *root* in the root directory of the file system creates an appropriate file.

4. Once the *quotas* files are present in each file system's root directory, you should then establish the quota amounts for individual users. The *edquota*(1M) command can be used to set the limits desired upon each user. For example, to set a limit of 100MB and 100 inodes on the user ID sedgwick, give the following command:

   ```
   edquota sedgwick
   ```

The screen clears, and you are placed in the *vi*(1) editor to edit the user's disk quota. You see:

```
fs /  kbytes(soft=0, hard=0)  inodes(soft=0, hard=0)
```

The file system appears first, in this case the root file system (/). The numeric values for disk space are in kilobytes, not megabytes, so to specify 100 megabytes, you must multiply the number by 1024. The number of inodes should be entered directly. Edit the line to appear as follows:

```
fs / kbytes(soft=102400, hard=0)  inodes(soft=100, hard=0)
```

Save the file and quit the editor when you have entered the correct values. If you leave the value at 0, no limit is imposed. Since we are setting only soft limits in this example, the hard values have not been set.

5. Where a number of users are to be given the same quotas (a common occurrence) the **-p** option to *edquota* will allow this to be accomplished easily. Unless explicitly given a quota, users have no limits set on the amount of disk they can use or the number of files they can create.

6. Once the quotas are set, issue the *quotaon*(1M) command. For quotas to be accurate, this command should be issued on a local file system immediately after the file system has been mounted. The *quotaon* command enables quotas for a particular file system, or with the **-a** option, enables quotas for all file systems indicated in */etc/fstab* as using quotas. See the *fstab*(4) reference page for complete details on the */etc/fstab* file.

   Quotas will be automatically enabled at boot time in the future. The script */etc/init.d/quotas* handles enabling of quotas and uses the *chkconfig*(1M) command to check the **quotas** configuration flag to decide whether or not to enable quotas.

7. If you need to turn quotas off, use the *quotaoff*(1M) command.

8. Periodically, the records retained in the quota file should be checked for consistency with the actual number of blocks and files allocated to the user. Use the *quotacheck*(1M) command to verify compliance. It is not necessary to unmount the file system or disable the quota system to run this command, though on active file systems, slightly inaccurate results may be seen. This command is run automatically at boot time by the /

etc/init.d/quotas script if the **quotacheck** flag has been turned on with *chkconfig*(1M). *quotacheck*(1M) can take a considerable amount of time to execute, so it is convenient to have it done at boot time.

Making New File Systems

Use the *mkfs*(1M) command to make file systems on formatted disks. For information on how to format and partition disks, see "Formatting Disks Using fx" on page 229, and "Repartitioning a Hard Disk" on page 231.

Here are a few definitions to help you:

Partition A section of disk, normally associated with a file system. A file system is made on partition boundaries, so that no file system overlaps another. One example is to think of a disk as a cake, and think of a partition as a slice of the cake. Here's a list of the standard partitions on your system:

```
0: root partition
1: swap partition
6: usr partition
7: entire usable disk partition
8: volume header
10: the entire usable disk + volume header (7 + 8)
```

Raw Device The raw device accesses data on a character by character basis.

Block Device The block device accesses data in blocks which come from a system buffer cache.

With this general terminology we can more thoroughly explain how to create our large file system. The following steps show how to make a new file system on an option disk that is already formatted and contains a single partition:

1. Log in as **root**.
2. Back up all data on the system. You can use either file system backup utilities, such as the backup tool in the System Manager, or file-oriented utilities, such as *tar*(1).

   It is always prudent to back up your data whenever you manipulate your hardware and file systems. If you perform this operation on your system disk, it will destroy all data resident in your */usr* partition. For this reason, the example assumes you are making the file system on a previously unused option disk.

   Before you proceed, verify your backups to be sure they are good.
3. Shut down the system and turn it off. Install the new hard disk and configure it. See the reference information provided with the disk for instructions about switch settings and cabling.
4. After the disk is installed, bring the system up to single-user mode.
5. Verify that the disk is formatted and contains the correct partitions with the *prtvtoc*(1M) command. If the disk is not already formatted, or you need to adjust existing partitions, see "Repartitioning a Hard Disk" on page 231 in this guide.
6. Format the new disk using the *mkfs* command to make a new file system. For example:

   ```
   mkfs /dev/rdsk/dks0d2s7
   ```

   This example constructs a file system on the second disk (d2) attached to the primary SCSI controller (0), and uses the entire usable portion of the disk (s7). You can use either the block interface (dsk) or the character interface (rdsk) to the disk. The character interface is faster.

   In the above example, *mkfs* uses default values for the file system parameters. These parameters are:

   - number of physical (512-byte) disk blocks the file system occupies
   - minimum number of inodes in the file system
   - number of sectors per track of the physical medium

- approximate size of each cylinder group in disk blocks
- boundary, in disk blocks, to which a cylinder group is aligned
- boundary, in disk blocks, to which each cylinder group's inode list is aligned

To determine these parameters, *mkfs* examines the device's volume header and uses that information to calculate the minimum number of inodes and the various alignment boundaries. If you want to use parameters other than the default, you can specify these on the *mkfs* command line. See the *mkfs*(1M) reference page for information about using command line parameters and proto files.

For information about specific disk driver interfaces, see the appropriate reference page, including *dks*(7M) for SCSI, or *ipi*(7M).

7. Create the *mount point* for the device. This is a directory on the system where the new file system will be mounted. For example:

   ```
   mkdir /rsrch
   ```

8. The mount point can be anywhere on the system. Mount the directory with the *mount*(1M) command:

   ```
   mount /dev/dsk/dks0d2s7 /rsrch
   ```

9. Most systems are configured so that file systems are automatically mounted when the system is booted up. Automatic file system mounting is done in the file */etc/rc2*, which is executed when the system comes up to multiuser mode.

   If you do not want to mount file systems when the system is booted, skip this next step.

   Add an entry in the file */etc/fstab* for each new file system. For example:

   ```
   /dev/dsk/dks0d2s7 /rsrch efs rw,raw=/dev/rdsk/dks0d2s7 0 0
   ```

   If you already mounted the file system, as described in the previous step, you can use the *mount* command to determine the appropriate */etc/fstab* entry. For example:

   ```
   mount -p
   ```

   displays all currently mounted file systems, including the new file system. Copy the line that describes the new file system, in this case */rsrch*, to */etc/fstab*.

   See the *fstab*(4) reference page for more information about *fstab* entries.

10. You are finished making file systems. Reboot your system and bring it up to multiuser mode and make sure the new file systems are mounted.

    If you want to make more than one file system, use *mkfs* to create a new file system on each disk partition.

Changing File System Size

There are three ways to change the size of file systems:

1. Add a new disk and mount it as a directory on an existing file system.
2. Change the size of the existing file systems by removing space from one partition and adding it to another. To do this, you must back up your existing data, run *fx*(1M) to repartition the disk, then remake both file systems with *mkfs*.
3. Add another hard disk and grow an existing file system onto that disk with *growfs*(1M).

In the first scenario, you simply add a new disk with a separate file system and create a new mount point for it within your file system. This is generally considered the safest and best way to add space. For example, if your */usr* file system is short of space, add a new disk and mount the new file system on a directory called */usr/work*. Once again see “Formatting Disks Using fx” on page 229 for full information on formatting and partitioning hard disks and making file systems on those disks. Then use the instructions in “Mounting and Unmounting File Systems” on page 305 to mount your file system.

The second method, running *fx* and *mkfs*, has serious drawbacks. It is a great deal of work, and has certain risks. For example, to increase the size of a file system, you must remove space from other file systems. You must be sure that when you are finished changing the size of your file systems, your old data still fits on all the new, smaller file system. Also, resizing your file systems may at best be a stop-gap measure until you can acquire additional disk space. This procedure is documented in “Formatting Disks Using fx” on page 229.

In particular, you should not change the size of the *root* partition by remaking your file systems or by using a logical volume. There are better solutions to space problems on the *root* file system. Alternate solutions to this

sort of problem are presented in "Insufficient Space on the root File System" on page 331.

Growing an existing file system onto an additional disk is another way to increase the available space in that file system. The *growfs* command preserves the existing data on the hard disk and adds space from the new disk on a logical volume. This process is simpler than completely remaking your file systems. The one drawback to growing a file system across disks is that if one disk fails, you cannot recover data from the other disk, even if the other disk still works. If your */usr* file system is a logical volume, you will be unable to boot the system into multiuser mode. For this reason, it is preferable, if possible, to mount an additional disk and file system as a directory on */usr* or the *root* file system.

The following steps show how to grow a fictional */work* file system onto a logical volume.

1. Log in as **root** and print out the following reference pages for use later in the process:
   - *lv*(7M)
   - *lvtab*(4)
   - *lvck*(1M)
   - *lvinit*(1M)
   - *fstab*(4)
2. Back up all data on the system. This is a prudent step to take whenever you are manipulating your hardware and file systems. Verify your backups to make sure they are good before you proceed.
3. Shut down the system. Install the new hard disk and configure it. (If the disk was not purchased from Silicon Graphics, it may need to be formatted with *fx*(1M) for use with IRIX.)
4. Bring up the system to multiuser mode.
5. Establish a *logical volume*. It will span the */work* file system on the existing disk and the partition on the new disk by editing the file */etc/lvtab*. Place an entry in the file for the logical volume. The entry should look something like this:

   ```
   lv0:First Logical Volume:devs=/dev/dsk/dks0d2s7, \
   /dev/dsk/dks0d3s7
   ```

An *lvtab* entry is made up of several fields. Each field is separated by a colon. In the above example:

- The first field is the device name of the logical volume (in this example, the entry is *lv0*).
- The second field is the volume label. This field is optional, but may be useful for utilities to verify the volume associated with the device.
- The third field describes which disk partitions make up the logical volume. The first partition you specify is the existing partition that you want to grow. In this case, the first partition in the list is */dev/dsk/dks0d2s7*, which is the existing */work* file system. The second partition in the logical volume is the new disk.

This example shows a logical volume composed of two disk partitions, but it could be made up of several partitions. The only limit is the maximum size of a file system, 8 GB.

6. Change the entry for */work* in the file */etc/fstab* to read:

   ```
   /dev/dsk/lv0 /work efs rw,raw=/dev/rdsk/lv0 0 0
   ```

7. Shut down the system to single-user mode and make sure the */work* file system is unmounted.

8. Run the *mklv*(1M) command to create the logical volume:

   ```
   mklv -f lv0
   ```

   In this example, the argument **lv0** is the volume device name, which is the first field in the *lvtab* entry for this logical volume.

9. After you set up the logical volume, you can grow the file system into the logical volume. Enter the *growfs* command:

   ```
   growfs /dev/rdsk/lv0
   ```

   If the file system needs to be cleaned, *growfs* prints an error message and stops.

10. Even if *growfs* did not print any errors, it is advisable to run *fsck* on the expanded file system:

    ```
    fsck /dev/rdsk/lv0
    ```

11. You are finished growing the file system. Reboot the system and verify that all data is intact.

For more information about setting up logical volumes, including information about *striping*, see "Logical Volumes and Disk Striping" on page 244 in this guide. Other useful information is contained in the *lvinit*(1M), *lvck*(1M), *mklv*(1M), and *lv*(7M) reference pages. For information about the format of the *lvtab* file see the *lvtab*(4) reference page.

Naming a File System

When you create a file system with *mkfs*, the program assigns a name to the file system. However, the name that *mkfs* assigns is not always as memorable or easy to type as one might like. Therefore, you probably want to create a more mnemonic name for the file system. An IRIX file system is generally named after the highest-level directory in its hierarchy.

Use the *mknod*(1M) command to create a special device node in */dev* for the file system, if you have not already done so.

Find the device numbers for the specific file system. For example:

```
ls -l /dev/dsk/dks0d1s6
brw------ 2 root sys 22, 32 Aug 23 17:08 ips0d1s6
```

In this example, the */usr* file system is on an ESDI disk (*ips*) attached to controller 0, it is on the first disk (*d1*), and it is partition 6 (*s6*). See the appropriate reference page for your disk, for example, *dks*(7M).

Mounting and Unmounting File Systems

To use a file system, it must be *mounted*. The *root* and */usr* file systems are always mounted as part of the boot procedure. This is done in the script */etc/rc2*.

You can mount file systems several ways:

- manually
- automatically when the system is booted
- automatically when the file system is accessed for the first time since the computer was rebooted (NFS file systems only)

Note that if you mount a file system over an existing subdirectory, the original subdirectory and its subtrees will be hidden until the file system is unmounted.

Mounting a File System Manually

To mount a file system manually, use this command:

```
mount /dev/dsk/dks0d1s6 /usr
```

If you have a shorthand name for the file system:

```
mount /dev/usr /usr
```

Mounting a File System on Boot Up

To mount a file system every time the computer is rebooted, place a line similar to this in the file */etc/fstab*:

```
/dev/usr /usr efs rw,raw=/dev/rusr 0 0
```

This line indicates a file system */dev/usr* should be mounted on */usr*. It is an IRIX EFS, and it should be mounted *read/write* so that anyone who has permission can write files in the file system. The *fstab* entry also specifies the name of the raw device, in this case */dev/rusr*.

The last two numbers refer to the frequency in days that the file system should be dumped with the *dump* program and the *fsck* parallel pass number, respectively. For more information on *dump*, see "Backing Up File Systems" on page 177 and for more information on *fsck* parallel passes, see "Further fsck Options" on page 309.

The following details the component fields of each entry in the */etc/fstab* file. Here is another sample entry:

```
/dev/dsk/dks0d2s7 /test efs rw, raw=/dev/rdsk/dks 0 d2s7 0 0
```

The fields in your new line are defined as follows:

/dev/dsk/dks0d2s7
: The block device where the file system is located.

/test	The name of the directory where the file system will be mounted.
efs	The type of file system. In this case, we are using an Extent file system. See the reference page on *efs* for complete information.
rw, raw	These are some of many options available when mounting a file system. In this instance, we are asking that the file system be mounted read-write, so that root and other users can write to it. The raw= option should be the last option in the options list. See the *fstab* reference page for all the options available.
0 0	These two numbers represent the frequency of dump cycle and the *fsck* pass priority. These two numbers must be added after the last option in the options list (raw =). The *fstab* reference page contains additional information.

The machine will *mount* our new file system on */test* every time the machine boots into multi-user mode.

Mounting a File System Automatically

If you have the optional NFS software, you can automatically mount any remote file system whenever it is accessed (for example, by changing directories to the file system with *cd*). The remote file system must be exported with the *exportfs*(1M) command.

For complete information about setting up automounting, including all the available options, see the *automount*(1M) and *exportfs*(1M) reference pages. These utilities are discussed more completely in the *NFS and NIS Administration Guide*.

Unmounting a File System

To unmount a file system, use the *umount*(1M) command:

```
umount /dev/usr
```

You should always unmount a file system before running *fsck* on it.

Checking File Systems with fsck

This section provides a quick overview of the steps to using *fsck*(1M). "Repairing Problems with fsck" on page 310 provides even more detailed information about using *fsck*.

File systems are usually checked for data integrity whenever the system is booted. You should also check the integrity of a file system before you make a complete backup of it; otherwise, you run the risk of backing up an inconsistent file system.

The *fsck* command checks file system consistency. To check a single file system, for example, prior to performing an image backup with the System Manager, follow these steps:

1. Log in as **root**.
2. Depending on which file system you want to check, shut down the system to single-user mode.

   To check a file system, the file system must be unmounted. However, you cannot unmount a file system if it is "busy." A file system is busy if any files are open or active in that file system, or if a user's current working directory is a subdirectory of that file system.

   For example, many daemons, such as */usr/lib/lpsched*, */usr/etc/ypbind*, and */usr/etc/syslogd*, execute from the */usr* file system. The simplest way to make sure the file system is not busy is to bring the system down to single-user mode.
3. If you do not bring the system to single-user mode, unmount the file system with the *umount*(1M) command. For example, to unmount the */usr* file system:

   ```
   umount /dev/usr
   ```
4. Run *fsck*:

   ```
   fsck /dev/rusr
   ```

   As *fsck* runs, it proceeds through a series of steps, or *phases*. You may see an error-free check.

   ```
   fsck: Checking /dev/usr
   ** Phase 1 - Check Blocks and Sizes
   ** Phase 2 - Check Pathnames
   ** Phase 3 - Check Connectivity
   ```

```
** Phase 4 - Check Reference Counts
** Phase 5 - Check Free List
7280 files 491832 blocks 38930 free
```

If there are no errors, you are finished checking the file system.

Mount the file system using *mount*(1M). For example:

```
mount /dev/usr /usr
```

5. If errors are detected in the file system, *fsck* displays an error message. If you encounter file system inconsistencies, proceed to "Repairing Problems with fsck" on page 310.

If you cannot shut down the system and cannot unmount the file system, but you need to perform the check immediately, you can run *fsck* in ''no-write'' mode. The *fsck* program checks the file system, but makes no changes and does not repair inconsistencies.

For example, the following command invokes *fsck* in no-write mode:

```
fsck -n /dev/usr
```

If any inconsistencies are found, they are not repaired. You must run *fsck* again without the **-n** flag to repair any problems. The benefit of this procedure is that you should be able to gauge the severity of the problems with your file system.

Further fsck Options

You may find it convenient to check multiple file systems at once. This is also known as *parallel* checking. The **-m** flag indicates parallel checking. Use the **-m** flag only when working from the */etc/fstab* file.

The **-q** option runs *fsck* in quiet mode. Since the program does not prompt for information, this is essentially the same as using **-y**, though with less verbose output.

For a complete list of options, see the *fsck*(1M) reference page.

dfsck

The *dfsck* utility runs simultaneous (dual) checks on two file systems. This functionality is superseded by the **-m** option to *fsck*, but the program is included with IRIX for backward compatibility.

It is strongly recommended that you use the multiple option, **-m** with *fsck*, instead of *dfsck*. The **-m** option with *fsck* does a much better job of checking file systems, and support for *dfsck* may be eliminated in a future release of IRIX.

Repairing Problems with fsck

This section describes the messages that are produced by each phase of *fsck*, what they mean, and what you should do about each one. The following abbreviations are used in *fsck* error messages:

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

The following sections use these single-letter abbreviations:

B	block number
F	file (or directory) name
I	inode 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

In actual *fsck* output, these abbreviations are replaced by the appropriate numbers.

Initialization Phase

The 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 if there are initialization errors.

General Errors Phase

Two error messages may appear in any phase. Although *fsck* prompts for you to continue checking the file system, it is generally best to regard these errors as fatal. Stop the program and investigate what may have caused the problem.

`CAN NOT READ: BLK B (CONTINUE?)`
: The request to read a specified block number *B* in the file system failed. This error indicates a serious problem, probably a hardware failure. Press **n** to stop *fsck*. Shut down the system to the System Maintenance Menu and run hardware diagnostics on the disk drive and controller.

`CAN NOT WRITE: BLK B (CONTINUE?)`
: The request for writing a specified block number *B* in the file system failed. The disk may be write-protected or there may be a hardware problem. Press **n** to stop *fsck*. Check to make sure the disk is not set to ''read only.'' (Some, though not all, disks have this feature.) If the disk is not write protected, shut down the system to the System Maintenance Menu and run hardware diagnostics on the disk drive and controller.

Phase 1 Check Blocks and Sizes

This phase checks the inode list. It reports error conditions resulting from:

- checking inode types
- setting up the zero-link-count table
- examining inode block numbers for bad or duplicate blocks
- checking inode size
- checking inode format

Phase 1 Error Messages

Phase 1 has three types of error messages:

- information messages
- messages with a CONTINUE? prompt
- messages with a CLEAR? prompt

Phase 1 Meaning of Yes/No Responses

Table 8-2 explains the significance of responses to phase 1 prompts:

Table 8-2 Meaning of *fsck* Phase 1 Responses

Prompt	Response	Meaning
CONTINUE?	n	Terminate the program.
CONTINUE?	y	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.
CLEAR?	n	Ignore the error condition. A "no" response is appropriate only if the user intends to take other measures to fix the problem.
CLEAR?	y	Deallocate inode *I* by zeroing its contents. This may invoke the UNALLOCATED error condition in Phase 2 for each directory entry pointing to this inode.

Phase 1 Error Messages

`UNKNOWN FILE TYPE I=I (CLEAR?)`
: The mode word of the inode *I* suggests that the inode is not a pipe, special character inode, regular inode, directory inode, symbolic link, or socket.

`LINK COUNT TABLE OVERFLOW (CONTINUE?)`
: There is no more room in an internal table for *fsck* containing allocated inodes with a link count of zero.

`B BAD I=I`
: Inode *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 inode *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 50) 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 inode *I*.

`B DUP I=I`
: Inode *I* contains block number *B*, which is already claimed by another inode. This error condition may invoke the EXCESSIVE DUP BLKS error condition in Phase 1 if inode *I* has too many block numbers claimed by other inodes. 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 50) of blocks claimed by other inodes.

`DUP TABLE OVERFLOW (CONTINUE?)`
: There is not more room in an internal table in *fsck* containing duplicate block numbers.

`PARTIALLY ALLOCATED INODE I=I (CLEAR?)`
: Inode *I* is neither allocated nor unallocated.

`RIDICULOUS NUMBER OF EXTENTS (%d) (max allowed %d)`
: The number of extents is larger than the maximum the system can set and is therefore ridiculous.

`ILLEGAL NUMBER OF INDIRECT EXTENTS (%d)`
: The number of extents or pointers to extents (indirect extents) exceeds the number of slots in the inode for describing extents.

`BAD MAGIC IN EXTENT`
: The pointer to an extent contains a "magic number." If this number is invalid, the pointer to the extent is probably corrupt.

`EXTENT OUT OF ORDER`
: An extent's idea of where it is in the file is inconsistent with the extent pointer in relation to other extent pointers.

`ZERO LENGTH EXTENT`
: An extent is zero length.

`ZERO SIZE DIRECTORY`
: It is erroneous for a directory inode to claim a size of zero. The corresponding inode is cleared.

`DIRECTORY SIZE ERROR`
: A directory's size must be an integer number of blocks. The size is recomputed based on its extents.

`DIRECTORY EXTENTS CORRUPTED`
: If the computation of size (above) fails, *fsck* will print this message and ask to clear the inode.

`NUMBER OF EXTENTS TOO LARGE`
: The number of extents or pointers to extents (indirect extents) exceeds the number of slots in the inode for describing extents.

`POSSIBLE DIRECTORY SIZE ERROR`
: The number of blocks in the directory computed from extent pointer lengths is inconsistent with the number computed from the inode size field.

`POSSIBLE FILE SIZE ERROR`
: The number of blocks in the file computed from extent pointer lengths is inconsistent with the number computed from the inode size field. *fsck* gives the option of clearing the inode in this case.

Phase 1B Rescan for More DUPS

When a duplicate block is found in the file system, the file system is rescanned to find the inode that previously claimed that block. When the duplicate block is found, the following information message is printed:

`B DUP I=I`
: Inode *I* contains block number *B*, which is already claimed by another inode. This error condition invokes the BAD/DUP error condition in Phase 2. Inodes with overlapping blocks may be determined by examining this error condition and the DUP error condition in Phase 1.

Phase 2 Check Path Names

This phase traverses the pathname tree, starting at the root directory. *fsck* examines each inode that is being used by a file in a directory of the file system being checked.

Referenced files are marked in order to detect unreferenced files later on. The program also accumulates a count of all links, which it checks against the link counts found in Phase 4, pointing to bad inodes found in Phase 1 and Phase 1B.

Phase 2 reports error conditions resulting from the following:

- root inode mode and status incorrect
- directory inode pointers out of range
- directory entries pointing to bad inodes

Initial Checks

fsck examines the root directory inode first, since this directory is where the search for all pathnames must start.

If the root directory inode is corrupted, or if its type is not *directory, fsck* prints error messages. Generally, if a severe problem exists with the root directory it is impossible to salvage the file system, although *fsck* allows attempts to continue under some circumstances.

Possible error messages caused by problems with the root directory inode include:

`ROOT INODE UNALLOCATED. TERMINATING`
: The root inode points to incorrect information. There is no way to fix this problem, so the program stops.

  If this problem occurs on the *root* file system, you must reinstall IRIX. If it occurs on another file system, you must recreate the file system using *mkfs* and recover files and data from backups.

`ROOT INODE NOT A DIRECTORY. FIX?`
: The root directory inode does not seem to describe a directory. If you enter **<n>**, *fsck* terminates. If you enter **<y>**, *fsck* treats the contents of the inode as a directory even though the inode mode indicates otherwise. If the directory is actually intact, and only the inode mode is incorrectly set, this may recover the directory.

`DUPS/BAD IN ROOT INODE. CONTINUE?`
: Something is wrong with the block addressing information of the *root* directory. If you enter**<n>**, *fsck* terminates. If you enter **<y>**, *fsck* attempts to continue with the check. If some of the *root* directory is still readable, pieces of the files system may be salvaged.

Phase 2 Types of Error Messages

Phase 2 has only one type of error message: messages with a REMOVE? prompt.

Phase 2 Meaning of Yes/No Responses

Table 8-3 describes the significance of responses to Phase 2 prompts:

Table 8-3 Meaning of Phase 2 *fsck* Responses

Prompt	Response	Meaning
REMOVE?	n	Ignore the error condition. A "no" response is appropriate only if the user intends to take other action to fix the problem.
REMOVE?	y	Remove a bad directory entry.

Phase 2 Error Messages

`I OUT OF RANGE I=I NAME=F (REMOVE?)`
A directory entry *F* has an inode number *I* that is greater than the end of the inode list.

`UNALLOCATED I=I OWNER=O MODE=M SIZE=S MTIME=T NAME=F(REMOVE?)`
A directory entry *F* has an inode *I* without allocate mode bits. The owner *O*, mode *M*, size *S*, modify time *T*, and file name *F* are printed. If the file system is not mounted and the **-n** option is not specified, and if the inode that the entry points to is size 0, the entry is removed automatically.

`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 inode *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*, inode *I*. The owner *O*, mode *M*, size *S*, modify time *T*, and file name *F* are printed.

Phase 3 Check Connectivity

Phase 3 of *fsck* locates any unreferenced directories detected in Phase 2 and attempts to reconnect them. It reports error conditions resulting from:

- unreferenced directories

- missing or full *lost+found* directories

Phase 3 Types of Error Messages

Phase 3 has two types of error messages:

- information messages
- messages with a RECONNECT? prompt

Phase 3 Meaning of Yes/No Responses

Table 8-4 explains the significance of responses to Phase 3 prompts:

Table 8-4 Meaning of *fsck* Phase 3 Responses

Prompt	Response	Meaning
RECONNECT	n	Ignore the error condition. This invokes the UNREF error condition in Phase 4. A "no" response is appropriate only if the user intends to take other action to fix the problem.
RECONNECT?	y	Reconnect directory inode *I* to the file system in directory for lost files (lost+found). This may invoke a lost+found error condition if there are problems connecting directory inode I to lost+found. If the link was successful, this invokes CONNECTED information message.

Phase 3 Error Messages

`UNREF DIR I=I OWNER=O MODE=M SIZE=S MTIME=T (RECONNECT?)`
The directory inode *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 inode *I* are printed. The *fsck* program forces the reconnection of a nonempty directory.

`SORRY. NO lost+found DIRECTORY`
No *lost+found* directory is in the root directory of the file system; *fsck* ignores the request to link a directory in *lost+found*. The unreferenced file is removed.

There is nothing you can do at this point, but you should remake the *lost+found* directory as soon as possible.

`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*. The unreferenced file is removed.

There is nothing you can do at this point, but you should clean out the *lost+found* directory as soon as possible.

`DIR I=I1 CONNECTED. PARENT WAS I=I2`

This is an advisory message indicating that a directory inode *I1* was successfully connected to the *lost+found* directory. The parent inode *I2* of the directory inode *I1* is replaced by the inode number of the *lost+found* directory.

Phase 4 Check Reference Counts

This phase checks the link count information seen in Phases 2 and 3 and locates any unreferenced regular files. 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 counts of total free inodes

Phase 4 Types of Error Messages

Phase 4 has five types of error messages:

- information messages
- messages with a RECONNECT? prompt
- messages with a CLEAR? prompt
- messages with an ADJUST? prompt

- messages with a FIX? prompt

Phase 4 Meaning of Yes/No Responses

Table 8-5 describes the significance of responses to Phase 4 prompts:

Table 8-5 Meaning of *fsck* Phase 4 Responses

Prompt	Response	Meaning
RECONNECT?	n	Ignore this error condition. This invokes a CLEAR error condition later in Phase 4.
RECONNECT?	y	Reconnect inode I to file system in the directory for lost files (*lost+found*). This can cause a lost+found error condition in this phase if there are problems connecting inode *I* to *lost+found*.
CLEAR?	n	Ignore the error condition. A "no" response is appropriate only if the user intends to take other action to fix the problem.
CLEAR?	y	Deallocate the inode by zeroing its contents.
ADJUST?	n	Ignore the error condition. A "no" response is appropriate only if the user intends to take other action to fix the problem.
ADJUST?	y	Replace link count of file inode *I* with the link counted computed in Phase 2.
FIX?	n	Ignore the error condition. A "no" response is appropriate only if the user intends to take other action to fix the problem.
FIX?	y	Fix the problem.

Phase 4 Error Messages

`UNREF FILE I=I OWNER=O MODE=M SIZE=S MTIME=T (RECONNECT?)`

Inode *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 inode *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*.

There is nothing you can do at this point, but you should create the *lost+found* directory as soon as possible.

`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*.

There is nothing you can do at this point, but you should clean out the *lost+found* directory as soon as possible.

`(CLEAR)` The inode mentioned in the immediately previous UNREF error condition cannot be reconnected, so it is cleared.

`LINK COUNT FILE I=I OWNER=O MODE=M SIZE=S MTIME=T COUNT=X SHOULD BE Y (ADJUST?)`
The link count for inode *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 inode *I*, which is a directory, is *X* but should be *Y*. The owner *O*, mode *M*, size *S*, and modify time *T* of directory inode *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* inode *I* is *X* but should be *Y*. The file name *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?)`
Inode *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 inode *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=I OWNER=O MODE=M SIZE=S MTIME=T (CLEAR?)`
: Inode *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 inode *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 (CLEAR?)`
: Phase 1 or Phase 1B found duplicate blocks or bad blocks associated with file inode *I*. The owner *O*, mode *M*, size *S*, and modify time *T* of inode *I* are printed.

`BAD/DUP DIR I=I OWNER=O MODE=M SIZE=S MTIME=T (CLEAR?)`
: Phase 1 or Phase 1B found duplicate blocks or bad blocks associated with directory inode *I*. The owner *O*, mode *M*, size *S*, and modify time *T* of inode *I* are printed.

`FREE INODE COUNT WRONG IN SUPERBLK (FIX?)`
: The actual count of the free inodes does not match the count in the super-block of the file system.

Phase 5 Check Free List

Phase 5 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

Phase 5 Types of Error Messages

Phase 5 has four types of error messages:

- information messages
- messages that have a CONTINUE? prompt
- messages that have a FIX? prompt

- messages that have a SALVAGE? prompt

Phase 5 Meaning of Yes/No Responses

Table 8-6 describes the significance of responses to Phase 5 prompts:

Table 8-6 Meanings of Phase 5 *fsck* Responses

Prompt	Response	Meaning
CONTINUE?	n	Terminate the program.
CONTINUE?	y	Ignore rest of the free-block list and continue execution of *fsck*. This error condition always invokes BAD BLKS IN FREE LIST error condition later in Phase 5.
FIX?	n	Ignore the error condition. A "no" response is appropriate only if the user intends to take other action to fix the problem.
FIX?	y	Replace count in super-block by actual count.
SALVAGE?	n	Ignore the error condition. A "no" response is appropriate only if the user intends to take other action to fix the problem.
SALVAGE?	y	Replace actual free-block bitmap with a new free-block bitmap.

Phase 5 Error Messages

`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.

Phase 6 Salvage Free List

This phase reconstructs the free-block bitmap. There are no error messages that can be generated in this phase and no responses are required.

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.

`SUPERBLOCK MARKED DIRTY`
: A field in the super-block is queried by system utilities to decide if *fsck* must be run before mounting a file system. If this field is not "clean," *fsck* reports and asks if it should be cleaned.

`PRIMARY SUPERBLOCK WAS INVALID`
: If the primary super-block is too corrupt to use, and *fsck* can locate a secondary super-block, it asks to replace the primary super-block with the backup.

`SECONDARY SUPERBLOCK MISSING`
: If there is no secondary super-block, and *fsck* finds space for one (after the last cylinder group), it asks to create a secondary super-block.

`CHECKSUM WRONG IN SUPERBLOCK`
: An incorrect checksum makes a file system unmountable.

`***** FILE SYSTEM WAS MODIFIED *****`
: This is an advisory message indicating that the current file system was modified by *fsck*.

`***** REMOUNTING ROOT... *****`
: This is an advisory message indicating that *w* made changes to a mounted root file system. The automatic remount ensures that incore data structures and the file system are consistent.

How the File System Works

This section describes how the IRIX file system is constructed and how it works.

In the IRIX 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 merely another type of file that the user is permitted to use, but not allowed to write; the operating system itself retains the responsibility for writing directories. The combination of directories and files make up a file system. The starting point of any IRIX file system is a directory that serves as the *root*. In the IRIX 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 (/).

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 IRIX EFS file system may contain the following types of files:

- directories (d)
- regular files (-)
- character devices (c)
- block devices (b)
- named pipes (p)
- symbolic links (l)
- UNIX domain sockets (s)

The letter in parentheses following each item is the character used by *ls -l* to identify the file type.

Tables in Memory

When a file system is identified to the IRIX 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. Disk inodes and free storage bitmaps are read in from the disk as needed.

The System I-Node Table

The IRIX system maintains a structure known as the system inode table. Whenever a file is opened, its inode is copied from the secondary storage disk into the system inode table. If two or more processes have the same file open, they share the same inode table entry. The entry includes:

- the name of the device from which the inode was copied
- the inode 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)

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:

- a flag to tell how the file was opened (read/write)
- a count of the processes pointing to this entry (when the count drops to zero, the system drops the entry)
- a pointer to the system inode-table entry that is referenced by this file-table entry
- a 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.

System Steps in Accessing a File

The next few paragraphs describe steps the operating system takes to open, create, read, and write a file.

Open

If you give the pathname */a/b* to the *open*(2) system call, the following is performed. (Your program probably uses the *fopen*(3) subroutine from the standard I/O library, but that in turn invokes the *open* system call.)

1. The operating system sees that the pathname starts with a slash, so the root inode is obtained from the inode table.
2. Using the root inode, 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 inode list at which the inode 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 inode for ''b''.
6. The inode for ''b'' is determined to be a file and is copied to the system inode table (assuming it's not already there), and the reference count is incremented.
7. The system file table entry is allocated, the pointer to the system inode 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 inode of a file illustrates why it is advisable to keep directories small.

Create

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

1. If the file does not already exist, a free inode is located by searching the inode areas of the file system.
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 inode.
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 calls 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 is obtained.
2. The user buffer address and number of bytes to read and 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 inode is found by following the pointer from the system file-table entry to the system inode table. The operating system copies the data from storage to the user's buffer.
4. (Writing) The same pointer chain is followed, but the system writes into the data blocks. If new blocks are needed, they are allocated from the file system's list of free blocks. The EFS file system always attempts to grow the last extent if the following blocks are free, or to preallocate a large number of contiguous free blocks when allocating through a new extent. Disk blocks that aren't needed are freed when the file is closed.

5. Before the system call returns to the user, the number of bytes read or written is added to the offset in the system file table.
6. 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 (as happens after a *fork*(1)), 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 inode 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 inode table is picked up. Given the inode, the system can find the data blocks that make up the file.

A running process can successfully read and write a file after it has opened it, even if someone has since renamed or unlinked it. This might be important when it seems that there are more used blocks than that accounted for by simply examining files in a directory hierarchy. In this case, the blocks are freed when the process closes the file with the *close*(2) system call or exits the file with the *exit*(2) system call.

Synchronization

The above description of pathname conversion, while complex, is rather neat and orderly. The situation is complicated, however, by the fact that the IRIX system 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 inodes 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 inodes 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.

Processes that must ensure that data is written to the disk immediately can open the file with the *O_SYNC* flag, which causes the data and inode information to be written to the disk at the time of the write system call. Alternatively, the *fsync*(2) system call flushes all data to the disk associated with the particular file descriptor argument.

Search Time

Several things affect 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
- the layout of the file extents

As described above, when the IRIX system is locating a file to be opened, it searches linearly through all the directories in the pathname. Search time is reduced by keeping the number of entries in a directory small.

File System Corruption

Most often, a file system is corrupted because the address and count information fail to make it out to the storage medium. This is caused by:

- hardware failure
- human error

Problems can occur from any combination of these factors.

Hardware Failure

There is no fool-proof way to predict hardware failure. The best way to avoid hardware failures is to conscientiously follow recommended diagnostic and maintenance procedures. Use the *fx* utility to flag bad blocks on a hard disk and remap them to good blocks.

Human Error

Human error is probably the greatest single cause of file system corruption. To avoid problems, follow these rules closely:

1. ALWAYS shut down the system properly. Do not simply turn off power to the system. Use a standard system shutdown tool, such as *shutdown*(1M).
2. NEVER remove a file system physically (pull out a hard disk) without first unmounting the file system.
3. NEVER physically write-protect a mounted file system, unless it is mounted read-only.

The best way to insure against data loss is to make regular, careful backups. See Chapter 6, "Backing Up and Restoring Files," for complete information on system backups.

Insufficient Space on the root File System

The *root* file system is typically very static. It contains only basic programs and utilities. A prime reason for running out of space on the *root* file system is application programs creating many, sometimes very large, files in */tmp*.

If you need to increase space, consider these alternatives:

- Identify applications that cause the most problems, and configure them to use */usr/tmp.O* instead of */tmp* for temporary files. Most applications

recognize the *TMPDIR* environment variable, which specifies the directory to use instead of the default. For example, with *csh*(1):

```
setenv TMPDIR /usr/tmp.O
```

With *sh*(1):

```
TMPDIR=/usr/tmp.O ; export TMPDIR
```

- Make */tmp* a mounted file system. You can ''carve'' a */tmp* file system out of other file systems if need be.
- Make */tmp* a symbolic link to */usr/tmp.O*.

Note: If you use either of the last two options listed above, you should exercise great care when your system is in single-user mode, since the mounted file systems will not be present and your mounts and links to */tmp* will therefore not be active.

Chapter 9

Administering Printers

Chapter 9 describes the printing facilities that are part of the IRIX system. Nearly all systems use some kind of printer, and IRIX provides ready-made configuration files for most industry standard printers. If you have a graphics system, the System Manager utility, described in the Personal System Administration Guide, provides a convenient graphical interface for printer installation and maintenance. Specific tasks covered in Chapter 6 include:

- *Installing serial printers.*
- *Installing parallel printers.*
- *Installing network printers.*
- *Removing printers.*
- *Using the LP administration tools.*
- *Printer Cabling Requirements.*

Chapter 9

Administering Printers

One of the basic peripheral devices used by almost all workstations and servers is a printer. This chapter deals with installing and maintaining a printer with your IRIS workstation or server. Most computing facilities use printers regularly. Your IRIS supports many industry-standard printers.

The most convenient way for you to install a printer on your IRIS is to use the graphical System Manager. This method of adding a printer is described in the *Personal System Administration Guide.* If you have a non-graphical workstation or server, you can use this chapter for instructions on how to administer your printing system using IRIX shell commands. In this chapter, the terms "workstation" and "server" are used interchangeably, because the interface described here is identical across all Silicon Graphics products.

The *lp* system allows information to be printed and is one of the main utilities that users need on a regular basis. This chapter discusses:

- Adding and removing certain types of printers. See "Adding a Printer" on page 336, and "Removing Printers" on page 340.
- Registering network and hardwired printers. See "Registering Parallel and SCSI Printers" on page 336, "Registering Serial Printers" on page 338, and "Registering Network Printers" on page 338.
- The *lp* spooling system. See "Using the lp Spooler" on page 340.
- *lp* User Commands. See "lp User Commands" on page 342.
- Maintaining the *lp* system. See "Maintaining the lp System" on page 350.
- Troubleshooting *lp* system problems. See "Troubleshooting Your Printing System" on page 353.
- *lp* error messages. See "lp Error Messages" on page 356.
- Printer cabling. See "Printer Cable Pin Signal Tables" on page 368.

- The BSD lpr software. “Configuring and Troubleshooting the BSD LPR Spooler System” on page 372.

Adding a Printer

To send print requests to your printer, you must first add your printer by registering it with the line printer (*lp*) spooler. The printer should be registered with the *lp* spooler of the computer to which the printer is directly connected (hardwired), and also with the *lp* spooler of any computer accessing the printer through a network. The procedures for registering a printer with *lp* vary, depending on whether the printer is hardwired or accessed across a network; these procedures are described in the next sections.

Registering Parallel and SCSI Printers

There are several different types of printers you can connect to your system. In this chapter, printers are divided into three groups. The first group is parallel and SCSI printers. The second are serial printers and the third group is printers of any type connected to other systems. The printer hardware documentation should tell you whether your printer is a parallel or serial printer.

If you need to create the devices to use your parallel port or ports (some systems, such as CHALLENGE and Onyx series have multiple parallel ports), log in as **root** and use the commands:

```
cd /dev
./MAKEDEV plp
```

The MAKEDEV script will determine the number of parallel ports on your system (there is always at least one) and make the correct devices for all available ports.

The parallel interface port on the back of your workstation or server is clearly labeled. The special file */dev/plp* refers to the parallel printer interface and port. Some larger servers and workstations can have multiple parallel ports. There is one parallel port on each IO4 board on CHALLENGE and Onyx systems. If you have four IO4 boards, you have four parallel ports.

If you have more than one parallel port on your system, the device files that refer to the ports are named according to the board slot in which the CPU board is installed. For example, if you have an IO4 board in slot 2 and another in slot 4, the device files for the parallel ports will be */dev/plp*2 and */dev/plp4*. One of these devices (the parallel port attached to the board designated as the primary board) will be linked to the default */dev/plp*. Each parallel port on the back of your system is clearly labeled.

Parallel and SCSI printers are installed with the same procedure, but to different ports. Your SCSI printer hardware documentation should detail the cabling requirements for SCSI interface. Most printers can be installed using the graphical System Manager and the procedures described in the *Personal System Administration Guide*. The procedure below is for those systems without access to the System Manager.

To register printers connected directly to your computer via the parallel or SCSI ports, follow these steps:

1. Become the superuser with the *su*(1M) command.
2. To stop the print spooler, type:

   /usr/lib/lpshut
3. Assuming you have a raster printer attached to the parallel port, use the *mkcentpr*(1M) utility to install the printer in the *lp* system:

   mkcentpr

   The *mkcentpr* utility is an interactive script that will prompt you for all necessary information about your printer. You should be prepared to specify the device file for the parallel port (*/dev/plp* unless you have multiple CPU boards installed) and other specific information. Consult the *mkcentpr*(1M) reference page for complete information about *mkcentpr* syntax.
4. To set up this printer as the default printer, type:

   /usr/lib/lpadmin -d*printer-name*
5. To restart the print spooler, type:

   /usr/lib/lpsched

Your printer is now registered with the *lp* system and is ready for printing.

Registering Serial Printers

If you have a graphics system, the best way to add a serial printer is to use the System Manager provided with the Operating System. Servers and other non-graphics users use the *lpadmin*(1M) command. To add a printer to port 2 on your system, perform the following steps:

1. To stop all printer activity on your system for the duration of this procedure, give the command:

   ```
   lpshut
   ```

2. To define the printer interface for the new device, give the following command:

   lpadmin -p *printer-name* **-m** *model* **-v** */dev/ttyd2*

   where *model* is a type of printer listed in the */var/spool/lp/model* file.

3. To make the new printer the default destination for your *lp*(1) print jobs, give the command:

   lpadmin -d *printer-name* `To enable the printer to receive requests, give the command:`

   enable *printer-name*

4. To enable *lp* to process requests for this printer, give the command:

   /usr/lib/accept *printer-name*

5. Finally, restart printer activity on your system with the command:

   /usr/lib/lpsched

Registering Network Printers

The most convenient way for graphical workstation users to add a network printer is to use the graphical System Manager. The Print Manager utility is described in detail in the *Personal System Administration Guide*.

To configure a printer for use across a network using IRIX shell commands, follow these steps:

1. Log in as the superuser to the workstation with the attached printer.
2. Replace *remote_workstation* in the command below with the name of the workstation that needs access to the printer. Type:

addclient *remote_workstation*

addclient grants permission for the specified *remote_workstation* to access printers across the network. It is important to realize that by giving this command, you are allowing anyone on the remote machine who has access to the *lp* account there to have the privileges of the lp account on your system.

If you want all remote workstations to be able to use printers on your system, type:

```
addclient -a
```

Note: Printers must be configured on the system to which they are attached before remote workstations can configure them successfully across the network.

3. Both workstations must be able to communicate across the network. For additional information on network communications, see Chapter 15, "Understanding Silicon Graphics' Networking Products."
4. On the *local* workstation, become superuser.
5. To stop the spooler, type the command:

   /usr/lib/lpshut

6. On the *local* workstation, add the printer to the *lp* spooler with the script *mknetpr*:

   mknetpr *printer hostname netprinter*

   printer is the local name you want for the remote printer and should be no more than 14 characters long. *hostname* is the name of the workstation or server the remote printer is on, and *netprinter* is the name of the printer on that workstation or server.

7. To set up this printer as the default printer, type this command on the *local* workstation:

   /usr/lib/lpadmin -d*printer-name*

8. Restart the spooler with this command:

   /usr/lib/lpsched

Removing Printers

Under some circumstances, you may want to remove one or more printers from the *lp* system. The *rmprinter*(1M) utility allows you to remove a specified printer. The *preset*(1M) utility allows you to reset your entire *lp* system to the way it was when you received your workstation from Silicon Graphics, Inc. To remove a specified printer, follow these steps:

1. Become the superuser.
2. Remove the printer by entering the command below. Replace *printer-name* with the name of the printer you wish to remove:

   `rmprinter` *printer-name*

Your printer is now removed from the *lp* system.

To remove all printers on your system, use the *preset* command.

Caution: Use *preset* with extreme care: it removes all printer configuration information.

1. Become the superuser.
2. Type:

   `preset`

`Your` *lp* system is now completely reset and all printers are removed.

Using the lp Spooler

The Line Printer (*lp*) Spooling Utilities are a set of eleven commands that allow you to *spool* a file that you want to print. Spooling is a technique that temporarily stores data until it is ready to be processed (in this case, by your printer). For *lp* spooling, a file (or group of files) to be printed is stored in a queue until a printer becomes available. When the printer is ready, the next file in the queue is printed.

lp spooling allows you to use your workstation without waiting for your file to print. *lp* spooling also lets you share printers among many users. The flow of printing throughout your system is regulated by the *lp* Spooling Utilities.

The *lp* Spooling Utilities allow:

- Customizing your system so that it will spool to a pool of printers. (These printers need not be the same type.)
- Grouping printers together into logical classes to maximize throughput.
- Queueing print requests, thus allowing a print request (job) to be processed by the next available printer.
- Canceling print requests, so that an unnecessary job will not be printed.
- Starting and stopping *lp* from processing print requests.
- Changing printer configurations.
- Reporting the status of the *lp* scheduler.
- Restarting any printing that was not completed when the system was powered down.
- Moving print requests and queues from one printer or class of printers to another.

The eleven *lp* spooling commands are divided into two categories: *user* commands for general use of the *lp* system; and *administrative* commands for system configuration and maintenance.

lp Terms and Conventions

These terms represent important concepts used in this document:

printer	A logical name that points to an interface file, which represents a physical device, that is, the actual printer.
class	The name given to an ordered list of one or more printers. A printer may be assigned to more than one class, but need not be a member of any class.
destination	The place an *lp* request is sent to await printing. The destination may be a specific printer or a class of printers. An output request sent to a specific printer will be printed only by that printer; a request sent to a class of printers will be printed by the first available printer in its class.

lp User Commands

The commands described in this section allow all users on your workstations and your network to access the printing facilities. Your users should use the *lp* and *cancel* commands most frequently, the *lpstat* command occasionally, and the *enable* and *disable* commands infrequently, if ever.

User Command Summary

This section describes the five basic *lp* commands.

lp	Routes jobs to a destination and places them in a queue. The destination may be either a single printer or a class of printers.
cancel	Cancels output requests.
disable	Prevents a printer from printing jobs in the queue.
enable	Allows a printer to print jobs in the queue.
lpstat	Reports the status of many aspects of the *lp* spooling system.

lp: Make an Output Request

The *lp* command routes a job request to a destination where it is placed in a queue to await printing. The destination may be a single printer or a class of printers. If you do not specify a destination, the request is routed to the default destination. For information on how to set the default printer destination, see “Changing the Default Printer Destination” on page 350

The form of the *lp* command is:

`lp` [*options*] *filenames*

Every time an *lp* request is made, a ‘‘request-ID’’ is assigned to the job, and a record of the request is sent to you. The request-ID has this form:

`destination-`*seqnum*

destination is the printer or class of printers to which the job has been routed. *seqnum* is an arbitrary sequence number assigned to the job by the *lp* system.

lp has three options which are particularly useful: -**n**, **-d**, and **-c**.

Use **-n** to print more than one copy of a document:

lp -n*number*

number is the number of copies to print. Note that there is no space between **-n** and **number**.

Use **-d** to specify a printer or class of printers other than the default printer (assuming your system has more than one printer defined):

lp -d*destination filenames*

Finally, use **-c** (for *copy*) to ensure that the edits that you make to your files once you have issued a print request do not show up in the printed output:

lp -c *filenames*

You can combine these command options in any order. For a complete list of *lp* options, see the *lp*(1) reference page. Some example uses of the *lp* command are shown here:

```
lp myfile
request id is myprinter-12 (1 file)
lp < myfile
request id is myprinter-13 (standard input)
cat myfile | lp
request id is myprinter-14 (standard input)
lp -n3 -dfoo -c myfile
request id is foo-15 (1 file)
```

To request a printout, you can use the *lp* command several different ways. The first three examples above perform identical functions, sending a simple print request to the default printer. The fourth example prints three copies on printer *foo* and creates a copy of the file for the printer to process, ensuring that if changes are made to the file after the print request, the original file will be printed.

cancel: Stop a Print Request

The *cancel* command removes a job from the queue. You can *cancel* a job either before or after it starts printing, but you can *cancel* only one at a time.

Any user can cancel any other user's job. If you cancel another user's print request, mail is sent to that user. Once you *cancel* a job, you can request that it be printed again only with the *lp* command:

`cancel` *printer-name*

`cancel` *request-ID*

Using the printer name cancels the job currently being printed. Using the request-ID cancels the specified job whether or not it is currently being printed, as shown below:

```
cancel myprinter
request "myprinter-16" cancelled
cancel myprinter-17
request "myprinter-17" cancelled
```

Issuing a *cancel* command does not work when the job is being printed on a remote workstation. To cancel a print job on a remote system, log in to the remote system and issue the *cancel* command.

disable: Stop Printer from Processing Requests

The *disable* command prevents the printer from printing jobs in the queue. Possible reasons for disabling the printer include malfunctioning hardware, paper jams, running out of paper, or end-of-day shutdowns. If a printer is busy at the time it is disabled, the request it was printing is reprinted in its entirety when you re-enable the printer.

You can send job requests to a printer that has been disabled. The jobs are put on the queue but are not printed until the printer is enabled.

To *disable* a printer, type:

`disable [-c] [-r``"`*reason*`"]` *printer(s)*

The **-c** option cancels the request currently being printed and disables the printer. This is useful if the current request causes the printer to behave abnormally.

The **-r** option lets you tell other users why you disabled a printer. *reason* is a character string and must be enclosed in double quotes (" "). This string is reported to anyone trying to use the disabled printer or to anyone issuing the *lpstat*(1M) command.

enable: Allow Printer to Process Requests

The *enable* command permits a printer that has been disabled to begin printing jobs from the queue. The example below demonstrates the *enable* command. To *enable* a printer, type:

enable *printer(s)*

```
disable -r"paper jam" myprinter
printer "myprinter" now disabled
enable myprinter
printer "myprinter" now enabled
```

lpstat: Report lp Status

The *lpstat* command gives you a report on the status of various aspects of the *lp* system. To check *lp* status, type:

lpstat [*options*]

The most useful option is **-t**, which gives a complete report on the status of the *lp* system. For a complete list of options, see the *lpstat*(1) reference page. The following example demonstrates the *lpstat* command:

```
lpstat -t
scheduler is running
system default destination: myprinter
members of class foo:
myprinter
device for myprinter: /dev/plp
myprinter accepting requests since Jul 31 21:40
foo accepting requests since Jul 30 12:23
printer myprinter now printing foo-18
```

```
enabled since Aug 5 15:34
foo-18 mylogin 3156 Aug 7 17:11 on myprinter
```

Administrative Commands

This section summarizes the commands that are used to administer the *lp* system. To execute the administrative commands, you must be logged in as either *root* (that is, the superuser) or as *lp*. Inexperienced users should not use the *lp* administrative commands.

Administrative Command Summary

lpsched	Starts the *lp* scheduler.
lpshut	Stops the *lp* scheduler.
reject	Prevents jobs from queueing at a particular destination.
accept	Permits job requests to queue at a particular destination.
lpmove	Moves printer requests from one destination to another.
lpadmin	Configures the *lp* system.

lpsched : Start the lp Scheduler

The *lpsched* command starts the *lp* scheduler. *lp* prints jobs only when the scheduler is running. *lpsched* is executed automatically each time the computer is booted.

Every time *lpsched* is executed, it creates a file called *SCHEDLOCK* in */var/spool/lp*. As long as this file exists, the system will not allow another scheduler to run. When the scheduler is stopped under normal conditions, *SCHEDLOCK* is automatically removed. If the scheduler stops abnormally, you must remove *SCHEDLOCK* before you use the *lpsched* command. You may need to use this procedure to restart the scheduler after the system shuts down abnormally.

To start the *lp* scheduler, type:

```
/usr/lib/lpsched
```

There is no response from the system to acknowledge the *lpsched* command; to verify that the scheduler is running, use *lpstat*.

lpshut: Stop the lp Scheduler

The *lpshut* command stops the *lp* scheduler and ends all printing activity. All requests that are being printed when you issue the *lpshut* command are reprinted in their entirety when the scheduler is restarted.

To stop the *lp* scheduler, type:

```
/usr/lib/lpshut
```

reject: Prevent Print Requests

The *reject* command stops *lp* from routing requests to a destination queue. For example, if a printer has been removed for repairs, or has received too many requests, you may wish to prevent new jobs from being queued at that destination.

If the printer is enabled, all requests that are in the queue when you issue the *reject* command are printed.

The *reject* command takes the form:

/usr/lib/reject [**-r**"*reason*"] *destination*

The **-r** option lets you tell other users why print requests are being rejected. *reason* is a character string and is enclosed in double quotes (" "). This string is reported to anyone trying to use *lp* to send requests to the specified destination.

accept : Allow Print Requests

The *accept* command allows job requests to be placed in a queue at the named printer(s) or class(es) of printers. As shown in the example below, *accept* allows a printer to receive job requests and *reject* disables printing:

```
/usr/lib/accept myprinter
destination "myprinter" now accepting requests
/usr/lib/reject -r"printer broken" myprinter
```

```
destination "myprinter" is no longer accepting requests
```

lpmove: Move a Request to Another Printer

The *lpmove* command moves print requests from one destination to another. For example, if you have a printer removed for repairs, you may want to move all jobs pending on the queue to a destination with a working printer. You may also use *lpmove* to move specific requests from one destination to another, but only after you have halted the scheduler with the *lpshut* command. *lpmove* automatically rejects job requests re-routed to a destination without a printer. The *lpmove* command takes two forms:

/usr/lib/lpmove *dest1 dest2*

/usr/lib/lpmove *request(s) destination*

dest1, *dest2*, and *destination* are printers or classes of printers. *request* is a specific request-ID.

In the first form of the command, all requests are moved from *dest1* to *dest2*. After the move, the printer or printers at *dest1* will not accept requests until you issue an *accept* command. All re-routed requests are renamed *dest2-nnn*, where *nnn* is a new sequence number in the queue for destination *dest2*. In the second form, which you may issue only after you stop the scheduler, the re-routed requests are renamed *destination-nnn*. When you restart the scheduler, the original destinations will still accept new requests. The three commands in the example below demonstrate the usage of the *lpmove* command:

/usr/lib/lpmove *myprinter yourprinter*

lpshut

/usr/lib/lpmove foo-19 foo-20 *yourprinter*

After the third command, you see the message:

```
total of 2 requests moved to yourprinter
```

lpadmin: Configure Printers

The *lpadmin* command has two primary uses: adding new printers to the system and changing printer classes and destinations. Silicon Graphics has some routines to automatically add some of the printers supported for use

with the workstation. These commands include *mkPS*(1M), *mkcentpr*(1M), and *mknetpr*(1M). These should be used whenever possible. For a list of supported printers, see the file */var/spool/lp/model*.

Unlike most IRIX commands, *lpadmin* requires an option. The *lpadmin* command takes three forms:

lpadmin -d*destination*

lpadmin -x*destination*

lpadmin -p*printer* [*options*]

The **-d** option sets the system default destination. The *destination* must already be installed when you issue the command.

The **-x** option removes the specified *destination* from the *lp* system. This form of the *lpadmin* command will not work while the scheduler is running.

You cannot remove a destination (printer or class) if it has pending requests; you must first either remove all requests with the *cancel* command or move them to other destinations with *lpmove*.

Removing the last remaining member of a class deletes that class from *lp*. Removal of a class, however, does not imply the removal of printers assigned to that class.

The **-p** form of the *lpadmin* command has two options that let you reassign printers to different classes. With these options, the *lpadmin* command takes the form:

lpadmin -p*printer* [**-c***class*] [**-r***class*]

The **-c** option assigns a *printer* to the specified *class*; the **-r** option removes a *printer* from the specified *class*.

```
/usr/lib/lpadmin -xmyprinter
/usr/lib/lpadmin -dmyprinter -rfoo -cboo
```

The **-p** options will not work while the scheduler is running. For a complete list of options, see the *lpadmin* (1M) reference page. After creating a printer, you may need to edit the interface file: */var/spool /lp/interface/printername*. The interface file controls such things as printer baud rate, log file site, and existence of banners.

Maintaining the lp System

This section contains procedures for changing your default printer, clearing printer *log* files, and printing over a network.

Changing the Default Printer Destination

The *lp* command determines a request's destination by checking for a **-d** option on the command line. If **-d** is not present, it checks to see if the environment variable LPDEST is set. If LPDEST is not set, then the request is routed to the default destination.

The system default destination can be a printer or a printer class.You can set it by using the *lpadmin* command with the **-d** option. The system default must be set by the user. A destination must already exist on the *lp* system before you can designate it as the default destination.

Setting the environment variable LPDEST allows a user to have a default destination other than the system default.

Clearing Out log Files

A *log* file keeps a record of all printing activity on a given printer. Each printer keeps a *log* file, located in */var/spool/lp/transcript/log*. You can change the name of the file by editing */var/spool/interface/printername* and changing the value of the LOGFILE variable.

Each file contains a running list of processed jobs, each of which includes the following:

- the *logname* of the user who made the request
- the request ID
- the name of the printer that processed the request
- the date and time the printing started

Any *lpsched* error messages that occur are also recorded.

If there are a lot of *lp* requests for a given printer, that printer's *log* file will soon get very large. You can manually remove the contents of these files from time to time, or you can set up the computer to do it for you automatically at regular intervals.

Included in */var/spool/lp/etc/lib* is a shell script *log.rotate* that automatically rotates (cleans out) your printers' log files once per day at 4:12 a.m. To set up the script for your printer(s), you must edit *log.rotate* and */var/spool/cron/crontabs/root* in the following manner:

1. Become the superuser and change directories to */var/spool/lp/etc/lib*:

   `cd /var/spool/lp/etc/lib`

2. In the file *log.rotate*, remove the comment marker (#) from the following line:

   `# printers="`*PRINTER1 PRINTER2*`"`

   In place of *PRINTER1* and *PRINTER2*, put the names of any parallel-interface (that is, color) printers and any remote printers. Any number of printers may be included. If you have no color or remote printers, use two double quotes to make a null string (" ") in place of the printer names.

3. In the file *log.rotate*, remove the comment marker (#) from the following line:

   `# LocalPS="`*PRINTER1*`"`

   In place of *PRINTER1*, put the names of any hardwired (that is, connected to the serial port) printers.

4. Change directories to */var/spool/cron/crontabs*:

   `cd /var/spool/cron/crontabs`

5. Edit *root* by removing the comment marker (#) from the line containing *log.rotate*.

6. Write and exit from the file. The *lp* log will be rotated by the script *log.rotate*, which is called by the *cron* daemon.

7. For more information about using *cron* to automate tasks, see "Automating Tasks with at(1), batch(1), and cron(1M)" on page 29.

Printing Over the Network

Remote printing on the workstation or server allows you to send print jobs over the network with the same commands you use to send jobs to a printer connected directly to your workstation. This is accomplished by giving a remote printer a local name so that the local *lp* scheduler is "fooled" into thinking it is sending the request to a local printer. After the local workstation's *lp* spooler queues the print request, it is sent across the network to the remote workstation or server, where it is processed by that workstation's *lp* spooler. As a result of this, you cannot accurately determine the status of a remote print request by using the *lpstat* command on the local workstation.

This section covers two aspects of remote printing:

- checking the status of remote print requests
- canceling remote print requests

Checking Remote Printer Status

When you send a print request across the network to a remote workstation, the local *lp* system always reports that the request is being printed, regardless of its actual status in the remote workstation's *lp* system. To check the true status, you must remotely access (using *rsh* or *rlogin*) the workstation whose printer is processing the job. The remote *lp* scheduler changes the request ID of any job sent to it over the net to reflect the actual name of the printer and gives it a new sequence number corresponding to its place in the remote queue. To determine a specific job's status, look in the remote printer's *log* file (that is, the *log* file on the remote workstation) with the *tail* command. The example below uses *rsh* to access the remote workstation:

`rsh` *host* **`tail`** *logpath*

host is the name of the remote workstation. *logpath* is the pathname of the remote printer's log file.

Canceling Remote Print Requests

Once you know the remote printer status, you can use the *cancel* command on the remote workstation to cancel any jobs on the printer's queue. You must cancel a remote print job from the remote workstation once it has been sent over the network by the local *lp* system.

Troubleshooting Your Printing System

If you send a print request to a printer with *lp*, *psroff*, or *imprint* and do not receive any output, you should use the checklists below to make sure your system is ready for printing. Use these lists as a supplement to the troubleshooting information in the manufacturer's hardware manual.

Hardware Troubleshooting Checklist

Use the following list of questions to determine if your printer hardware is working as designed:

- Is the printer turned on?

  Printers do not always indicate clearly if they are turned on. Make sure the printer is plugged into the power socket and the power switch is on.

- Does the printer have paper?

  Frequently, printers run out of paper in a high-volume situation.

- Is there a paper jam?

  Make sure the entire paper pathway is clear of sheets or fragments of paper. Refer to your printer hardware documentation before attempting to put any unusual paper or other media through your printer.

- Is the printer set to the correct baud?

  Be sure the baud rate of the printer matches that of the serial port.

- Is the serial cable attached correctly?

  Often, reseating the serial cable where it connects to the printer will restore correct operation.

- Is the correct cable being used?

The usage of the pins in serial cables varies somewhat in different applications. Cables designed for specific hardware may or may not function correctly with different hardware. Check your workstation *Owner's Guide* and the documentation supplied with your printer and cable to determine if the cable is correct for your hardware.

Software Troubleshooting Checklist

The *lp* scheduler is the program in charge of spooling your files to the printer, and it is invoked whenever you use the *lp*, *psroff*, or *imprint* print commands. The scheduler can be in a number of states, and each printer registered with *lp* can be in a number of states as well.

To check on the complete status of the *lp* system, type:

```
lpstat -t
```

This gives you a complete description of the status of *lp*. You may also want to examine the contents of the file */var/spool/lp/log*. Use the information you find to answer the following questions:

- Is your printer registered with *lp*?

  If you do not see the name of your printer in the list of information produced by *lpstat*, then you must register your printer with *lp*.

- Is the printer enabled?

  If your printer is not enabled, the *lpstat* listing will contain this line:

  `printer` *yourprinter* `disabled since...`

  In order to enable the printer, type:

  `enable` *yourprinter*

  lp sometimes disables a printer automatically if it is unable to send a file to a remote printer, so a disabled printer often indicates a hardware problem, such as a host that is not communicating with the network.

- Is the printer accepting requests?

  If the printer is not accepting requests, the *lpstat* listing will contain this line:

  yourprinter `not accepting requests since...`

You must execute the *accept* command for that printer destination. Become the superuser (with *su*) and type:

/usr/lib/accept *yourprinter*

- Is the *lp* scheduler running?

  If the scheduler is not running, the *lpstat* listing will contain the message:

  ```
  scheduler is not running
  ```

  To restart the *lp* scheduler, become superuser (with *su*) and type:

  /usr/lib/lpsched

- Did you specify the right printer?

  If your system has more than one printer, and you wish to send a job to a printer other than the default, remember to use the **-d** option:

  lp -d*otherprinter*

  psroff -d*otherprinter*

Troubleshooting Network Printers

If you are having trouble with a printer you are accessing over a network, you should check the status of the *lp* scheduler on your workstation *and* the printer's host workstation.

Emergency Measures

If none of the above procedures works, there are several "last resort" procedures:

1. Stop the *lp* scheduler and then restart it. As *root*, type the following sequence of commands:

   /usr/lib/lpshut

   Then kill any jobs running as *lp.* You can identify these processes with the command:

   ps -fu lp

   Then give the command:

   /usr/lib/lpsched

2. Remove the offending printer destination from the *lp* scheduler, and then register it again. Before you can do this you must either cancel any print requests going to the printer or move them to another print destination (if you have more than one).
3. As an absolute last resort, remove all printers from the *lp* system, reboot the computer, and register them all once again.

lp 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, *myprinter-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.

These messages can be found in the printer log files if you miss them on the system console. 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.

- 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 view a list of all the printers and requests waiting to get printed.

- xx is not a request

  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 */var/spool/lp/FIFO* is incorrect. The mode should be 600 with the owner lp and the group lp.

- 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 *lp*.

- destination "printer-name" unknown

  Use the *accept* command to enable the printer so that 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.

- can't create class "xx"- existing printer name

  The class name you are trying to use has already been given to a printer. You need 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 */var/spool/lp* directory. It should be 755 with the owner lp and the group lp.

- can't create new class file

  The mode may be wrong on the */var/spool/lp* directory. It should be 755 with the owner *lp* and the group *lp*.

- can't create new interface program

  The mode may be wrong on the */var/spool/lp/interface* directory. It should be 755 with the owner *lp* and the group *lp*.

- can't create new member file

  The mode may be wrong on the */var/spool/lp/member* directory. It should `be 755 with the owner lp and the group lp.`

- can't create new printer status file

  The mode may be wrong on the */var/spool/lp/pstatus*. It should be 644 with the owner *lp* and the group *lp*.

- can't create new request directory

  The mode may be wrong on the */var/spool/lp/request* directory. It should be 755 with the owner *lp* and the group *lp*.

- can't create "printer-name" -- existing class name

  The printer name you are trying to use has already been used as a class name. You need to assign another name for the printer.

- can't create new output queue

  The mode on the file */var/spool/lp/seqfile* is incorrect. It should be 644, and the mode on the directory should be 755. The owner and the group should be *lp*. You can correct this by typing the command at a later time.

- can't create new sequence number file

  The mode on the file */var/spool/lp/seqfile* is incorrect. The mode of the file should be 644, and the mode of the directory should be 755. The owner and the group should be *lp*. You can correct this by typing the command at a later time.

- can't create request file xx

The mode on the file */var/spool/lp/request/printer-name/r-id* is incorrect. *Printer-name* is the name of the printer such as 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 and the group should be *lp*. You can correct this by typing the command at a later time.

- can't fork

  You either have several processes running and are not allowed to run any more, or the system has all the processes running that it can handle. You must rerun this command later.

- can't lock acceptance status

  This is a temporary file in */var/spool/lp* that prevents more than one *lp* request from being taken at any one time. You must rerun this command later.

- can't lock output queue

  The file */var/spool/lp/QSTATLOCK* prevents more than one *lp* request from being printed on a printer at one time. You must rerun this command later.

- can't lock printer status

  The temporary file */var/spool/lp/PSTATLOCK* prevents more than one *lp* request from being printed on a printer at one time. You must rerun this command later.

- can't lock sequence number file

  The file */var/spool/lp/SEQLOCK* prevents more than one *lp* request from getting the next printer-id (request identification) number at one 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 */var/spool/lp/request*. Also, after you shut down the *lp* scheduler, you must manually move the request from the disabled printer directory to the new destination.

- 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. You can correct this by typing the command at a later time.

- can't open member file

  The *lp* program is trying to access the list of members in the directory */var/spool/lp/member*. The system could have the maximum number of files open that are allowed at any time. You can correct this by typing the command at a later time.

- can't open xx file in MEMBER directory

  There are a number of reasons why file *xx* in the */var/spool/lp/member* directory cannot be opened. The mode on the file could be incorrect; it should be 644. The system could have the maximum number of files open that are allowed at any time; you can correct this by typing the command at a later time.

- can't open xx file in class directory

  If file *xx* cannot be opened, it's possible 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 the */dev/tty* file. The mode should be 622.

- can't open FIFO

  The mode on the named pipe file */var/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. You can correct the latter problem by typing the command at a later time.

- can't open MEMBER directory

  The mode on the directory */var/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 this is the case, try typing the command at a later time.

- can’t open acceptance status file

  The mode on the file */var/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. You can correct the latter problem by typing the command at a later time.

- can’t open default destination file

  Check the mode on the file */var/spool/lp/default*; it should be 644. If the mode is correct, it could be that the system has the maximum number of files open that are allowed at any one time. You can correct this by trying the command at a later time.

- can’t open file filename

  You incorrectly typed the filename or you don’t have the correct modes set. If you are the owner, the mode should be at least 400.

- can’t open output queue

  Check the mode on the file */var/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 */var/spool/lp/pstatus* is incorrect; it should be 644. This message is also generated if the system has the maximum number of files open that are allowed at any one time. You can correct this by trying the command at a later time.

- can’t open request directory

  The mode on the directory */var/spool/lp/request* is incorrect; it should be 655. The system may also have the maximum number of files open that are allowed at any one time. You can correct this by trying the command at a later time.

- can’t open request file xx

  The mode on the file */var/spool/lp/member/request/xx* is incorrect. The mode should be 644. The system may also have the maximum number of files open that are allowed at any one time. You can correct this by trying the *lpmove* command at a later time.

- can’t open system default destination file

The mode on the file */var/spool/lp/default* is incorrect. The mode should be 644. The system may also have the maximum number of files open that are allowed at any one time. You can correct this by trying the command again at a later time.

- can't open temporary output queue

  The mode on the file */var/spool/lp/outputq* is incorrect. The mode should be 644. The system may also have the maximum number of files open that are allowed at any one time. You can correct this 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 then try 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 */var/spool/lp/class*. It should be 755. The owner and the group should be *lp*. The file in that directory may also have the wrong mode; it should be 644.

- can't remove printer

  The mode may be wrong on the */var/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 *lp*.

- can't remove request directory

  The mode may be wrong on the */var/spool/lp/request* directory. It should be 755 and should be owned by *lp*, and the group should be *lp*. The directory may still have pending requests to be printed, which 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 be used only 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 /var/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* or */dev/plp* file. It should be 622 and owned by *lp*.

- cannot create temp file filename

  The system may be out of free space on the */var* file system. Use the command

  ```
  df /var
  ```

  to determine the number of free blocks. Several hundred blocks are required to insure that the system performs correctly.

- class xx has disappeared!

  Class *xx* was probably removed after the scheduler was started. The system may be out of free space on the */var* file system. To find out, use the following command:

  ```
  df /var
  ```

  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 */var* file system. To find out how much free space is available, use the following command:

  ```
  df /var
  ```

  The class will probably have to be restored from a backup.

- class directory has disappeared!

  The */var/spool/lp/class* directory has been removed. The system may be out of free space on */var*; use the **`df /var`** 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 */var/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* command to set up a default destination or set your LPDEST environment variable to the value of the destination.

- destination dest has disappeared!

  A destination printer, *dest*, has been removed after *lpsched* was started. Use the *lpadmin* command to remove the printer.

- destination printer no longer accepting requests

  The printer has been disabled using the *reject* command. Use the *accept* command to re-enable the printer.

- 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 after the scheduler was started.

- destination printer was already accepting requests

  The destination printer was previously enabled. Once a printer is accepting requests, any further *accept* commands are ignored.

- destination printer 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 need to specify a printer to move the print requests from and a different printer to move the requests to.

- disabled by scheduler: login terminal

The login terminal has been disabled by the *lp* scheduler. Use the *enable* command to re-enable the printer.

- error in printer request printer-id

  Printer-id is the actual request identification number. An error has likely occurred in the printer. Check the printer, and reset it if needed.

- illegal keyletter xx

  An invalid option, *xx*, was used. See the reference 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 command line:

  ```
  lpadmin -m model
  ```

  the argument to the **-m** option is the name of a model interface program.

- keyletters -e, -i, and -m are mutually exclusive

  These options to the *lpadmin* command cannot be used together. Refer to the reference 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 */var/spool/lp/member* directory has been removed. The system is probably out of free disk space in the */var* file system. You need to clean up the */var* file system, and then install the *lp* commands or retrieve them from a backup.

- model xx non-existent

  The name that you are using for a model interface program is not valid. A list of valid models is in the */var/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* reference 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 */var/spool/lp/pstatus* file but no longer exists. Use the *lpadmin* command to remove the printer.

- 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 xx

  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!

  The printer has been removed, and the *enable* command cannot find it. The printer was most likely removed after the workstation was rebooted or after the scheduler was started.

- printer printer-name non-existent

  Printer-name is the name of a printer that has been removed after the scheduler has been started. You must use the command:

  `lpadmin -x`*printer-name*

- printer status entry for printer has disappeared

  The */var/spool/lp/pstatus* file must have been corrupted. You need 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 need to use the *lpshut* command first.

- spool directory non-existent

  The directory */var/spool* has been removed. You need to use the *mkdir* command to restore the directory. This has probably removed some of the necessary *lp* files. You may have to reinstall the *lp* commands.

- standard input is empty

  You specified an invalid file name either by incorrectly typing a name or by specifying a nonexistent file. Nothing will be 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, refer to the *lp* reference page.

Printer Cable Pin Signal Tables

Your workstation or server has one or more serial ports and one or more parallel ports. The serial ports on your system are either DB-9 format or Mini-DIN8. The parallel port is the industry standard 25-pin parallel port. For your use in determining correct cabling, the following sections provide pin signal tables for the serial and parallel ports on your system.

Parallel Port Pin Signal Table

The parallel port on your system uses industry standard parallel printer cables to connect to common printers. It is recommended that you use a parallel cable no longer than 10 feet. Using a parallel hookup usually results in faster printing from your system. Also, because serial ports are generally more in demand for modems, terminals, and other peripheral devices, using the parallel port, when possible, saves a serial port for other uses.

Table 9-2 shows the parallel port pin signal table for IRIS systems:

Table 9-1 Parallel Port Pins and Signals

Pin	Signal
1	STB
2	DATA1
3	DATA2
4	DATA3
5	DATA4
6	DATA5
7	DATA6
8	DATA7
9	DATA8
10	ACK
11	BUSY

Table 9-1 (continued) Parallel Port Pins and Signals

Pin	Signal
12	PE
13	ONLINE
14	PR/SC
15	FAULT
16	RESET
17	NO INK
18	NC*
19-25	Signal Ground

*NC stands for "no connect," meaning the wire is not used.

Serial Port Pin Signal Tables

The serial port on your workstation or server is either a DB-9 (9 pin) or a Mini-DIN8 port. In either case, the port is clearly labeled as a serial port. Consult your hardware *Owner's Guide* to determine which kind of port your system has. There are 2 basic printer cable configurations for the DB-9 serial ports and one for the Mini-DIN8 serial port. Depending on the cables used some serial functionality may be sacrificed due to unconnected wires in the cable. Note that the pinout of the DB-9 connectors is different than that of the full size DIN connectors that may be next to them. These DIN connectors also have different pinouts than the Mini-DIN8 connectors used on some systems and documented in this section. The DB-9 and full size DIN connectors are connected to the same internal port hardware.

DB-9 Connector Cabling

For most serial printers, you should use the following cable. This cable uses the normal 3-wire connection and be used as a */dev/ttyd** device. Table 9-2 shows typical DB-9 serial cabling:

Table 9-2 DB-9 Serial Cable

Function	DB-9-Male	DB25-Male
TXD	1	nc*
RXD	2	3
	3	2
	4	nc
	5	nc
	6	nc
GND	7	7
DCD**	8	20
	9	nc

*nc stands for "no connect," meaning the wire is not used.

** DCD is only used with */dev/ttym** devices if the system must notice when the printer powers off. Normally it is not used.

Note: Do not use a cable designed for an IBM PC/AT® compatible 9-pin connector. It does not work correctly with your workstation.

For printers using RTS/CTS hardware flow control, the following pin-out allows ‘‘full flow control.’’ This cable is required to implement */dev/ttyf**

devices. This cable also supports */dev/ttym** devices. Table 9-3 illustrates the correct pin-out for these devices:

Table 9-3 DB-9 RTS/CTS Flow Control Cable

Function	DB-9-Male	DB25-Male
	1	nc
TXD	2	2
RXD	3	3
RTS*	4	4
CTS*	5	5
	6	nc
GND	7	7
DCD	8	8
DTR	9	20

* RTS and CTS are ignored (optional) if using */dev/ttym** but required if using */dev/ttyf**

Note: This cable can be used with a null modem adapter for printers, however it is recommended that you use this cable exclusively for modem connections. The IBM PC/AT to modem cable (“off the shelf cables") does not work properly with your workstation. For additional information, see the *serial*(7) reference page.

Mini-DIN8 Connector Cabling

Many workstations and servers use the Mini-DIN8 serial port. Check your hardware *Owner's Guide* to see if your system supports this type of connection. Note that the pinout of these Mini-DIN8 connectors is different than that of the DIN connectors on larger systems. These larger systems also have DB-9 connectors that are connected to the same internal port hardware.

For most serial printers you should use a commercially available cable, "Macintosh SE® to Imagewriter1®." This cable uses the normal 3-wire

connection and is used as a */dev/ttyd** device. Table 9-4 shows the pin configuration:

Table 9-4 Mini-DIN8 Serial Cable

Function	Mini-DIN8-Male	DB25-Male
	1	nc
TXD	2	nc
GND	3	3
RXD	4	7
	5	2
DCD*	6	nc
GND	7	20
	8	7

* */dev/ttym** devices should be used with this cable only if the system must notice when the terminal or printer is powered off.

Note: A Macintosh SE cable also has some other pins connected, but they can be ignored.

Configuring and Troubleshooting the BSD LPR Spooler System

Silicon Graphics does not support configuring the BSD lpr print spooler locally (you cannot have the printer physically connected to a Silicon Graphics system). If you want to use the system as a print server, you will need to refer to "Adding a Printer" on page 336.

The purpose of this section is to show by example how to configure any Silicon Graphics computing system so that you will be able to submit and print your files on the BSD print server.

The BSD lpr print spooler will allow you to access printers that are attached to other systems on the network. Please be sure to check the other systems, or contact the System Administrator, to verify the type of spooling system

those systems are using. Generally speaking, if a system has an */etc/printcap* file configured, it is using the BSD lpr print spooling system.

Note: Do not use the Printer Tool (accessed through the System toolchest or from the System Manager menu) to configure a BSD lpr spooling system. Also, do not use the *mknetpr*(1M) command for configuring a network printer. These utilities support only the System V LP Spooling System.

Please verify that the System Administrator of the print server includes your hostname in their */etc/hosts.equiv* file, and that your IP address and hostname appear in their */etc/hosts* file. You will need to add the print servers IP address and hostname to your */etc/hosts* file. If your files (documents) do not print once you have configured the BSD print spooler, please see "Troubleshooting the BSD LPR Spooling System" on page 377. The troubleshooting section will take you to the point where you can see a copy of your document on the print server. If the document disappears from the server's queue without printing, contact the System Administrator of that system for further assistance.

Verifying Installation of the BSD LPR Subsystem

Enter the following command to verify that the BSD lpr print spooling system was properly installed:

```
versions long | grep eoe2.sw.bsdlpr
```

Your output should be:

```
eoe2.sw.bsdlpr etc/init.d/lpd eoe2.sw.bsdlpr etc/printcap
eoe2.sw.bsdlpr etc/rc0.d/K26lpd eoe2.sw.bsdlpr etc/rc2.d/
S61lpd eoe2.sw.bsdlpr usr/bsd/lpq eoe2.sw.bsdlpr usr/bsd/lpr
eoe2.sw.bsdlpr usr/bsd/lprm eoe2.sw.bsdlpr usr/bsd/lptest
eoe2.sw.bsdlpr usr/etc/lpc eoe2.sw.bsdlpr usr/etc/lpd
eoe2.sw.bsdlpr usr/etc/pac eoe2.sw.bsdlpr usr/lib/lpf
eoe2.sw.bsdlpr usr/spool/lpd
```

The BSD spooler is not loaded by default. Check to see if the subsystem is installed. If not, then refer to the *Software Installation Administrator's Guide* and/or your release notes. Most users will have to use the inst(1M) command and then select the manual option to install this subsystem.

Please use the *versions* command (*versions long | grep lpr*) to verify that you have the entire subsystem loaded. Loading only the */etc/printcap* file is not sufficient.

After you verify that the eoe2.sw.bdslpr subsystem is properly installed, you will need to edit the */etc/printcap* file to configure the lpr spooling system. There are no tools to perform this function, so you will need to edit the file manually. The remainder of this article will take you through this process. Please ensure that you format the entries correctly. The */etc/printcap* file expects information in a format similar to the */etc/termcap* file.

Configuring the Printcap File

Before you begin *editing* the */etc/printcap* file, login to your system as **root**.

Note: The printcap file is very sensitive to syntax errors. The name field must begin at the first character on a line of the */etc/printcap* file. The printer names must be separated by pipe symbols (the vertical bar "|"). The name line must be terminated with a colon followed by a backslash (":\"). Make sure that there are no spaces, tabs, or any other character after the backslash.

The definition lines must begin with a tab character followed by a colon (:), followed by the field you are defining, followed by an equal sign. The definition line must end with a colon. See "Printcap Examples".

There is only one name and three definition fields that must be defined. They are:

name	This field should contain all the names the printer will be accessed as. It should include *lp* because, by default, *lpr* looks for *lp* in the name field of the */etc/printcap* file. The names are separated with pipe symbols.
:rm	Remote machine name. This is the name of the system that has the printer physically connected to it.
:rp	Remote printer name. This is the name of the remote printer on the remote system that you are trying to access.
:sd	Spool directory. This is the name of your local spool directory. If you don't use the default directory, */usr/spool/lpd*, you will need to create the directory by using the *mkdir*(1M) command.

Printcap Examples

Following are two examples that will help you to edit the */etc/printcap* file. The first example will show you how to configure the printer configuration file in two lines. The second example will show you how to configure each option of the printcap file on separate lines. There will be explanations of both examples.

Printcap Example 1

```
lp|sleepy|sleepyprinter:\
:lp=:\:rm=snowwhite.story.land:rp=doc:sd=/usr/spool/lpd:
```

This example can access the printer by the following names:

```
lp
sleepy
sleepyprinter
```

The remote system (where the printer is physically attached) is called: `snowwhite.story.land`

The name of the printer on the print server (remote system) is called: `doc`

The local spool directory is called */usr/spool/lpd*

Printcap Example 2

```
lp|sleepy|sleepyprinter:\
 :lp=:\
 :rm=snowwhite.story.land:\
 :rp=doc:\
 :sd=/usr/spool/lpd:
```

This example can access the printer by the following names:

```
lp
sleepy
sleepyprinter
```

The remote system (where the printer is physically attached) is called: `snowwhite.story.land`

The name of the printer on the print server (remote system) is called: `doc`

The local spool directory is called */usr/spool/lpd*

Using the lpr Command to Print

Now that you have the */etc/printcap* file configured, make sure that the daemon is running. Enter the following command:

```
ps -ef | grep lpd
```

Your system should return something similar to:

```
root 195 1 0 11:06:04 ? 0:00 /usr/etc/lpd
root 1293 753 2 13:20:39 ttyq6 0:00 grep lpd
```

The **/usr/etc/lpd** path at the end of the line indicates that the daemon is running. If only one line was returned (`grep lpd`), then start the *lpd* daemon by entering the following command:

```
/usr/etc/lpd
```

Now if you type, *ps -ef | grep lpd*, you should see 2 lines returned to you.

As shown in "Printcap Examples" above, note that the printer could be accessed by three names; `lp`, `sleepy` and `sleepyprinter`. By default, the *lpr* command will look for the field, *lp*, in the */etc/printcap* file. If this is not the first name in the name field, then you will need to do one of two things:

Whenever you enter the *lpr* command you must use the *-P* option to specify a printer name other than *lp*.

```
lpr -Psleepyprinter filename
```

Alternately, in your shell you can set the *environment* variable PRINTER to the name of the printer you wish to use.

In the C shell:

```
setenv PRINTER sleepy
```

In the Bourne shell:

```
PRINTER=sleepy; export PRINTER
```

Now you can enter the command:

```
lpr filename
```

After submitting your request to the printer, you can see if your job has made it to the print spooling queue by entering the following command:

```
lpq
```

Your system should return something similar to:

```
lp is ready and printing
Rank Owner Job Files Total Size
1st nina 113 filename 851 bytes
```

Troubleshooting the BSD LPR Spooling System

If your print request does not make it to the queue, then:

- Check for error messages.
- Double-check the command that you entered.
- Try submitting the */etc/group* file to the queue.

The file you submitted may not be in the proper format for the printer to print your request.

If your print request makes it to the queue and never gets to the print server, then:

- Do you have the print servers IP address and hostname in the */etc/hosts* file?
- Does the print server name match the name in the */etc/hosts* file? Do they match the hostname of the print server?
- Did you get this error message? `Waiting for remote queue to be enabled`.

This usually means that your hostname is not in the print servers **/etc/hosts.equiv** file. If your print request disappears from the queue, and doesn't print or prints incorrect information, then:

1. Become **root** and enter the commands:

   ```
   /usr/etc/lpc stop lp (or your printername)
   lpr /etc/group
   cd /usr/spool/lpd (or your spool directory)
   ```

ls -l

Your system should return something similar to:

```
-rw-rw---- 1 root lp 69 Aug 23 14:02 cfA117tls
-rw-rw---- 1 root lp 227 Aug 23 14:02 dfA117tls
-rwxr----- 1 root lp 0 Aug 23 14:01 lock
-rw-rw-r-- 1 root lp 25 Aug 23 14:46 status
```

2. Check the contents of the control file with the following command:

   cat cfA117tls

   Your system should return something similar to:

```
Htls H the hostname that sent the print request
Proot P the person who sent the request
Jgroup J the jobname
Ctls C class/hostname
Lroot L the person who sent the request
fdfA117tls f name of the file to print
UdfA117tls U name of the file to remove after printing

N/etc/group N the original file name
```

3. Check the copy of the print file.

   We recommend that you use the *more*(1) command just in case your test file is not as short as the */etc/group* file. The *df* file should look exactly like the file you attempted to print. In this case the file dfA117tls should be exactly the same as the */etc/group* file.

   more dfA117tls

   The system should return something similar to:

```
sys::0:root,bin,sys,adm
root::0:root
daemon::1:root,daemon
bin::2:root,bin,daemon
adm::3:root,adm,daemon
mail::4:root
uucp::5:uucp
rje::8:rje,shqer
lp::9:
nuucp::10:nuucp
user::20:
other::995:
demos:*:997:
guest:*:998:
```

Now that you have verified that the request is properly spooling on the local system, check the print server. First, you may need to contact the system administrator of the print server. You will need the root password, and once you enter the *stop* command on their system no one will see their print request print. It will just remain in the queue. Make sure that there are no requests in the queue that are currently printing. It would be ideal if there were no requests currently in the queue.

4. On the print server log in as root and enter the command:

   /usr/etc/lpc stop lp

5. On the local system enter the command:

   /usr/etc/lpc start lp

6. On the print server, *cd* to the spool directory.

   If you do not know where the spool directory is, *cat*(1) or *more*(1) the */etc/printcap* file and look at what is set in the sd: variable.

7. On the print server (after step 6), enter the following command:

   ls -l

   The print server should return something similar to:

   ```
   -rw-r----x 1 root 4 Aug 15 10:27 .seq
   -rw-rw---- 1 root 69 Aug 23 14:02 cfA117tls.csd.sgi.com
   -rw-rw---- 1 root 227 Aug 23 14:02 dfA117tls
   -rwxr------ 1 root 0 Aug 23 14:01 lock
   -rw-rw-r-- 1 root 25 Aug 23 14:46 status
   ```

8. Check the contents of the control file.

   cat cfA117tls.csd.sgi.com

   The print server should return something similar to:

   ```
   Htls H the hostname that sent the print request
   Proot P the person who sent the request
   Jgroup J the jobname
   Ctls C class/hostname
   Lroot L the person who sent the request
   fdfA117tls f name of the file to print
   UdfA117tls U name of the file to remove after printing
   N/etc/group N the original file name
   ```

9. Examine the *df** file by entering the following command:

   more dfA117tls

The system should return something similar to:

```
sys::0:root,bin,sys,adm
root::0:root
daemon::1:root,daemon
bin::2:root,bin,daemon
adm::3:root,adm,daemon
mail::4:root
uucp::5:uucp
rje::8:rje,shqer
lp::9:
nuucp::10:nuucp
user::20:
other::995:
demos:*:997:
guest:*:998:
```

The *df* file should look exactly like the file you attempted to print. In this case the print servers *dfA117tls* file should be exactly the same as the *dfA117tls* file that was on your system.

10. On the print server enter the following command:

    `/usr/etc/lpc start lp`

    Your file should now print on the printer. It should look exactly like the output of the *more* command. If it doesn't then contact the System Administrator of the print server.

Chapter 10

Terminals and Modems

Chapter 10 covers information on installing serial terminals and modems. You can also use the graphical System Manager to perform these tasks conveniently if your system supports graphics. The System Manager is described in the Personal System Administration Guide. Tasks covered in this chapter include:

- *Installing and maintaining serial terminals.*
- *Installing and configuring modems.*
- *Serial device cabling requirements.*

Chapter 10

Terminals and Modems

This chapter describes some of the software and hardware considerations for terminals and modems. This information also applies to most generic serial devices and specialized devices such as the Dial & Button Box and the Spaceball. For more information on these devices, please contact Silicon Graphics.

The TTY system supports serial communication between users and serial devices, such as modems and terminals. ''Dumb'' serial printers are also connected using the TTY system, but they are covered in Chapter 9, "Administering Printers." This chapter tells you how to administer the TTY system with respect to terminals and modems. It includes information about:

- Connecting an ASCII terminal or modem. See "Adding a Terminal or Modem" on page 384.
- Information on the TTY subsystem in general. See "The TTY System" on page 399.
- Serial port cabling and pin signals for the standard serial ports. See "Serial Ports" on page 402.

To connect peripherals that are not covered in this chapter, see the documentation that accompanies the peripheral.

Terms

The following terms are used when describing serial devices and the TTY system:

TTY — Derived from the near-classic abbreviation for teletypewriter, the term covers the whole area of access between the IRIX system and peripheral devices, including the system console. It shows up in commands such as

	getty(1M) and *stty*(1), in the names of device special files such as */dev/ttyd1*, and in the names of files such as */etc/gettydefs*, which 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, and 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).

Adding a Terminal or Modem

Most workstations have some form of external *tty* devices attached. The following information describes the tasks necessary to add a terminal or a modem to your workstation.

Attaching an ASCII Terminal

This section describes the procedures for connecting and configuring an ASCII terminal. The *diagnostics terminal* is the ASCII terminal connected to the port labeled **1** on the I/O panel of a server. The messages produced by the power-on diagnostics appear on the screen of this terminal.

Connecting the Terminal Hardware

Connect the ASCII terminal to the port labeled **1** on the I/O panel. See "Cabling the Serial Ports" on page 404 for information about port cabling and pin signals.

Configuring the Terminal Software

This section tells you how to configure IRIX software to use an ASCII terminal with your workstation.

The utilities described in this section are distributed as part of the *eoe2.sw.terminfo* package. You must have installed this package on your system in order to use these utilities. See the *Software Installation Guide* for information and specific instructions on installing this package.

The */usr/lib/terminfo* directory contains files that describe different terminal models, their capabilities, and how they operate. For most ASCII terminal models, you do not need to edit this database. Follow the procedure below to see whether the information on your terminal model appears in the directory */usr/lib/terminfo*. If your terminal is not in the database, or if it does not work properly after you have configured the software, you must write a terminal description. See the optional *IRIX Programmer's Guide* and the *tset*(1), *stty*(1), and *terminfo*(4) reference pages.

The directory */usr/lib/terminfo* is divided into numerical and alphabetical subdirectories. Each subdirectory contains entries for terminals whose names begin with that character. For example, */usr/lib/terminfo/v* contains the entry for the Visual 50. The entry name listed in the subdirectory is `v50am`.

To find the entry name for your terminal and to configure software for an ASCII terminal, follow these steps:

1. Log in as **root** or become the superuser by entering the *su* command.
2. To change directories to */usr/lib/terminfo*, type:

   ```
   cd /usr/lib/terminfo
   ```

3. To find the name of your terminal, issue a *grep* command with a string that you suspect could make up part of your terminal name. If this fails, examine the subdirectories of */usr/lib/terminfo*, which contain all the terminal entries.

   ```
   ls -R | fgrep string
   ```

4. Once you know the terminal name as it appears in */usr/lib/terminfo*, issue the *infocmp* command to find out the model name of your terminal. For example, for a Visual 50, issue this command:

   ```
   infocmp -I v50am
   ```

   You see a display that begins with this line:

   ```
   v50am|visual50 (v50 emulation) with automatic margins,
   ```

   The data in the first field (v50am) is the model name of your terminal. In the next step, you'll enter this model name in the */etc/ttytype* file.

5. Edit */etc/ttytype*.

   This file tells which type of terminal is connected to which port. In the line that contains the port you are using, replace `du` with the model name of your terminal. The question mark (?) at the beginning of the line in */etc/ttytype* causes *tset* to prompt for the kind of terminal you are using when you log on through that port. An */etc/ttytype* might look like this example:

   ```
   iris-ansi systty
   ?v50am    ttyd1
   ?v50am    ttyd2
   ?v50am    ttyd3
   ?v50am    ttyd4
   ?v50am    ttyd5
   ?v50am    ttyd6
   ?v50am    ttyd7
   ?v50am    ttyd8
   ?v50am    ttyd9
   ?v50am    ttyd10
   ?v50am    ttyd11
   ?v50am    ttyd12
   ```

   You normally call *tset* in your login startup script (*.login* or *.profile*). *tset* commands use information from */etc/ttytype* and */usr/lib/terminfo* to initialize the terminal. These files also provide information on setting up environment variables so that editors and other programs know how to communicate with the terminal. See *tset*(1).

6. Edit */etc/inittab* so that you can log in to the computer ports. This is a sample from an */etc/inittab* file:

```
t1:23:respawn:/etc/getty -s console ttyd1 co_9600 # port 1
t2:23:off:/etc/getty -N ttyd2 co_9600 # port 2
t3:23:off:/etc/getty -N ttyd3 co_9600 # port 3
t4:23:off:/etc/getty -N ttyd4 co_9600 # port 4
```

Here are two example entries, with an explanation of each field in the entries:

```
t1:23:respawn:/etc/getty -s console ttyd1 co_9600
```

```
t2:23:off:/etc/getty -N ttyd2 co_9600
```

t1 t2 uniquely identifies the entry.

23 defines the run level in which this entry is to be processed. A 23 means this entry is to be processed in run levels two and three.

off means never perform the action on the process field of *init*.

respawn means perform the action on the process field of *init*. See *inittab*(4) for a description of all possible actions.

/etc/getty -s console ttyd1 co_9600
runs the *getty* process on the port labeled **1** at the baud rate and with the options specified in the co_9600 entry in the */etc/gettydefs* file. The **-s console** option instructs *getty* that the login shell generated is a system console, and therefore receives system error messages.

/etc/getty -N ttyd2 co_9600
runs the *getty* process on the port labeled **2** at the baud rate and with the options specified in the co_9600 entry in the */etc/gettydefs* file. The **-N** option instructs *getty* to honor the presence of the */etc/nologin* file, which disallows remote logins over the network.

To enable you to log in to the terminal connected to the port labeled **2,** find this line:

```
t2:23:off:/etc/getty -N ttyd2 co_9600
```

Change it to:

```
t2:23:respawn:/etc/getty -N ttyd2 co_9600
```

7. */etc/inittab* refers to */etc/gettydefs* for information about the terminal baud rate. In the example from */etc/inittab* above, co_9600 refers to the name of an entry in */etc/gettydefs*; it defines a 9600 baud setting. If you don't plan to run the terminal at 9600 baud, replace co_9600 in */etc/inittab* with the correct entry name from */etc/gettydefs*. To see which entries are defined in */etc/gettydefs*, examine the file, or see "Checking Line Settings Using IRIX Shell Commands" on page 400. To make a new entry, see "Creating and Testing Line Settings" on page 401. The entries in */etc/gettydefs* look like this:

```
dx_9600# B9600 # B9600 SANE TAB3 HUPCL #\r\n\n$HOSTNAME
login: #dx_9600
dx_4800# B4800 # B4800 SANE TAB3 HUPCL #\r\n\n$HOSTNAME
login: #dx_4800
dx_2400# B2400 # B2400 SANE TAB3 HUPCL #\r\n\n$HOSTNAME
login: #dx_2400
dx_1200# B1200 # B1200 SANE TAB3 HUPCL #\r\n\n$HOSTNAME
login: #dx_1200
du_1200# B1200 # B1200 SANE TAB3 HUPCL #\r\n\n$HOSTNAME
login: #du_300
du_300#  B300  # B300  SANE TAB3 HUPCL #\r\n\n$HOSTNAME
login: #du_2400
du_2400# B2400 # B2400 SANE TAB3 HUPCL #\r\n\n$HOSTNAME
login: #du_1200
```

The entries beginning with `dx` are typically used for terminals; those beginning with `du` are typically for modems. See *gettydefs*(4) for more information on the fields of each entry.

When the terminal is powered on, the workstation sends a login prompt to the terminal screen. Press **<Enter>** if the login prompt doesn't appear. If the default line speed set in */etc/inittab* is incorrect, the prompt may be garbled or may not appear.

8. Inform *init* of the change to */etc/inittab*, and start a *getty* process for the port:

```
telinit q
```

Setting Terminal Options

The TTY system described thus far establishes a basic style of communication between the user's terminal and the IRIX operating system. Once a user has successfully logged in, he or she may prefer terminal options other than the default set.

The *stty*(1) command controls terminal options. Many users add an *stty* command to their *.profile* or *.login* files so the options they want are automatically set as part of the *login* process. Here is an example of a simple *stty* command:

```
stty cr0 nl0 echoe -tabs erase '^H'
```

The options in the example mean:

`cr0 nl0`	No delay for carriage return or new line. 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 Ctrl-H when the **<backspace>** key is pressed. Specifying this option makes **<backspace>** useful.

Attaching a Modem

Complete information on installing and configuring a modem is available in the *Personal System Administration Guide* and it is recommended that you use that information as your primary guide. The information provided here is sufficient to install a modem using IRIX shell commands. See "Cabling the Serial Ports" on page 404 for information about modem cabling.

Silicon Graphics supports Hayes 2400 and Telebit modems. This section gives you the information you need to set up only these two types of

modems; it does not provide all the in-depth information necessary to set up a "Hayes- compatible" modem.

You can use the standard system software to set up a modem in three different ways:

- As a dial-in modem, other users can call your modem to log in to your system.
- As a dial-out modem, you use the modem to call a remote modem to log in to the system to which the remote modem is connected.
- As a dial-in/dial-out modem, other users can call your modem, and you can call other modems.

To set up a modem, you must first physically connect the modem to a serial port on your system, and connect the modem to a telephone jack. See your *Owner's Guide* or "Cabling the Serial Ports" on page 404 for cabling information.

Note: Do not connect your system to the modem with a cable for an IBM-PC/AT or for an Apple Macintosh as it will not function correctly.

The *eoe2.sw.uucp* subsystem must be installed on your system. It is shipped with your IRIX system software distribution on tape or CD, but it is not installed by default. You can determine whether *eoe2.sw.uucp* is installed by using the command:

```
versions
```

and checking the *versions* output for the following line:

```
I eoe2.sw.uucp uucp utilities
```

If the line above is not present, *eoe2.sw.uucp* is not installed. See the *IRIS Software Installation Guide* for instructions on installing a subsystem and install *eoe2.sw.uucp*.

Use the documentation that came with your modem to determine the modem's baud rate. Next, determine what use you want to make of the

modem and go to the appropriate following section in this chapter. There are three options:

- "Turning On Dial-In Modem Software" on page 391 for modems that answer only incoming calls.
- "Turning On Dial-Out Modem Software" on page 393 for modems that only call out.
- "Turning On Dial-In/Dial-Out Modem Software" on page 395 for modems that serve both incoming and outgoing calls.

Turning On Dial-In Modem Software

You enable dial-in modem software by editing the */etc/inittab* file, and running several commands. Perform the following steps in order:

1. Log in as **root**.
2. Edit */etc/inittab* to turn off the port you will be using so you can configure it correctly.

   Find the line for your port. The line begins with *tportnumber*, where *portnumber* is the number of a serial port. For example, if you connected your modem to port 2, look for a line similar to the following:

   ```
   t2:23:respawn:/etc/getty -N ttyd2 co_9600 #port 2
   ```

   Change *respawn* to *off*. This tells the system not to restart the */etc/getty* program when it exits. If it is already off, continue to the next step. The line should look like this when you are finished:

   ```
   t2:23:off:/etc/getty -N ttyd2 co_9600 #port 2
   ```

3. Write and exit the */etc/inittab* file.
4. Inform init about the change to *inittab* with the following command:

   ```
   telinit q
   ```

5. If you have a Hayes 2400 modem, use the *fix-hayes* command and specify a dial-in modem (**-i**) and the port number to which the modem is connected. For example, if the modem is connected to port 2, enter the command:

   ```
   /etc/uucp/fix-hayes -i d2
   ```

If you have a Telebit modem, use the *fix-telebit* command and specify a dial-in modem (**-i**), the model number of the modem (*tb+*, *t1000*, *t1600*, *t2500*) and the port number to which the modem is connected. (Older models of Telebit modems should first be reset with a paperclip.)

For example, if the modem is connected to port 2, enter the following command:

```
/etc/uucp/fix-telebit -i -m t2500 d2
```

6. Edit the */etc/inittab* file again so the system will recognize which port will be used for the modem (as opposed to a terminal). This editing also sets the correct baud rate on the port and enables user logins through the port.

   Find the line for the port to which the modem is connected. The line begins with *tportnumber*, where *portnumber* is the number of a serial port. For example, if you connected your modem to port 2, look for a line similar to the following:

   ```
   t2:23:off:/etc/getty -N ttyd2 co_9600 #port 2
   ```

   If you are setting up a Hayes 2400 modem, make the following changes to this line:

   - change *off* to *respawn*
   - change *ttyd2* to *ttym2*
   - change *co_9600* to *du_2400* (this change lets the modem answer and connect at 2400, 1200, and 300 baud, but not 4800 or 9600 baud)
   - change the comment at the end of the line to reflect the fact that this port is now used for a modem

   For example, if you connected a Hayes 2400 modem to serial port 2, and found a line similar to this line in */etc/inittab*:

   ```
   t2:23:off:/etc/getty -N ttyd2 co_9600 #port 2
   ```

   You would change it to read:

   ```
   t2:23:respawn:/etc/getty -N ttym2 du_2400 #Modem
   ```

   If you are setting up a Telebit modem, make the following changes to the line in */etc/inittab*:

   - change *off* to *respawn*
   - change *ttyd2* to *ttyf2*

- change *co_9600* to *dx_19200*
- change the comment at the end of the line to reflect the fact that this port is now used for a modem

For example, suppose you connected a Telebit modem to serial port 2, and found a line similar to this one:

```
t2:23:off:/etc/getty -N ttyd2 co_9600 #port 2
```

You would change it to read:

```
t2:23:respawn:/etc/getty -N ttyf2 dx_19200 #Modem
```

7. Write and exit the */etc/inittab* file.
8. Inform *init* about the change to */etc/inittab* with the following command:

   ```
   telinit q
   ```

To test the dial-in setup, have someone try to dial in to the system through this modem. If the login prompt is garbled or does not appear, the caller should press the **<Break>** key and try again. If it still does not work, check all cable connections and make sure you edited the files correctly.

Turning On Dial-Out Modem Software

You can turn on dial-out modem software by editing the */etc/uucp/Devices* file. You also have to run the following commands:

1. Log in as **root**
2. Edit */etc/inittab* to turn off the port you will be using so you can configure it correctly. Find the line for your port. The line begins with *tportnumber*, where *portnumber* is the number of a serial port. For example, if you connected your modem to port 2, look for a line similar to the following:

   ```
   t2:23:respawn:/etc/getty -N ttyd2 co_9600 #port 2
   ```

   Change the *respawn* to *off*. If it is already off, continue to the next step. When you are finished, the line should look like this:

   ```
   t2:23:off:/etc/getty -N ttyd2 co_9600 #port 2
   ```

3. Write and exit the */etc/inittab* file.
4. Inform *init* about the change to *inittab* with the command:

   ```
   telinit q
   ```

5. If you have a Hayes 2400 modem, use the *fix-hayes* command and specify a dial-out modem (**-o**) and the port number to which the modem is connected. For example, if you connected the modem to serial port 2, enter the command:

   ```
   /etc/uucp/fix-hayes -o d2
   ```

   If you have a Telebit modem, use the *fix-telebit* command and specify a dial-out modem (**-o**), the model number of the modem (*tb+*, *t1000*, *t1600*, *t2500*) and the port number to which the modem is connected. (Older models of Telebit modems should first be reset with a paperclip.) For example, if the modem is connected to port 2, enter the command:

   ```
   /etc/uucp/fix-telebit -o -m t2500 2
   ```

6. Edit the */etc/uucp/Devices* file to specify the port to which the modem is connected and to set the correct baud rate. Add one or more lines to specify the port number and baud rate for the modem. Also, add a "Direct" line for use in debugging problems. The lines have the following forms:

   ```
   ACU tty<hi/low> <portnumber> null baudrate 212 x dialer
   Direct ttyd<portnumber> - baud direct
   ```

   If your modem runs at 2400 baud or lower, the <*hi*/*low*> field should be **ttym**. If it runs faster than 2400 baud, the <*hi*/*low*> field should be **ttyf**. The replacement for *portnumber* is the number of the serial port to which the modem is connected. The *baud* variable is the baud rate at which you will run the modem.

   For example, consider a Hayes 2400 modem connected to serial port 2 which can accept and send data at 2400, 1200, and 300 baud. You would add the following lines to the *Devices* file:

   ```
   ACU ttym2 null 2400 212 x hayes24
   ACU ttym2 null 1200 212 x hayes24
   ACU ttym2 null 300 212 x hayes24
   Direct ttyd2 - 1200 direct
   ```

   For a Telebit modem connected to serial port 2 and running only at 19200 baud, add the following lines:

   ```
   ACU ttyf2 null 19200 212 x telebit
   Direct ttyd2 - 19200 direct
   ```

   Save and exit the *Devices* file.

7. Change the ownership of the device files. For example, if you are using serial port 2, give the following command (substituting the appropriate *tty* device as determined above):

   chown uucp /dev/tty[*dfm***]2**

When you have finished the above process, try to dial out through this modem to verify the dial-out setup. For more information on dialer options, see the file */etc/uucp/Dialers*.

Turning On Dial-In/Dial-Out Modem Software

You turn on dial-in/dial-out modem software by combining the processes for enabling a dial-in and a dial-out modem. You must edit two files: */etc/uucp/Devices* and */etc/inittab*. You also have to run several shell commands. Perform the following steps:

1. Log in as **root**
2. Edit */etc/inittab* to turn off the port you will be using so you can configure it correctly. Find the line for your port. The line begins with *tportnumber*, where *portnumber* is the number of a serial port. For example, if you connected your modem to port 2, look for a line similar to the following:

   ```
   t2:23:respawn:/etc/getty -N ttyd2 co_9600 #port 2
   ```

   Change the *respawn* to *off*. If it is already off, continue to the next step. When you are finished, the line should look like this:

   ```
   t2:23:off:/etc/getty -N ttyd2 co_9600 #port 2
   ```

3. Write and exit the */etc/inittab* file.
4. Inform init about the change to inittab with the command:

   telinit q

5. If you have a Hayes 2400 modem, use the *fix-hayes* command and specify a dial-in/dial-out modem (**-io**) and the port number to which the modem is connected. For example, if you connected the modem to serial port 2, enter the command:

   /etc/uucp/fix-hayes -io d2

If you have a Telebit modem, use the *fix-telebit* command and specify a dial-in/dial-out modem (**-io**), the model number of the modem (*tb+*, *t1000*, *t1600*, *t2500*) and the port number to which the modem is connected. (Older models of Telebit modems should first be reset with a paperclip.) For example, if the modem is connected to port 2, enter the command:

```
/etc/uucp/fix-telebit -io -m t2500 2
```

6. Edit the */etc/uucp/Devices* file to specify the port to which the modem is connected and to set the correct baud rate. Add one or more lines to specify the port number and baud rate for the modem. Also, add a "Direct" line for use in debugging problems. The lines have the following forms:

   ACU tty<*hi/low*> <*portnumber*> null *baudrate* 212 x *dialer*

   Direct ttyd<*portnumber*> - *baud* direct

   If your modem runs at 2400 baud or lower, the <*hi/low*> field should be **ttym**. If it runs faster than 2400 baud, the <*hi/low*> field should be **ttyf**. The replacement for *portnumber* is the number of the serial port to which the modem is connected. The *baud* variable is the baud rate at which you will run the modem.

   For example, consider a Hayes 2400 modem connected to serial port 2 which can accept and send data at 2400, 1200, and 300 baud. You would add the following lines to the *Devices* file:

   ```
   ACU ttym2 null 2400 212 x hayes24
   ACU ttym2 null 1200 212 x hayes24
   ACU ttym2 null 300 212 x hayes24
   Direct ttyd2 - 1200 direct
   ```

   For a Telebit modem connected to serial port 2 and running only at 19200 baud, add the following lines:

   ```
   ACU ttyf2 null 19200 212 x telebit
   Direct ttyd2 - 19200 direct
   ```

   Finally, save and exit the *Devices* file.

7. Change the ownership of the device files. For example, if you are using serial port 2, give the following command (substituting the appropriate *tty* device as determined above):

   chown uucp /dev/tty[*dfm***]2**

8. Edit the */etc/inittab* file again so the system will recognize which port will be used for the modem (as opposed to a terminal). This editing also sets the correct baud rate on the port and enables user logins through the port.

   Find the line for the port to which the modem is connected. The line begins with *tportnumber*, where *portnumber* is the number of a serial port. For example, if you connected your modem to port 2, look for a line similar to the following:

   ```
   t2:23:off:/etc/getty -N ttyd2 co_9600 #port 2
   ```

   If you are setting up a Hayes 2400 modem, make the following changes to this line:

   - change *off* to *respawn*
   - change *ttyd2* to *ttym2*
   - change *co_9600* to *du_2400* (this change lets the modem answer and connect at 2400, 1200, and 300 baud, but not 4800 or 9600 baud)
   - change */etc/getty* to */usr/lib/uucp/uugetty*
   - add some options to *uugetty*
   - change the comment at the end of the line to reflect the fact that this port is now used for a modem

   For example, suppose you connected a Hayes 2400 modem to serial port 2, and found a line similar to this line in */etc/inittab*:

   ```
   t2:23:off:/etc/getty -N ttyd2 co_9600 #port 2
   ```

   You would change it to read (all on the same line):

   ```
   t2:23:respawn:/usr/lib/uucp/uugetty -Nt 60 -d8 -
   ihayes24in, conn ttym2 du_2400 #Modem
   ```

   If you are setting up a Telebit modem, make the following changes to the line in */etc/inittab*:

   - change *off* to *respawn*
   - change *ttyd2* to *ttyf2*
   - change *co_9600* to *dx_19200*
   - change */etc/getty* to */usr/lib/uucp/uugetty*
   - Add some options to *uugetty*

- change the comment at the end of the line to reflect the fact that this port is now used for a modem

For example, if you connected a Telebit modem to serial port 2, find a line similar to this line:

```
t2:23:off:/etc/getty -N ttyd2 co_9600 #port 2
```

If you are using a *T1000* or *TB+*, change the line to read (all on one line):

```
t2:23:respawn:/usr/lib/uucp/uugetty -Nt 60 -itelebitin,
conn ttyf2 dx_19200 #Modem
```

If you are using a *T1600*, use *t16in* instead of *telebitin* in the above command line. If you are using a *T2500*, use *t25in* instead of *telebitin* in the above command line.

9. Write and exit the */etc/inittab* file.
10. Inform *init* about the change to */etc/inittab* with the following command:

    ```
    telinit q
    ```

When you have finished the above process, try to dial out through the modem to verify the dial-out setup. Then, to test the dial-in setup, have someone try to dial in to the system through this modem. If the login prompt is garbled or does not appear, the caller should press the **<BREAK>** key and try again. If the connection still does not work, check all cable connections and make sure you edited and saved all files correctly.

Dialing Out to Another Modem

The *cu*(1) utility dials your modem. Before you attempt to dial out, make sure your local modem is connected to both your system and a working telephone line. Also, make sure the modem is turned on. The *cu* syntax to dial the number *1-800-555-1234* is simply:

```
cu 18005551234
```

Refer to the *cu*(1) reference page for complete information on *cu*. If everything is working, you should hear the modem dialing, and after a short time you should see the message on your console:

```
Connected
```

If no login prompt is displayed, press **<Return>** once. If a prompt is displayed, continue and log in to the remote system. To disconnect, press the **<Return>** key, the tilde key (~), a period (.), and the **<Return>** key again.

If there are any problems with the dial-up process, you may want to use the (**-d**) option to *cu* to instruct the system to print diagnostic messages to your system console. The syntax to force diagnostic messages is.

```
cu -d 18005551234
```

The TTY System

This section covers the following topics:

- the terms used in discussing TTY management
- how the TTY system works
- how to check line settings
- how to create new line settings and hunt sequences
- how to modify TTY line characteristics

A series of four processes *(init*(1M), *getty*(1M), *login*(1), and one of *sh*(1) or *csh(1))* connects a user to the IRIX 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 *getty*(1M) reference page.

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) reference page 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 if 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.

/bin/sh executes */etc/profile* and then executes the user's *.profile*, if it exists. *.profile* or *.login* often contain *stty* commands that reset terminal options that differ from the defaults. */bin/csh* executes */etc/cshrc*, *.cshrc*, and *.login*. The connection between the user and the IRIX system has now been made.

Checking Line Settings Using IRIX Shell Commands

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

The following example shows a few lines from a *gettydefs* file:

```
console# B9600 # B9600 SANE TAB3 #\r\n\n$HOSTNAME console
login: #console
co_9600# B9600 # B9600 SANE TAB3 #\r\n\n$HOSTNAME login:
#co_4800
co_4800# B4800 # B4800 SANE TAB3 #\r\n\n$HOSTNAME login:
#co_2400
co_2400# B2400 # B2400 SANE TAB3 #\r\n\n$HOSTNAME login:
#co_1200
co_1200# B1200 # B1200 SANE TAB3 #\r\n\n$HOSTNAME login:
#co_300
co_300# B300 # B300 SANE TAB3 #\r\n\n$HOSTNAME login:
#co_9600
dx_9600# B9600 # B9600 SANE TAB3 HUPCL #\r\n\n$HOSTNAME
login: #dx_9600
```

These entries 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, press the **<Break>** key to force *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).

Creating and Testing Line Settings

Create new lines for the *gettydefs* file by following the example shown above. 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. Any unrecognized modes or improperly constructed entries are reported.

Modifying Line Characteristics

You can modify TTY line characteristics using an IRIX editor, such as *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 as follows:

identification:level:action:process

The four colon-separated fields are as follows:

identification	A unique one or two 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) reference page for complete information).
process	The shell command to be executed.

/etc/inittab contains several entries that spawn *getty* processes. The following example is a selection of such entries from a sample */etc/inittab*:

```
t1:23:respawn:/etc/getty -s console ttyd1 co_9600
t2:23:respawn:/etc/getty ttyd2 co_9600
```

There are at least three things you might want to do to an *inittab* entry for a TTY line:

- Change the action. Two actions that apply to TTY lines are ''respawn'' and ''off'' (see the *inittab*(4) reference page for complete information on this field).
- Add or change arguments to */etc/getty* in the process field. A frequently used argument is **-tnn**. 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.
- Add or change comments. You can start your comments after a pound sign (#), and insert comments after a semi-colon (;) to end the command.

Serial Ports

This section outlines the serial support on IRIS workstations and servers and provides instructions for connecting peripheral devices to the serial ports. For information on the optional serial ports, see the document that accompanies that hardware option.

Defining the Serial Interface

The IRIS workstation or server provides an RS-232 (or RS-423) or a DB-9 (9 pin) compatible serial interface. Additionally, some systems are equipped with the Mini-DIN8 serial interface. These ports do not lay out the pins in a conventional pattern such as DB-9 or RS232; instead, the plug is cylindrical. Adapter cables are available commercially or through Silicon Graphics to connect Mini-DIN8 ports with DB-9 and RS232 ports. Consult your system's *Owner's Guide* for complete information on your system's serial ports. All serial data cables that you connect to the computer should be shielded. The computer can easily drive and receive signals on a 50-foot cable, and it typically drives and receives signals on a cable up to 200 feet long.

There are two types of serial interface equipment available: *Data Terminal Equipment* (*DTE*) and *Data Communications Equipment (DCE)*. The primary difference between DTE and DCE is the designation of several pins on the connector. For example, DTEs output on pin 2 and input on pin 3. DCEs output on pin 3 and input on pin 2. You can connect a DTE interface directly to a DCE interface.

To connect either a DCE to a DCE, or a DTE to a DTE, use a *null modem* cable. A null modem cable has the wires to pins 2 and 3 swapped in one connector and may have other swapped wires as well. A signal on pin 2 at one end appears on pin 3 at the other end, and vice versa.

The serial ports for IRIS workstations and servers are all configured as DTE. Most terminals are also configured as DTE. Therefore, to connect a terminal to a workstation that has RS232 or DB-9 connectors, use a cable that has pins 2 and 3 swapped in one connector. To connect a modem to the workstation, use a cable that connects each pin of the serial port to the corresponding pin of the modem. No signals need to be swapped. Note that on a modem cable, if pins 5 and 7 become swapped, any process attempting to use the modem may become hung up. Some commercially available personal computer serial cables swap pins 5 and 7. Ensure that your cable is a straight connection before using it with your IRIS. Connect other peripheral devices according to the configuration data provided with the device.

Silicon Graphics provides three kinds of special files, which determine which driver is used on each port. The special files beginning with *ttyd* are used for devices such as terminals; the files *ttymX* are used for modems; and *ttyfX* files are used for flow control to devices that use hardware flow control

on the RTS and CTS pins. Some workstations use “Mini-DIN8” style ports. These ports do not lay out the pins in a conventional pattern such as DB9 or RS232; instead, the plug is cylindrical. Adapter cables are available commercially or through Silicon Graphics to connect Mini-DIN8 ports with DB9 and RS232 ports.

Cabling the Serial Ports

This section describes the cables typically used to connect the computer to terminals and modems. The serial ports are designed to connect directly to Data Communications Equipment (DCE) devices such as modems through a serial cable. Each wire on this cable connects the same pin of the computer to the device, that is, it has a pin-to-pin correspondence. Connecting to Data Terminal Equipment (DTE) devices such as terminals requires a different cable arrangement: a *null modem* cable. A null modem cable has the wires for pin 2 and pin 3 swapped in one connector, and may have other wires swapped as well.

DB-9 Serial Connector Cabling

For most “dumb" terminals SGI recommends the following cable. This cable uses the normal 3-wire connection and is used as a `/dev/ttyd*` device. Table 10-1 shows typical cabling:

Table 10-1 DB-9 Serial Terminal Cable

Function	DB9-Male	DB25-Male
	1	nc*
TXD	2	3
RXD	3	2
	4	nc
	5	nc
	6	nc
GND	7	7

Table 10-1 (continued) DB-9 Serial Terminal Cable

Function	DB9-Male	DB25-Male
DCD**	8	20
	9	nc

*nc stands for "no connect," meaning the wire is not used.

DCD is only used with */dev/ttym devices if the system must notice when the terminal or printer powers off. Normally it is not used.

Note: Do not use a cable designed for an IBM PC/AT® compatible 9-pin connector. It does not work correctly with your workstation.

Table 10-2 lists the pin definitions for an example of a 9-pin (DB-9) null modem cable. Connect pins that are shown separated by commas in the DTE Device column together with a wire or jumper at that end of the cable. Then connect these joined pins to the pin shown in the 9-pin column at the 9-pin end of the cable.

Table 10-2 Pin Definitions for a Null Modem Cable

9-Pin	DTE Device	Signal
2	3	Transmit and Receive Data
3	2	Receive and Transmit Data
4	5	Request To Send, Clear To Send
5	4	Clear To Send, Request To Send
7	7	Signal Ground
9	6,8	Data Terminal Ready, Data Set Ready and Data Carrier Detect
8	20	Data Carrier Detect, Data Terminal Ready

Most terminals do not require the various handshaking lines such as *Clear To Send* or *Data Set Ready* and work with a three-wire null modem cable. To make one of these, you simply need to swap the signals for pins 2 and 3, and you need to connect pin 7 of the computer to pin 7 of the terminal. Table 10-3 lists the pin definitions for a three-wire null modem cable.

Table 10-3 Sample Three-wire Null Modem Terminal Cable

Computer	Terminal	Signals
2	3	Transmit Data
3	2	Receive Data
7	7	Signal Ground

Note that the pinout of these DB-9 connectors is different from that of the DIN connectors that may be next to them. These DIN connectors also have different pinouts from the Mini-DIN8 connectors used on some systems and documented in the next section. The DB-9 and DIN connectors are connected to the same internal port hardware.

For modem devices using RTS/CTS hardware flow control, the following pin-out will allow "full flow control." This cable is required to implement */dev/ttyf** devices. This cable also supports */dev/ttym** devices. Table 10-4 illustrates the correct pin-out for these devices:

Table 10-4 DB9 RTS/CTS Modem Control Cable

Function	DB9-Male	DB25-Male
	1	nc
TXD	2	2
RXD	3	3
RTS*	4	4
CTS*	5	5
	6	nc
GND	7	7

Table 10-4 (continued) DB9 RTS/CTS Modem Control Cable

Function	DB9-Male	DB25-Male
DCD	8	8
DTR	9	20

* RTS and CTS are ignored (optional) if using */dev/ttym** but required if using */dev/ttyf**

Note: This cable can be used with a null modem adapter for terminals and printers (see Table 10-2 above); however, it is recommended that you use this cable exclusively for modem connections. The IBM PC/AT to modem cable ("off the shelf cables") will not work properly with your workstation. For additional information see the *serial*(7) reference page.

Mini-DIN8 Serial Connector Cabling

There are three basic cable configurations for the Mini-DIN8 serial ports. See your *Owner's Guide* to determine if you have a Mini-DIN8 port on your workstation or server. Depending on the cables used, some functionality may be sacrificed. Note that the pinout of these Mini-DIN8 connectors is different from that of the DIN connectors on larger systems. These larger systems also have DB-9 connectors that are connected to the same internal port hardware.

For most dumb terminals you should use a commercially available cable, "Macintosh SE® to Imagewriter1®." This cable uses the normal 3-wire connection and is used as a */dev/ttyd** device. Table 10-5 shows the pin configuration:

Table 10-5 Mini-DIN8 Serial Terminal Cable

Function	Mini-DIN8-Male	DB25-Male
	1	nc
	2	nc
TXD	3	3
GND	4	7

Table 10-5 (continued) Mini-DIN8 Serial Terminal Cable

Function	Mini-DIN8-Male	DB25-Male
RXD	5	2
	6	nc
DCD*	7	20
GND	8	7

* */dev/ttym** devices should be used with this cable only if the system must notice when the terminal or printer is powered off.

Note: A Macintosh SE cable also has some other pins connected but they can be ignored.

For modem devices using RTS/CTS hardware flow control, the following pin-out allows "full flow control." This cable is required to implement */dev/ttyf** devices. This cable also supports */dev/ttym** devices. Table 10-6 shows the pinout:

Table 10-6 Mini-DIN8 RTS/CTS Modem Cable

Function	Mini-DIN8-Male	DB25-Male
DTR	1	20
CTS*	2	5
TXD	3	2
GND	4	7
RXD	5	3
RTS*	6	4
DCD	7	8
GND	8	7

* RTS and CTS are ignored (optional) if using */dev/ttym** but required if using */dev/ttyf**

Note: This cable is available from Silicon Graphics. Contact your sales representative or SGI Express. This cable can be used with a null modem adapter for terminals and printers (see Table 10-2); however, you should use this cable exclusively for modem connections. The commercially available cable that connects a Macintosh SE to a modem does not work properly with SGI software.

Dial & Button Box and Spaceball Serial Cabling

For SGI peripherals such as the Dial & Button Box and Spaceball, the following Mini-DIN8 to SGI 9 Pin connector cable is available. Table 10-7 shows the pinout for this application:

Table 10-7 SGI Peripheral Cable

Function	Mini-DIN8-Male	DB25-Male
DTR	1	9
CTS	2	5
TXD	3	2
GND	4	7
RXD	5	3
RTS	6	4
DCD	7	8
GND	8	7

Note: All pins not shown are "no connects" (nc). For additional information, see the *serial*(7) reference page.

Chapter 11

Administering the Cadmin Object System

Chapter 11 describes the administration of the cadmin object system. This object system provides utilities for system administration in the IndigoMagic user environment. Topics covered include:

- *A system overview of cadmin.*
- *Information on manipulating the cadmin system.*
- *Troubleshooting the objectserver daemon.*
- *Troubleshooting the directoryserver daemon.*

Chapter 11

Administering the CADMIN Object System

This chapter describes the administration of the *cadmin* object system. This object system provides utilities for system administration in the Indigo Magic user environment, which is available only on graphics workstations. If you do not have a graphics workstation, or if you do not have this environment enabled on your workstation, you cannot use this system.

The *cadmin* object system should be considered distinct from the *cadmin* tools it supports. Your primary resource for using the *cadmin* system administration tools is the *Personal System Administration Guide*. This chapter describes only the administration that can be performed on the *cadmin* software itself, not the use of the *cadmin* software to administer your system. Information about the *cadmin* system is also available in the *release notes* that came with your system (or your most recent system software upgrade) and through the *desktop help* utilities on your system.

Topics covered in this chapter include:

- A system overview of *cadmin*. See “The cadmin Object System” on page 414.
- Information on manipulating the cadmin system. See “Starting the cadmin Daemons” on page 416, and “Stopping the cadmin Daemons” on page 417.
- Troubleshooting the objectserver daemon. See “Troubleshooting the Objectserver” on page 422.
- Troubleshooting the directoryserver daemon. See “Troubleshooting the Directoryserver” on page 423.

The *cadmin* Object System

The *cadmin* system has been designed to provide useful system administration tools to the majority of system administrators using IRIX. The *cadmin* object system is a collection of daemon programs, which run in the background and provide software services to the tool utilities that the user sees. The *cadmin* object system includes the following major parts:

- "The Objectserver"
- "The Directoryserver"
- "The File Manager"
- "The Desks Overview"
- "The Background Daemon"
- "The Media Daemon"
- "The Soundscheme Audio Server"

The Objectserver

The *objectserver* daemon handles requests for system resources such as disk drives, tape drives, and user accounts. The *objectserver* also modifies system files in response to adminstrator requests, such as for adding new users. See the *objectserver*(1M) reference page for complete information.

The Directoryserver

The *directoryserver* daemon maintains a database of all the managed objects (such as disks, tape drives, and CD drives) for all systems running an objectserver on the network. See the *directoryserver*(1M) reference page for complete information.

The File Manager

The graphical interface to the file system is an alternative to the IRIX shell for running applications and organizing information. It is similar to the

WorkSpace(1G) application of past IRIX releases. For complete information, see the *fm*(1G) reference page.

The Desks Overview

The desks overview provides controls for manipulating IndigoMagic environment "desks." The overview is completely described in the *ov*(1X) reference page and can create, change, copy, rename, and delete desks. Windows can also be dragged from one desk to another or placed on the global desk.

The Background Daemon

The Background Daemon (described in the *bgdaemon*(1X) reference page) manages the screen background. It switches backgrounds automatically when the user switches desks, and it communicates to the file manager when icons are on the background, so the file manager can maintain the background.

The Media Daemon

The Media Daemon (described in the *mediad*(1M) reference page) monitors the removable media devices on a system. When a CD or floppy disk is inserted, *mediad* mounts it as a file system, if possible. Some CDs (such as audio disks), and some floppies (for example, *tar*(1) floppies) are not mountable. When a user issues the eject command, eject sends *mediad* a message which causes it to attempt to unmount the media and eject it.

The Soundscheme Audio Server

The audio cue server daemon (described in the *soundscheme*(1) reference page) provides high-level audio playback services for user applications. Based on the audio and audiofile libraries, *soundscheme* mixes and plays sounds on demand as requested by multiple client programs using a single audio port.

Starting the cadmin Daemons

The following list describes how each daemon in the *cadmin* object system is started. By default, all these daemons are started at boot time. If you have the *cadmin* system installed on your machine, you should not need to start these daemons manually. This information is provided in the event that someone has turned these daemons off or the software is not working properly.

objectserver The bootup script */etc/init.d/cadmin* checks the value of the *objectserver* variable with the *chkconfig* command at boot time. Use the *chkconfig*(1M) command if you need to check that this daemon is running or make a change to its status at the next boot.

To change the status of this daemon while the system is running, use the command script:

```
/etc/init.d/cadmin [ start | stop ]
```

directoryserver The bootup script */etc/init.d/cadmin* also checks the value of the *directoryserver* variable with *chkconfig* at boot time. Use the *chkconfig*(1M) command if you need to check that this daemon is running or make a change to its status at the next boot.

To change the status of this daemon while the system is running, use the command script:

```
/etc/init.d/cadmin [ start | stop ]
```

fm The File Manager is started by default on most systems. The existence of a file named *.desktop* or *.nodesktop* in a user's home directory causes the daemon to abort. If the File Manager is not running, it can easily be invoked by choosing the *Desktop* and then the *Home Directory* items from the Toolchest on your screen.

ov The Desktop Overview is controlled by the user through the Toolchest. Select the *Desktop* item on your toolchest and then the *Desks Overview* item.

bgdaemon The */usr/lib/X11/xdm/Xsession.dt* file sets the background daemon to be started at boot time. This can be prevented only by commenting out the appropriate line in the *Xsession.dt* file.

mediad The bootup script */etc/init.d/mediad* checks the value of the *mediad* variable with *chkconfig*(1M) at boot time or whenever the *mediad* script is invoked as a direct command. Use the *chkconfig*(1M) command if you need to check that this daemon is running or make a change to its status at the next boot.

To change the status of this daemon while the system is running, use the command script:

```
/etc/init.d/mediad [ start | stop ]
```

soundscheme The */usr/lib/X11/xdm/Xsession.dt* file checks the value of the *soundscheme* variable with *chkconfig* at boot time. Use the *chkconfig*(1M) command if you need to check that this daemon is running or make a change to its status at the next boot.

Stopping the *cadmin* Daemons

From time to time, you may need to disable some parts of the *cadmin* object system. For example, to test new audio software, you may need to turn off the *soundscheme* daemon temporarily. The following sections describe how to disable the elements of the *cadmin* system safely, so that they can be easily restarted when necessary.

Stopping the Objectserver

To stop the *objectserver*, log in as *root* and issue the command:

```
/etc/init.d/cadmin stop
```

This stops the *objectserver* until you use the same script to restart the daemon with the *start* command.

If the *objectserver* daemons are not running, much of the administrative functionality is lost:

System Manager
: The System Manager tool will not start up. If you select the System Manager, you see a message from the *chost* tool saying "Cannot communicate with <*hostname*>. Perhaps there is no objectserver available on this system."

User Manager
: The User Manager also does not start up. When you first select the User Manager, it appears to be working correctly in that you see the message "Looking up user accounts. Please wait." However, after some time a message appears from the *cpeople* tool saying "The network did not respond correctly. Please try again. If it still does not respond, see the section on Troubleshooting Network Errors in the online Personal System Administration Guide."

Monitor Disk Space
: Monitor Disk Space does not work without the *objectserver*. If you attempt to select this service, the *cfile* tool gives this error message: "The network did not respond correctly. Please try again. If it still does not respond, see the section on Troubleshooting Network Errors in the Online *Personal System Administration Guide*."

These tools will work again if you restart the *objectserver* with the *chkconfig* command and a reboot or the command:

```
/etc/init.d/cadmin start
```

Stopping the Directory Server

The most convenient way to stop the *directoryserver* is to log in as *root* and issue the command:

```
/etc/init.d/cadmin stop
```

This directs the system not to run the *directoryserver* daemon. If you do not run this daemon, some of the system administration tools will fail because they will not be able to collect information from remote systems.

These tools will work again if you restart the *directoryserver* with the *chkconfig* command and a reboot or the command:

```
/etc/init.d/cadmin start
```

Stopping the File Manager

If you wish to stop the File Manager (*fm*) daemon from running, give the command:

```
touch $HOME/.desktop
```

When you next log in, the File Manager will not start up automatically. Note though, that the File Manager can be started up at any time by choosing the *Desktop* item and then the *Home Directory* item from the System Toolchest menu.

As the administrator, you can remove the *Home Directory* item from the System Toolchest by removing the relevant entry from the default */usr/lib/X11/system.chestrc* file. However, any user can make a custom *.chestrc* file in their home directory and reinclude the option.

When the File Manager is not running, no icons appear on the main background window. You cannot drag icons from another location, such as the Icon Catalog, onto the main background. You do not have iconic access to system or network peripherals. For many users, the biggest repercussion of not running the File Manager is that there is no graphical access to the directory structure, especially the user's home directory.

Stopping the Desks Overview

To stop the Desks Overview daemon, log in as *root* and issue the command:

```
killall ov
```

This kills the current instance of the Desks Overview. If the desktop updating mode (which is set in the *Desktop* Toolchest, *Customize* submenu, *Windows* item) is set to *explicit*, be sure to click on the *Set Home Session* button. This prevents the Desktop Overview from starting up when the user next logs in. Note that the Desks Overview can always be started by choosing *Desks Overview* from the *Desktop* Toolchest. The choice can be removed from the default */usr/lib/X11/system.chestrc* file if you choose, but any user can make a custom *.chestrc* file in their home directory and reinclude the option.

If the Desks Overview is not running, users can not switch between their desktops. The desktops are not removed, they are merely inaccessible.

Stopping the Background Daemon

The Background daemon is controlled by the */usr/lib/X11/xdm/Xsession.dt* file. If you move this file to another name, or comment out the individual line within the file that invokes the daemon, the daemon will not be started. For an example of moving the file, change directories to */usr/lib/X11/xdm* and use the command:

```
mv Xsession.dt Xsession.dt.orig
```

When you are ready to have the daemon started again, move the file back to its original name.

To restrict the daemon without losing all the instructions in the */usr/lib/X11/xdm/Xsession.dt* script file, you must edit the file. The relevant sections of the file look something like this:

```
# # BEGIN Desktop MODIFICATIONS # #
/usr/lib/desktop/bgdaemon &
```

To make any part of this file a comment, place a crosshatch (#) at the beginning of each line you wish to be a comment. A comment line is not interpreted as part of the command sequence in a script. It is assumed to be there to explain the command sequence to a reader. If you place a crosshatch in front of the line that invokes the Background daemon, the line looks like this:

```
# /usr/lib/desktop/bgdaemon &
```

When you next log in or reboot the system, the Background daemon will not be invoked. To regain the Background daemon, edit the file again and remove the comment mark. Then when you next log in or reboot the system, you will have your background daemon again.

Never delete lines from scripts such as these, as there is no convenient way to retrieve the line when it is needed. Use commenting to change the action of the script if you desire to restrict part of the script, or move the file to a different name if you wish to bypass the entire script.

When the Background daemon isn't running, several utilities are unavailable. Most noticeably, none of the specialized IndigoMagic backgrounds are available on the desktops. These backgrounds can still be

invoked with the *ipaste*(1) or *xsetroot*(1) commands. However, a background brought up this way applies to all desktops.

Since this daemon also controls management of icons on the main screen background, icon functionality is also lost when this daemon is inactive. For example, users cannot drag an icon from a directory view to the main background and users do not see an icon on the main background when a file is created the user's home directory.

Stopping the Media Daemon

To stop the media daemon (*mediad*), log in as *root* and issue the command:

```
/etc/init.d/mediad off
```

This stops the media daemon until you use the same script with the *start* command to restart it.

If *mediad* is not running, the user must mount all removable media themselves. This would have a major impact on users who are not familiar with the mount process for CD ROM disks and floppies. Another problem is that there is no indication on the icon for the media drive to indicate what type of media is in the peripheral device. For instance, if *mediad* is running and there is an audio CD in the CD-ROM drive (and *cdman* is not running), there will be musical notes above and to the left of the CD-ROM icon.

Stopping the Soundscheme Daemon

The most convenient way to stop the Soundscheme Audio Server is to log in as *root* and issue the command:

```
killall /usr/sbin/soundscheme
```

To eliminate the Soundscheme Server when you next reboot, use the command:

```
chkconfig soundscheme off
```

This directs the system not to run the *soundscheme* daemon. You can also stop the *soundscheme* daemon by editing the *Xsession.dt* file as described for the Background daemon.

If you disable the soundscheme daemon, the user will get no audio cues when events take place on their systems. For instance, saving a new file does not result in a "beep" and moving an icon on the background is not accompanied by a "cymbal swish" sound. When running the full IndigoMagic environment, this is equivalent to deselecting "Audio Feedback" from the Toolchest.

Troubleshooting the cadmin Object System

Troubleshooting the Objectserver

The *objectserver* may require occasional troubleshooting, especially if new scripts and tools have added to the standard configuration. Suppose you see the following error message:

```
Can't contact objectserver
```

There are several steps you can take to restore the system to correct operation. Follow these steps in order:

Log into the system as **root**.

1. Make sure that two *objectserver* daemons are running. In a shell window, enter the command:

   ```
   ps -ef | grep objectserver
   ```

   You should see three lines of information. At the far right of these columns, you should see the following words:

   ```
   /usr/Cadmin/bin/objectserver
   ```

   ```
   /usr/Cadmin/bin/objectserver
   ```

   ```
   grep objectserver
   ```

   The last item is the *grep* command you just entered, and there should be two separate instances of the *objectserver*.

2. If you see no occurrences or only one occurrence of the *objectserver*, check to see if the *objectserver* configuration flag is on. In a shell window, enter the command:

   ```
   chkconfig | grep objectserver
   ```

You should see the line:

```
objectserver on
```

Suppose the response that you see is:

```
objectserver off
```

Then you must turn on the *objectserver* by entering the command:

```
chkconfig objectserver on
```

Then cycle the *cadmin* init script by issuing the commands:

```
/etc/init.d/cadmin stop
```

```
/etc/init.d/cadmin start
```

3. Wait approximately 60 seconds and then reissue the command:

   ```
   ps -ef | grep objectserver
   ```

   Check the output to see that both *objectserver* daemons are running. If both required instances of the *objectserver* are running, the error message should no longer appear. If you see the message again, go on to the next step.

4. If two instances of the *objectserver* are running, but you still see the message:

   ```
   Can't contact objectserver
   ```

   The *objectserver* database may be corrupted. You must stop the *objectserver* daemons, repair the database and then start the *objectserver* daemons again with the following commands:

   ```
   /etc/init.d/cadmin stop
   ```

   ```
   /etc/init.d/cadmin clean
   ```

   ```
   /etc/init.d/cadmin start
   ```

Troubleshooting the Directoryserver

If you are attempting to place an icon on your desktop from a remote system on your network and you receive an error message from the *directoryserver*, check the system with the following steps.

1. Log into the system as **root**.
2. Make sure that the *directoryserver* is running. In a shell window, enter the command:

   `ps -ef | grep directoryserver`

   You should see two lines of information. At the far right of these columns, you should see the following words:

   `/usr/Cadmin/bin/directoryserver`

   `grep directoryserver`

   The second item is the *grep* command you just entered, and the other is the *directoryserver*.
3. If you see no indication of the directoryserver, check to see if the *directoryserver* configuration flag is on. In a shell window, enter the command:

   `chkconfig | grep directoryserver`

   You should see the following line:

   `directoryserver on`

   Suppose the response that you see is:

   `directoryserver off`

   Then you must turn on the objectserver by entering the command:

   `chkconfig directoryserver on`

   Then cycle the *cadmin* init script by issuing the commands:

   `/etc/init.d/cadmin stop`

   `/etc/init.d/cadmin start`
4. Now wait approximately 60 seconds and then reissue the command:

   `ps -ef | grep directoryserver`

   Check the output to see that the directoryserver is running. If it is, the error message should no longer appear. If you see the message again, go on to the next step.
5. If the *directoryserver* is running but you still see the error message, then check the *directoryserver* on the remote system with the resource you are trying to use.

Chapter 12

System Security

Chapter 12 describes the system software security functions of IRIX. Security is a topic of concern to almost all administrators, and IRIX provides a full set of UNIX system security features. Specific tasks covered in this chapter include:

- *How to set a PROM password.*
- *How to set up a dialup password.*
- *How to maintain user passwords.*
- *Steps to lock an account and completely deny access to the system.*
- *Network security procedures.*
- *Guidelines for the use of set-user-ID and set-group-ID programs. .*

Chapter 12

System Security

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

- General security guidelines. See “Security Guidelines” on page 428.
- How to set a PROM password to restrict access to the Command Monitor. See “PROM Passwords” on page 435.
- How to set up a dialup password to protect the system from unauthorized intrusion through your modem. See “Second (Dialup) Passwords” on page 437.
- How to use passwords to protect user and administrative accounts from unauthorized use. See “Protecting the System with Accounts and Passwords” on page 445.
- How to manage passwords to keep them safe. See “Logins and Passwords” on page 431.
- Steps to lock an account and completely deny access. See “Locking Unused Logins” on page 443.
- Security issues regarding transparent network access. See “Transparent Network Access” on page 448.
- Guidelines to prevent the unauthorized use of programs that allow one user to have the effective permissions as another user (illegitimate use of set-user-ID and set-group-ID programs). See “Set-UID and Set-GID Permissions” on page 450.
- A list of files and directories that are universally available for read and write access as shipped with the default system. You may wish to restrict permissions on these files. See “Universally Accessible Files and Directories” on page 453.
- A list of accounts shipped with IRIX that have no password set. See “Accounts Shipped Without Passwords” on page 454.

A great strength of the IRIX system is the ease with which users can share files and data. However, some of the security guidelines presented in this chapter are at odds with easy system access. It is up to the site administrator to balance the needs and concerns of the user community.

How Secure Is IRIX?

IRIX was never designed to be a truly secure system, but it has several features that allow you to achieve a generally acceptable level of security without adding any new software. Standard security features of IRIX are:

- classes of users—called *groups*—who can access information that belongs to other members of the same group; access to information can be granted or denied to members of the group by the user who owns the information
- permissions split into three categories: read, write, and execute
- the ability to encrypt data, using the *crypt*(1) command
- individual user accounts, protected by individual, encrypted passwords
- tools for monitoring login attempts
- tools for monitoring system activity, including:
  - finding out which processes are running, using the *ps*(1) command
  - determining who is logged on the system, using the *who*(1) command
  - maintaining logs of system activity, using process accounting commands

Security Guidelines

Computer security is the responsibility of not only the site administrator, but of everyone who has access to a computer at the site.

System users should safeguard their data by using appropriate file and directory permissions and guarding their account passwords.

Site administrators, and to some extent system users, should be aware of the following:

- Anyone with physical access to a computer can simply take it or take its disk drives(s).
- The same caveat applies to backups of the system: keep backups in a secure place. Anyone with physical access to backup tapes can gain access to any information stored on them.
- Permissions for directories and files should be set to allow only the necessary access for owner, group, and others. This minimizes the damage that one compromised account can cause.
- All active accounts need passwords, which should be changed regularly. Do not use obvious passwords, and do not store them online in "plain-text" format. If you must write them down on paper, store them in a safe place.

  Some sites with many workstations keep a log of each workstation's root password in a locked cabinet and limit access to the cabinet.

  For information about choosing passwords, see "Choosing Passwords" on page 446.
- Common-use accounts are a potential security hole. An example of a common-use account is one that is shared by all members of a department or work group. Another example is a standard "guest" account on all the workstations at a site. This allows all users at the site access to different workstations without requiring specific accounts on each workstation.

  A pitfall of common use accounts is that you cannot tell exactly who is responsible for the actions of the account on any given workstation. Another risk is that anyone trying to break into workstations at your site will try obvious account names such as *guest*.

  Common-use accounts can be helpful, but be aware that they can pose serious security problems. Needless to say, common-use accounts that do not have passwords are especially risky.
- Accounts that are no longer used should be either locked or backed up and removed, since unused accounts can be compromised as easily as current accounts.

Also, you should change critical passwords, including dial-up passwords, whenever anyone leaves the organization. Former employees should not have access to workstations at the site.

- Systems with dial-up ports should have special dial-up accounts and passwords. This is very important for sites that have common-use accounts, as discussed above.

  However, even with this added precaution, you should not store sensitive data on workstations that have dial-up access.

- If your site allows access to the Internet network (for example, using *ftp*(1C)), you should take precautions to isolate access to a specific gateway workstation.

  Access from this gateway should be controlled, for example by disallowing *ftp* connections between the gateway and other workstations within the company.

- Discourage use of the *su*(1) command unless absolutely necessary. The *su* command allows a user to change his or her user ID to that of another user. It is sometimes legitimately necessary to use *su* to access information owned by another user, but this presents an obvious temptation: the person using *su* to switch user IDs now knows another person's password and has full access to his or her account.

  The file */var/adm/sulog* contains a log of all successful and unsuccessful attempts to use the *su* command.

- Make sure that each user's home account, and especially the shell-startup files (*.profile*, or *.login* and *.cshrc*) are writable only by that user. This ensures that "trojan horse" programs are not inserted in a user's login files. (A trojan horse program is a file that appears to be a normal file, but in fact causes damage when invoked by a legitimate user.)

- Be sure that system directories such as */bin*, */usr/bin*, and */etc* and the files in them are not writable except by the owner. This also prevents trojan horse attacks.

- Sensitive data files should be encrypted. The *crypt*(1) command, together with the encryption capabilities of the editors (*ed* and *vi*), provides protection for sensitive information.

- If you must leave your console, workstation, or terminal unattended, log off the system. This is especially important if you are logged in as **root**.

- Use only that software that is provided by reputable manufacturers. Be wary of programs that are distributed "publicly," especially already-compiled binaries. Programs that are available on public bulletin board systems (as opposed to BBSs run and sponsored by vendors) and on public computer networks could contain malicious "virus" and "worm" routines that can violate system security and cause data loss.

  Public-domain source code is safer than already-compiled programs, but only if you examine the code thoroughly before compiling it. Be suspicious of programs that must be installed set-UID *root* in order to run.

- Safeguard and regularly check your network hardware. One possible way to break into computer systems is to eavesdrop on network traffic using physical taps on the network cable. Taps can be physical connections (such as a vampire tap) or inductive taps.

  Run networking cable through secure areas and make sure it is easy to examine regularly. Create and maintain a hardcopy map of the network to make it easier to spot unauthorized taps. Another way to make this sort of attack more difficult is to use fiber-optic (FDDI) network hardware, which will not operate correctly if there is any break in the cable.

Logins and Passwords

System security under IRIX is primarily dependent on system login accounts and passwords. Proper administration, user education, and use of the facilities provided will yield adequate security for most sites. Most security breaches are the result of human error and improper use of the provided security features. No extra measures will yield more security if the basic features are not being used. This discussion of logins and passwords covers the following:

- System login options, such as the maximum allowable number of unsuccessful login attempts, whether to record successful logins, and whether to force users who do not have passwords to choose them immediately. See "System Login Options" on page 432.
- Setting the PROM password on your machine. See "PROM Passwords" on page 435.

- How to assign a system password (called a *dialup password*). See "Second (Dialup) Passwords" on page 437 and "Accounts and Passwords" on page 81.
- How to move your password file to a "shadow" password file to increase security. See "Creating a Shadow Password File" on page 439.
- How to set up password aging. See "Password Aging" on page 440.
- How to lock unused login accounts. See "Locking Unused Logins" on page 443.
- Information about administrative and special accounts. See "Special Accounts" on page 444.
- How to protect the system with accounts and passwords. See "Protecting the System with Accounts and Passwords" on page 445.
- How to choose good passwords. See "Choosing Passwords" on page 446.
- How to run *pwck*(1M) to find discrepancies in the password file. See "Using pwck(1M) to Check the Password File" on page 446.

Managing passwords is also described in Chapter 3, "User Services."

System Login Options

For security, you can set the following login options:

- the number of times an attempt to log in can fail before a delay is introduced in the login process
- if the login sequence is delayed, how long before the login process is resumed
- whether to maintain a log of logins and what information to store: all logins or only those that were unsuccessful
- whether to force a user who does not have a password to choose one immediately upon logging in
- whether or not to display, when a user logs in, the date and time that user last logged in

Login options are set in the file */etc/config/login.options*, which is a normal text file. The file contains one option specification per line. The options are described in the rest of this section.

These options are important because the login procedure is your system's main defense against unauthorized access. For example, you can determine whether someone is trying to break into your system from a pattern of failed login attempts recorded in */var/adm/SYSLOG* (when logging is enabled).

Note that the best way to keep a system secure is to slow down attempts to guess passwords and account names. The login options described in this section add delays to unsuccessful login attempts, which drastically slows down the process of randomly guessing passwords.

See also the *login*(1) and *login*(4) reference pages.

Maximum Login Attempts (*maxtries*)

Setting this parameter slows attempts by unauthorized persons to break into a system. A common method of breaking into a system is to try to guess the password of a known account. This method is most successful if the person trying to break in knows the names of as many accounts as possible, and can make guesses very quickly. If you introduce a delay in the login process after a certain number of failed login attempts on the same *tty* line, you can make it much more time-consuming to guess a correct password.

To set the maximum number of login attempts, edit the file */etc/config/login.options*. Place a line similar to this in the file:

```
maxtries=3
```

This sets the maximum number of login attempts to three. The system default, without this option set, is five.

When the maximum number of login attempts is exceeded, the *login* program sleeps for a certain number of seconds, thus preventing further login attempts on that line for a while. The system default is twenty seconds.

Length of Time to Disable a Line (*disabletime*)

Use this option along with the **maxtries** option. To set the number of seconds after a certain number of unsuccessful login attempts that a line is disabled, edit the file */etc/config/login.options* and add a line similar to this:

```
disabletime=30
```

This disables a line for thirty seconds. You can choose any value you consider appropriate for your system. The system default is twenty seconds.

Recording Login Attempts

You can log both successful and unsuccessful login attempts in the file */var/adm/SYSLOG*. To log all attempts to log in, place this line in the file */etc/config/login.options*:

```
syslog=all
```

To log only unsuccessful attempts, place this line in the *login.options* file:

```
syslog=fail
```

A large number of failed logins, especially with the same account name, may indicate that someone is trying to break into the system.

Note that the visual login process *pandora*(1) does not provide these security options. To use the login security functions, you must turn off *pandora* and use the standard login processes, *getty*(1) and *login*(1). Use *chkconfig* to turn off the *visuallogin* and *xdm* configuration variables. See "Altering the System Configuration" on page 39 and the *visuallogin*(4) reference page for information about turning the visual login process on and off. You may also use *chkconfig* to set the *noiconlogin* variable to disallow logging in using the user icons in *pandora*.

Forcing a Password

To force users who do not have passwords for their accounts to choose their passwords immediately, add this line to the file */etc/config/login.options*:

```
passwdreq
```

Displaying the Last Login Time

Users can help maintain system security by noticing unauthorized use of their accounts. To help them, you can configure *login* to display the date and time when they last logged in successfully.

To do this, edit the file */etc/config/login.options*, and add this line:

```
lastlog
```

This displays the most recent login date, time, and the name of the terminal line (*tty* name) or remote host from which the user logged in. This login attempt information is recorded in files, one per user account and with the same name as the account, in the directory */var/adm/lastlog*.

PROM Passwords

Your system may have a facility that allows you to require a password from users who attempt to gain access to the Command (PROM) Monitor. This allows you to exercise greater control over who may perform system administration tasks and affords you some protection against severe ‘‘denial of service’’ attacks from malicious intruders. These attacks are designed not to steal your information, but to make your system useless to you.

Traditionally, if an intruder gains access to your system hardware, there is little you can do to maintain system security. In the simplest case, the intruder switches off the system, then turns it on again, and instructs the system from the console to boot a program other than your usual operating system. Alternately, the intruder could simply remove the hard disk from your machine and install it on another machine and read your files. While there is nothing you can do with system software to prevent physical theft of the hardware, you can limit the ability of intruders to boot their programs or to otherwise damage your system at its lowest levels with a PROM password.

To determine if your system supports PROM passwords, use the *nvram*(1M) command. Issue the command with no arguments and observe the output. The output shows the current settings of all non-volatile memory variables. If the variable *passwd_key* is present, your system supports PROM passwording. Alternately, you can select option 5 from the System

Maintenance Menu to enter the Command Monitor. You see the Command Monitor prompt:

```
Command Monitor. Type "exit" to return to the menu.
>>
```

Enter the command:

```
help
```

The system prints a list of available commands for the Command Monitor. If the *passwd* command is among those listed, your system supports the PROM password. If it is not listed, your system hardware does not support passwording. If you would like to upgrade your system to support passwording, please contact your sales representative.

Setting the PROM Password Using nvram(1M)

To set the PROM password using the *nvram*(1M) command, perform the following steps:

1. Log in as **root**.
2. Give the following command:

   nvram passwd_key "*new_password***"**

   In this example, only the term *new_password* should be replaced; the quotation marks must be used to contain the password text. Your PROM password is now set.

Note that if you forget your PROM password, but you still know your *root* password, you can reset the PROM password on some machines through the *nvram* command. If you cannot successfully reset the PROM password, you must remove the PROM or a jumper from your CPU board. See your *Owner's Guide* for information on this procedure.

Setting the PROM Password From the Command Monitor

If you wish to set your PROM password from within the Command Monitor, perform the following steps:

1. Log in as **root** and shut the system down.
2. When you see the message:

```
Starting up the system...
To perform system maintenance instead, press <Esc>
```

press the **<Esc>** key to see the System Maintenance Menu.

3. Select option 5 from the System Maintenance Menu to enter the Command Monitor. You see the Command Monitor prompt:

   ```
   >>
   ```

4. Issue the *passwd* command and press **<Return>**:

   ```
   passwd
   ```

   You see the prompt:

   ```
   Enter new password:
   ```

5. Enter the password you want for your machine and press **<Return>**. You see the following prompt:

   ```
   Confirm new password:
   ```

6. Enter the password again, exactly as you typed it before. If you typed the password the same as the first time, you will see the Command Monitor prompt again.Your password is now set. Whenever you access the Command Monitor, you will be required to enter this password.

Second (Dialup) Passwords

If your system requires additional protection, you can establish a *system password*. If you do this, users who log in on specific ports (ttys) are prompted for a system password in addition to their account passwords. This feature cannot be imposed on the system console, or any terminal where *pandora* or *xdm* is used.

System passwords are normally used only on dialup lines and are often referred to as *dialup passwords*. You can use them on standard lines, but this is usually not necessary.

To establish a system password, follow these steps:

1. Log in as **root**.
2. Edit the file */etc/dialups*.

   Place in the file a list of ports (ttys) that require the second password. For example:

   ```
   /dev/ttyd1
   /dev/ttyd2
   /dev/ttyd3
   ```

   Write the file and exit the editor.
3. Decide on the desired password or passwords. System passwords are assigned on a shell-by-shell basis. You can assign the same password for all the possible shells on the system, assign different passwords for each shell, or use some combination of approaches.
4. Encrypt the desired password. You must use the *passwd* program to perform the encryption. You cannot use the *crypt*(1) command for this purpose.

   To encrypt the password, simply change the password of some account (for example the *bin* account) to the password you wish to use in */etc/d_passwd*. Before you do this, note what the existing password is (or if the account is locked). You should return the account to this state when you are finished assigning a system password.

   For example, to change the password of the *bin* account to ''2themoon'' you enter:

   ```
   passwd bin
   ```

   You see:

   ```
   New password:
   ```

   Now enter the string ''**2themoon**'' and then press **<Return>**.The string ''2themoon'' is not displayed as you type it. Next you see:
   ```
   Re-enter password:
   ```

   Enter the string ''**2themoon**'' again and then press **<Return>**. The string is still not displayed as you type it.

Examine the entry for the *bin* account in the file */etc/passwd*. You should see something like this:

```
bin:SaXub4uaL5NP:2:2:System Tools Owner:/bin
```

The second field (between the first and second colons) is the encrypted version of the password "2themoon."

5. Edit the file */etc/d_passwd*. In the file, place lines of this form:

   shell:password:

   shell is the command interpreter (shell) you wish to have a password, and *password* is the encrypted password.

   For example, this assigns the password "2themoon," which you encrypted in the previous step, to all C shell users who log in on the ttys specified in */etc/dialups*:

   ```
   /bin/csh:SaXub4uaL5NP2:
   ```

   You must place a colon at the end of the encrypted password, and you must enter the shell program pathname exactly as it appears in */etc/passwd*.

   Write the file and exit from the editor.

6. Make sure the files have appropriate permissions by issuing the command:

   ```
   chmod 640 /etc/d_passwd /etc/dialups
   ```

7. Remove the password you assigned to the system account in step 4. To do this, edit the file */etc/passwd* and remove the string of characters in the second field. Return this field to the same state as when you began this procedure.

Now, whenever C shell users log in on the ttys specified in */etc/dialups*, they are prompted for the system password "2themoon" in addition to their account password.

Creating a Shadow Password File

A "shadow" password file is simply a copy of the standard password file, but it is not accessible by non-privileged users. In the standard configuration, the */etc/passwd* file is publicly readable. Since the */etc/passwd*

file contains the encrypted versions of the users' passwords, anyone can make a copy and attempt decryption of the passwords for malicious purposes. By invoking a shadow password file, you prevent intruders from attempting to decrypt your passwords.

The shadow password file is called */etc/shadow*. Once shadow passwords have been initialized, the password field in each */etc/passwd* entry is replaced by an "x" character.

To initialize */etc/shadow* (and thus invoke shadow passwords), run the *pwconv*(1M) command. Once this command has been run, shadow passwords will be in effect. All standard password tools work transparently with shadow passwords. The difference should not be noticeable to your users, except that they will not be able to see the encrypted passwords in the */etc/passwd* file.

One difference in system operation is that older applications will not get the proper value of *pw_passwd* from the *getpwent*(3C) and *getpwnam*(3C) library calls. This primarily affects "screen saver" programs.

Password Aging

The password aging mechanism forces users to change their passwords periodically. It also prevents a user from changing a new password before a specified time interval. You can also force a user to change his or her password immediately.

Realistically, password aging forces users to adopt at least two passwords for their accounts. This is because, when password aging is enforced, most users alternate between two passwords that they find easy to remember rather than inventing brand new passwords every time their old ones expire. IRIX does not provide a utility that determines whether users are choosing between a set of passwords and that forces them to choose completely different passwords.

Password Aging with the *passwd* Command

To set the maximum number of days that can elapse before a user must change his or her password, use the *passwd*(1) command.

For example, this command forces user *alice* to change her password every two weeks (14 days):

```
passwd -x 14 alice
```

If you set **-x** to 0, the user must change her password when it next expires, but thereafter password aging is not in effect for her. If you set **-x** to -1, password aging is turned off immediately for that user.

You can also set the minimum time that must elapse before users are allowed to change their passwords. This is useful to prevent users from changing their passwords, then changing them back to their old passwords immediately. For example:

```
passwd -x 14 -n 7 ralph
```

This forces user *ralph* to change his password every fourteen days and prevents him from changing it more frequently than once every seven days. Note that if you set the minimum value greater than the maximum value, the user may not ever change his or her password.

To force users to change their passwords immediately, use the **-f** option. For example:

```
passwd -f trixie
```

Using Password Aging Manually

Another way to enforce password aging is to edit the */etc/passwd* file and insert the appropriate information after the password fields in the desired account entries.

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
```

The meaning of these fields is as follows:

,	The comma separates the password and the aging information.
M	The **M**aximum duration of the password.

m	The **m**inimum time interval before the existing password can be changed by the user.
ww	The week (counted from the beginning of 1970) when the password was last changed and 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. The following chart shows the relationship between the numerical values and character codes. Any of the character codes can be used in the four fields of the password aging information. Table 12-1 lists the password aging codes and their meanings.

Table 12-1 Password Aging Character Codes

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

Two special cases apply for the character codes:

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

The following shows the password aging information required to establish a new password every two weeks (0) and to deny changing the new password for one week (/) for user *ralph*:

```
ralph:RSOE2m.E,0/:100:1:Ralph P.Cramden:/usr/people/ralph:
```

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

```
ralph:RSOE2m.E,0/W9:100:1:Ralph P. Cramden:/usr/people/ralph:
```

In this example, *ralph* changed his password in week W9. To force *ralph* to change his password at the next login (and to cause this only once), you can add the code **,..** to the password field:

```
ralph:RSOE2m.E,..:100:1:Ralph P.Cramden:/usr/people/ralph:
```

After *ralph* changes his password, the system automatically removes the aging code (**,..**) from the password field. To prevent *ralph* from changing his password, use the code **,./**. Edit the */etc/passwd* file and add a comma, period, and slash to the password field:

```
ralph:RSOE2m.E,./:100:1:Ralph P. Cramden:/usr/people/ralph:
```

Now only *root* can change the password for the *ralph* account. If *ralph* tries to change the password, he sees the message "`permission denied`."

Locking Unused Logins

If a login is not used or needed, you should disable (lock) the login. You should not remove the account, though, because of the danger of re-using the UID in the future. User ID numbers are meant to be permanently associated with the person who used the account. If you re-use the UID number, the new user may find files that belonged to the previous owner of the ID number. These files may contain "trojan horse" programs that could do damage to your system. If you are extremely short on disk space, you may remove the user's home directory and files (after making a backup), but you should never remove an entry from your */etc/passwd* file.

There are two ways to lock an account. The first is using the *passwd* command with the **-l** option. For example:

```
passwd -l norton
```

This command changes the password field of the entry in */etc/passwd* for account *norton* to `*LK*`.

The second way to lock an account is by editing the password file directly. Change the password field to any string of characters that is not used by the password encryption program to create encrypted passwords. The *passwd* command with the **-l** option uses the string `*LK*`.

You can use other strings to lock accounts. For example, you can use a descriptive phrase such as "LOCKED;" to remind you that the account was deliberately disabled:

```
ralph:LOCKED;:100:1:Ralph P. Cramden:/usr/people/ralph:
```

The semicolon is not used in an encrypted password and causes the account to be locked. The text "LOCKED" is merely to remind you that the account is locked.

Another common method of disabling a password is to put an asterisk (*) in the password field. The default IRIX */etc/passwd* file disables unused logins in this manner.

Special Accounts

Special accounts are used by daemons to perform system functions, such as spooling UUCP jobs and print requests. Because key files are owned by these accounts, someone who obtained access to one of the accounts, or was able to start a daemon on your system, could not completely breech security. Ownership of the various system files is distributed among the special accounts.

Guard access to all the special accounts as you would the *root* account. We recommend you either assign passwords to these accounts, or lock them using one of the methods described in previous sections.

The following is a list of all the administrative and special accounts on the system and what they are used for.

root — This login has no restrictions, and it overrides all other logins, protections, and permissions. It allows you 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*. Its login should be disabled.
bin	This login has the power of a normal user login over the files it owns, which are throughout the system. Its login should be disabled.
adm	This login has the power of a normal user login over the object files it owns, which are located in */var/adm*. You may *su* to the adm login. This login should be disabled.
uucp	This login owns the object and spooled data files in */usr/lib/uucp* and */etc/uucp*.
nuucp	This login is used by remote workstations to log into the system and initiate file transfers through */usr/lib/uucp/uucico*.
daemon	This login is the system daemon, which controls background processing. Its login should be disabled.
lp	This login owns the object and spooled data files in */var/spool/lp*. Its login should be disabled.

Protecting the System with Accounts and Passwords

There are several ways accounts and passwords protect the system:

- By requiring users to log in with specific accounts, you can determine who is responsible for specific actions on the system.
- Using the IRIX system of file permissions, users can keep data reasonably secure. Other users on the system are less likely to accidentally view confidential material.
- If all accounts have passwords, the chance of an unauthorized person accessing the system is greatly reduced. However, the possibility of unauthorized access increases if users are lax about changing their passwords regularly and choosing good passwords. The next section describes how to choose good passwords.

Choosing Passwords

A system is most secure if nobody can access the system without an account and password, and if all the passwords on the system are difficult to guess and obtain. Surprisingly, many users choose passwords that are easy for potential intruders to guess, or write their passwords down on paper and leave them near their workstations and terminals.

Many site administrators use the same password for multiple administrative accounts. This is not a good practice. Do not deliberately use the same password for more than one account.

Good passwords are:

- long, up to eight characters
- multiple words that are combined or arranged in an unusual manner
- words from multiple languages, combined in a unique way
- composed of different kinds of characters, such as digits and punctuation

Bad passwords are:

- short
- single words that are in a dictionary
- the same as the account name, or the account name spelled backward
- the name of the user's department or project
- the user's name or initials
- the license number of the user's car, a spouse or friend's name, the user's home address, phone number, age, or some other obvious information

Using pwck(1M) to Check the Password File

From time to time, you should run the *pwck*(1M) utility to scan the password file. This program reads the file and checks each entry for completeness and

notes any inconsistencies. The password checks include validation of the following items:

- the number of fields in each entry
- the login name
- the user ID number
- the group ID number
- the login directory
- the executed program

The default password file to be checked is */etc/passwd*. If shadow passwords (described in "Creating a Shadow Password File" on page 439) are enabled, the */etc/shadow* file is checked.

Similarly, the *grpck*(1M) command verifies all entries in the */etc/group* file. The default group file to be checked is */etc/group*. With either command, an alternate file may be specified on the command line.

Network Security

Unless drastic steps are taken to ensure security, networks are generally quite open and thus, insecure. This section touches on some aspects of network security in addition to keeping the physical network cable safe from tampering and eavesdropping. For more information on network security, see "Planning for Network Security" on page 541.

Controlling Network Access

There are three files that help you control access to a host across the network:

/etc/hosts.equiv
: A list of hosts that are considered trusted, or *equivalent* to you.

rhosts
: A list of hosts that are allowed access to a specific user account.

/etc/passwd
: The list of system accounts.

These three files control whether access is granted or denied when a remote host issues an *rlogin*(1C), *rcp*(1C), *rdist*(1C), or *rsh*(1C) request.

When a request for access is received, the file *hosts.equiv* is checked, and if the host is listed in that file, and the target user account is listed in */etc/passwd*, no further checking is performed and remote access is allowed. In this case, a remote user with a local user ID has equivalent access from a remote host.

Users can expand this equivalence by listing hosts and specific accounts in *.rhosts* files in their home directories. The *root* login bypasses the */etc/hosts.equiv* file and typically uses only the */.rhosts* file in the root directory for equivalence checking. If there is an entry in the */.rhosts* file for *root*, the *root* user on the remote system will have *root* privilege on your system. For obvious reasons, this is not a secure practice. It is much more secure to handle file transfers through the non-privileged *guest* account.

The owner of the *.rhosts* file must be either the user in whose home directory it resides, or the superuser, *root*. If it is owned by another user, or if the file permissions allow anyone who is not the owner of the file to modify it, the contents of a user's *.rhosts* file are disregarded as a security practice.

For complete information about the */etc/hosts.equiv* and *.rhosts* files, see the *hosts.equiv*(4) reference page.

Transparent Network Access

With the X Window System™, workstations can run client programs transparently on other hosts on the network. This access is completely independent of such controls as login accounts and passwords and is done through X protocols.

By default, IRIX workstations are configured to allow complete, transparent access for all workstations on the network that use the X Window System. You can change this using the *xhost*(1) server access control program and the configuration file */etc/X*.hosts*. In the configuration file name, the asterisk (*) corresponds to the number of the server on the local host. This is usually 0, so for most workstations the file is */etc/X0.hosts*.

When the X server starts, it checks the file */etc/X*.hosts*. For example, server 0 checks for */etc/X0.hosts*, server 1 checks for */etc/X1.hosts*, and so forth. If the

file is missing, or is empty, no remote hosts are allowed access to the server. If the file contains a single plus sign (+), all remote hosts are allowed access.

Next, the *xhost* command is run from the file */usr/lib/X11/xdm/Xsession*. In the default *Xsession* file, *xhost* is used to allow access to all remote hosts. To change the default server-access permissions, you must change how the *xhost* command is run from the *Xsession* file. Then, you can customize the */X*.host* file.

The *xhost* program modifies the internal state of the X server. Using *xhost*, you can allow or deny server access for specific hosts, or for all hosts. Note that the *xhost* options that affect access control can be run only from the same workstation as the server.

To completely deny access to all hosts on your network through X protocols, use this command:

```
xhost -
```

To allow complete access to all hosts on your network, use this command:

```
xhost +
```

To selectively grant or deny access, specify the name of the specific host or hosts on the command line. For example, this command grants access to a host named *brooklyn*:

```
xhost +brooklyn
```

When granting access, the plus sign (+) is optional.

This command denies access to both *brooklyn* and *bronx:*

```
xhost -brooklyn -bronx
```

To see which hosts are currently allowed access to the server, run *xhost* from the command line with no options:

```
xhost
```

You can selectively allow access to remote hosts by listing their names in the */etc/X*.host* file. For example, if the file */etc/X0.hosts* contains the following

line, the remote hosts *bronx* and *brooklyn* are the only workstations allowed to access the local server for server 0:

```
bronx
```

In the above example, all other hosts are denied access to the local server.

The *xhost* command overrides the configuration file *X*.host*. To alter the default system configuration, you must not only modify the configuration file, but also change the *xhost* command in the */usr/lib/X11/xdm/Xsession* file.

Note: Do not link the file X*.hosts to any other network host database, such as */etc/hosts* or */etc/hosts.equiv*. When the X server starts, it attempts to establish a connection to all hosts that are allowed access permission in the *X*.hosts* file. If this file contains a large number of hosts that are allowed access to the server, you will have to wait until connections are established with each of the hosts before the server is started.

For more information about X security and authorization, see the *xhost*(1), *xauth*(1), *xserver*(1), and *X*(1) reference pages.

Set-UID and Set-GID Permissions

The set user identification (set-UID) and set group identification (set-GID) permission bits must be used very carefully. When a user runs an executable file that has either of these permission bits set, the system gives the user the permissions of the owner of the executable file. You can add these permissions to any executable file with the *chmod*(1) command.

Set-UID and set-GID programs have legitimate uses, but because they are potentially harmful, there should be very few of them on your system. Beware of programs in publicly writable directories (such as */tmp*, */usr/tmp.O*, */var/tmp*, and */usr/spool/uucppublic*) that have the same name as common systems files (such as *vi* and *rm*). One reason the PATH environment variable of the *root* account does not include the current directory (as does the default PATH of most other users) is so that *root* won't accidentally execute such ''booby-trap'' programs.

System security can be compromised if a user copies another program onto a file with `-rwsrwxrwx` permissions. To take an extreme example, if the *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 sections provide some example commands that identify files on the system with set-UID permissions. For more information about the set-UID and set-GID bits, see the *chmod*(1) and *chmod*(2) reference pages.

Checking for Set-UIDs Owned by *root*

The following command line lists all set-UID files owned specifically by *root*:

```
find / -user root -perm -4000 -print
```

The results of this command are printed on the screen. All paths are checked starting at **/**, including all mounted directories. A great number of files will be found. It is up to you to scan these files for any unusual names. One possibility is to direct the output of this program to a file soon after installation and compare the results with later outputs. If this command reports any unusual files, you should investigate them immediately.

A suspicious file might turn up like this:

```
-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-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  jbond 45376 Aug 18 15:11 /usr/jbond/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
```

In this example, the user *jbond* has a personal copy of */bin/sh* and has made it set-UID to *root*. This means that *jbond* can execute */usr/jbond/bin/sh* and become the superuser.

Checking for Set-UIDs in the Root File System

The following command line reports all files with a set-UID for the root file system (not just those owned by *root*):

```
ls -l '/etc/ncheck -s /dev/root | cut -f2 | grep -v dev'
```

The *ncheck*(1M) command, by itself, can be used on a mounted or unmounted file system. Only the superuser may use *ncheck*. The normal output of the *ncheck -s* command includes special files.

Here, the *grep* command removes device files from the output. This filtering is applicable only for the root file system. The output of the modified *ncheck* is then used as an argument to the *ls* command. The file system must be mounted for the *ls* command to succeed. In this example output, nothing looks suspicious:

```
-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
```

Checking Set-UIDs in Other File Systems

This example uses the *ncheck* command to examine the *usr* file system (*/dev/usr*, assuming a single-disk system with default partitioning) for files that have set-UID permissions:

```
/etc/ncheck -s /dev/usr | cut -f2
```

In this partial example below, complete pathnames for the files start with */usr*. */usr* is not part of the *ncheck* output.

In this sample output, the program */usr/jbond/bin/sh* should be investigated. This program is the only one in the listing of set-UID programs that is not

found in a system directory. It is a command shell residing in a user's home directory. Users should, in general, not possess of set-UID binaries.

```
/dev/usr:
/bin/at                 /bin/uux
/bin/crontab            /lib/mv_dir
/bin/shl                /lib/expreserve
/bin/sadp               /lib/exrecover
/bin/timex              /lib/accept
/bin/cancel             /lib/lpadmin
/bin/disable            /lib/lpmove
/bin/enable             /lib/lpsched
/lib/reject             /lib/lpshut
/lib/sa/sadc            /bin/lp
/lib/uucp/uucico        /bin/lpstat
/lib/uucp/uusched       /bin/ct
/bin/uucp               /bin/cu
/bin/uuname             /lib/uucp/uuxqt
/bin/uustat             /jbond/bin/sh
```

Universally Accessible Files and Directories

The following files and directories are universally available for read and write access. Depending on your site requirements, you may wish to change the permissions on these files to be more restrictive. Remember, though, that restricting permissions on historically open directories, such as */tmp*, */usr/tmp.O*, and */var/tmp* (linked to */usr/tmp*), can cause serious malfunctions in many programs, applications, and system utilities that write temporary files on behalf of users in these directories.

- */tmp*
- */usr/demos/.xsession*
- */usr/Insight/tmp*
- */usr/Insight/tmp/ebtpriv*
- */usr/Insight/tmp/ebtpub*
- */usr/Insight/tmp/install.insight.log*
- */usr/lib/emacs/maclib*
- */usr/lib/showcase/fonts*

- */usr/lib/showcase/images*
- */usr/lib/showcase/models*
- */usr/lib/showcase/templates*
- */usr/tmp.O*
- */var/spool/locks*
- */var/spool/uucppublic*
- */var/tmp*

Accounts Shipped Without Passwords

The following accounts in your default */etc/passwd* file are shipped without passwords. You should create passwords for these accounts immediately.

- demos
- guest
- lp
- nuucp
- root
- tour
- tutor
- 4Dgifts

Chapter 13

Administering the System Audit Trail

Chapter 13 describes the system audit trail facilities available with IRIX. With these utilities, you can keep track of system usage on a per-system call basis. Topics covered here include:

- *Enabling and configuring auditing.*
- *How to audit specific users, processes, and files.*

Chapter 13

Administering the System Audit Trail

The System Audit Trail is a feature that allows the administrators to review a record of all system activity. The ongoing record of system activity shows general trends in system usage and also violations of your system use policy. For example, any unsuccessful attempts to use system resources can be recorded in the audit trail. If a user consistently attempts to access files owned by other users, or attempts to guess the **root** password, this can be recorded also. The site administrators can monitor all system activity through the audit trail. Sections of this chapter include:

- Starting the auditing process. See “Enabling Auditing” on page 458, and “Customizing Auditing” on page 459.
- Specifying events to audit. See “Customizing Auditing” on page 459, “How to Audit a Specific User” on page 471, “How to Audit a File” on page 472, and “How to Audit a Label Under Trusted IRIX/B” on page 473.
- Reading and interpreting the audit data. See “Understanding the Audit Data” on page 470.
- Archiving and removing audit data. See “Archiving Audit Data” on page 480, and “Removing Audit Data” on page 481.

Note that references are made in this chapter to auditable "MAC" and "Mandatory Access Control" events, such as an event generated when an attempt is made to access a file protected by a higher MAC clearance. The audit system provides facilities to audit all events on all IRIX operating systems. Mandatory Access Control (MAC) is available only in the Trusted IRIX/B optional operating system. No MAC audit events are generated by standard IRIX. If you have installed Trusted IRIX/B, you will have received additional documentation describing the special security features in that product. Users of standard IRIX can safely ignore all references to MAC, labels, and the *dbedit*(1), *chlabel*(1) and *newlabel*(1) commands. To find out if your system is running Trusted IRIX/B, use the *uname*(1) command with the **-a** option. Standard IRIX systems will give a response that looks like this:

```
IRIX  System_name  5.1  02131441  IP12
```

If your machine is running Trusted IRIX/B, the name IRIX in the above example will be replaced with "Trusted IRIX/B."

Discretionary Access Control (DAC) is the term used by the auditing subsystem for the standard UNIX system of file permissions. IRIX uses the standard permissions system common to all UNIX based operating systems.

Enabling Auditing

The audit subsystem is distributed with your IRIX operating system media, but is not installed by default. To enable auditing, you must use the *inst*(1M) utility to install the **eoe2.sw.audit** software package from your distribution media. Using Inst, the command line interface to *inst*(1M) is described in detail in the *Software Installation Administrator's Guide*. Once this package has been installed, reboot your system and use the *chkconfig*(1M) utility to enable auditing. The *chkconfig* reference page provides complete information on the use of *chkconfig* but, simply described, you will see a list of configurable options and a notation of "off" or "on" for each option. The list is in alphabetical order. For example, here is a partial *chkconfig* listing that includes the audit option:

```
Flag                State
====                =====
audit               off
automount           on
windowsystem        on
xdm                 off
```

Give the command:

```
chkconfig audit on
```

This command enables auditing on your system. The system will immediately being collecting audit data on the default set of audit events. The default audit events are listed and described below.

Default Auditing

The default auditing environment is already set up when you install IRIX. You need not take any action to maintain the default auditing environment. Within your default IRIX distribution, there is a file called */etc/init.d/audit*. This file contains the default audit trail initialization. The default auditing selections produce a full record of system activity with a minimum of disk space usage. The following is a list of all event types audited by default. (The individual event types are not described in this list, but a description for all event types is given below in "Auditable Events" on page 462.)

Table 13-1 Events Audited By Default

sat_access_denied	sat_chdir	sat_chroot
sat_open	sat_file_crt_del	sat_file_crt_del2
sat_file_write	sat_mount	sat_file_attr_write
sat_exec sat_sysacct	sat_fchdir	sat_tty_setlabel
sat_fd_attr_write	sat_proc_read	sat_proc_write
sat_proc_attr_write	sat_fork sat_exit	sat_proc_attr_write
sat_proc_attr_write2	sat_svipc_create	sat_svipc_remove
sat_svipc_change	sat_bsdipc_create	sat_bsdipc_create_pair
sat_bsdipc_shutdown	sat_bsdipc_mac_change	sat_bsdipc_expl_addr
sat_hostid_set	sat_clock_set	sat_hostname_set
sat_domainname_set	sat_ae_custom	sat_ae_identity
sat_ae_dbedit	sat_ae_mount	

Customizing Auditing

When you have installed your system, you can select the level and type of auditing that you wish to use. The default auditing environment described above is created for you at installation time. For most purposes this auditing environment is satisfactory. However, remember that the System Audit Trail is completely configurable at any time through the *sat_select*(1M) and

satconfig(1M) utilities. The *satconfig* utility is the preferred tool for use on graphics systems, since it provides a convenient graphical interface for switching each auditable event type on or off. The *sat_select* command is useful for server users and others who do not wish to use the *satconfig* utility. Both utilities are discussed in detail below.

What Should I Audit?

You can audit all system activity or certain types of activity, such as file removal or access denial. Users are tracked through the audit trail by User ID (UID) numbers. Any audited activity is associated with the UID of the person who performed that action. It is a central feature of the System Audit Trail that though the effective UID changes with the use of the *su*(1) command, the SAT ID does not. All of a user's actions after logging in are audited at the original login UID.

When you select the type of activities to audit, there are still several options for auditing. For example, if you wish to monitor the removal of files, you can generate an audit record under two conditions:

- When the action fails (sat_access_denied, sat_access_failed)
- When the action succeeds (sat_file_crt_del, sat_file_crt_del2)

Many different types of activities take place on your trusted computer system. There are login attempts, file manipulations, use of devices (such as printers and tape drives), and administrative activity. Within this list of general activities, you may choose to audit many specific kinds of actions.

Below is a list of auditable actions with a short definition of each action and one or more of the appropriate event types that can be audited. Important actions contain a note that they should always be audited:

- login (sat_ae_identity)

  Any login attempt, whether successful or not, should be audited.

- su (sat_check_priv, sat_ae_identity)

  Whenever a user invokes the *su*(1) command, whether to super-use some administrative account, such as **root** or another user account, the event should be audited. This is especially true for unsuccessful attempts, as they may indicate attempts at unauthorized access.

- chlabel and newlabel (file_attr_write, sat_proc_own_attr_write)

  Any time a user changes a MAC label on a Trusted IRIX/B system, it is wise to make an audit record of the event. (This does not happen under standard IRIX.)

- password change (sat_ae_identity)

  Whenever a user changes his or her password, it is wise to make an audit record of the event.

- administrative activity (sat_ae_mount, sat_clock_set, sat_hostid_set, etc)

  Any activity related to system administration should be carefully audited; for example, editing the */etc/fstab* file.

- DAC permissions change (sat_fd_attr_write, sat_file_attr_write)

  When a user invokes the *chmod*(1) command to change the DAC permissions on a file or the *chown*(1) command to change the ownership of a file.

- file creation (sat_file_crt_del, sat_file_crt_del2)

  Whenever a new link, file, or directory is created.

- file deletion (sat_file_crt_del, sat_file_crt_del2)

  Whenever a link, file, or directory is removed.

- process activity (sat_exec, sat_exit, sat_fork)

  When a new process is created, forked, exited, or killed.

  The audit administrator (**auditor**) can change the audited events by entering a new *sat_select* command. It is possible to change the selected event types at different times of day, by using the *cron* utility to execute *sat_select* periodically.

  To tailor your auditing for your specific needs, use the *sat_select* or *satconfig* utilities.

Auditable Events

The following is a complete list of auditable event types:

sat_access_denied
: Access to the file or some element of the path was denied due to enforcement of MAC or DAC permissions.

sat_access_failed
: Access to a file was denied because the path specified does not exist.

sat_chdir
: Current working directory was changed with *chdir*(2).

sat_chroot
: Current root directory was changed with *chroot*(2).

sat_open
: A file was opened with write permission.

sat_open_ro
: A file was opened read-only.

sat_read_symlink
: The contents of a symbolic link were read with *readlink*(2). Note that the file the link "points" to is not accessed in any way.

sat_file_crt_del
: A file was added or removed from a directory.

sat_file_crt_del2
: This is the same as sat_file_crt_del, but reports that two files (perhaps a link) were removed.

sat_file_write
: The data in a file was modified by *truncate*(2).

sat_mount
: A file system was mounted or unmounted.

sat_file_attr_read
: The attributes of a file were read by *stat*(2).

sat_file_attr_write
: The attributes of a file were written by *chmod*(2).

sat_exec
: A new process has been introduced by *exec*(2).

sat_sysacct
: System accounting has been turned on or off.

sat_fchdir
: The user changed current working directory to the directory "pointed" to by the given open descriptor.

sat_fd_read Information was read from a file descriptor using *read*(2).

sat_fd_read2 The same event as sat_fd_read, but with multiple file descriptors.

sat_tty_setlabel
The user set the label of a port via *ioctl*.

sat_fd_write The user finalized a change to a file descriptor.

sat_fd_attr_write
The user changed the attributes of the file "pointed" to by the given file descriptor using *fchmod*(2).

sat_pipe The user created an unnamed pipe.

sat_dup The user duplicated a file descriptor.

sat_close The user closed a file descriptor.

sat_proc_read The user read from a process's address space using *ptrace*(2).

sat_proc_write
The user finalized a changes to a process's address space using *ptrace*(2).

sat_proc_attr_read
The user read a process's attributes.

sat_proc_attr_write
The user finalized a change to a process's attributes.

sat_fork The user duplicated the current process (thereby creating a new process).

sat_exit The user ended the current process.

sat_proc_own_attr_write
Process attributes were changed.

sat_clock_set The system clock was set.

sat_hostname_set
The host name was set.

sat_domainname_set
The domain name was set.

sat_hostid_set The host ID was set.

sat_check_priv
: Action requiring superuser privilege was performed.

sat_control
: The *sat_select*(1M) command was used.

sat_svipc_access
: The user accessed a System V IPC data structure.

sat_svipc_create
: The user created a System V IPC data structure.

sat_svipc_remove
: The user removed a System V IPC data structure.

sat_svipc_change
: The user set some attribute of a System V IPC data structure.

sat_bsdipc_create
: The user created a socket.

sat_bsdipc_create_pair
: The user created a socket pair.

sat_bsdipc_shutdown
: The user shut down a socket.

sat_bsdipc_mac_change
: The user changed the MAC label on a socket.

sat_bsdipc_address
: A network address was used explicitly via the *accept*(2), *bind*(2), or *connect*(2) system calls.

sat_bsdipc_resvport
: A reserved port was successfully bound.

sat_bsdipc_deliver
: A packet was delivered to a socket.

sat_bsdipc_cantfind
: A packet was not delivered because the socket could not be found.

sat_bsdipc_snoop_ok
: A packet was delivered to a raw (snoop) socket.

sat_bsdipc_snoop_fail
: A packet was not delivered to a raw socket because it was prevented by MAC policy.

sat_bsdipc_rx_ok
: A packet was received on an interface.

sat_bsdipc_rx_range
: A packet was not received due to MAC violation not within the allowed label range on that interface.

sat_bsdipc_rx_missing
: A packet was received on an interface with a missing or damaged MAC label.

sat_bsdipc_tx_ok
: A packet was sent on the interface.

sat_bsdipc_tx_range
: A packet was not sent due to a MAC violation.

sat_bsdipc_tx_toobig
: A packet was not sent because the MAC label was too large for the IP header to contain.

sat_bsdipc_if_config
: An interface structure's attributes were changed.

sat_bsdipc_if_invalid
: Attempt to change MAC labels was disallowed for lack of MAC privilege.

sat_bsdipc_if_setlabel
: The MAC labels on an interface structure were changed.

All *sat_ae* events are used for application auditing, which means that a privileged program generated the record, rather than the kernel.

sat_ae_identity
: A login-related event occurred.

sat_ae_dbedit
: A file was modified using the *dbedit*(1M) utility. (This utility is available only with the Trusted IRIX/B optional product.)

sat_ae_mount
: An NFS file system was mounted.

sat_ae_custom

An application-defined event occurred. Application developers can engineer their applications to generate this event.

Using satconfig

satconfig is a graphical utility that you use to configure exactly which events will be audited on your system. Any user can invoke *satconfig*, but only the super-user may actually change the auditing environment. When you invoke *satconfig*, a new window opens on your screen. The main body of the window has a list of all the available event types. Next to each event type name is a "button," At any time, each button is either “up” or “down.” If the button is down, the event type is selected for auditing. If the button is up, the event type is not audited. Use your mouse and the left mouse button to select whether you want the event type in question to be on or off.

When you first begin using the audit trail, there is a default set of audited events. You can modify that selection using *satconfig*, but the *satconfig* window contains a pulldown menu labeled “edit” that you can use at any time to set the auditing environment to a few preset environments. These include the original SGI default audit selections, your local default selections, all event types selected, no event types selected, and a current events selection. The current events selection restores the auditing environment that was last saved on your machine. The local default environment can be any combination of event types that you choose. You create a local default environment by following the instructions in “Saving and Retrieving Your Auditing Environment” on page 467.

At the bottom of the *satconfig* screen there are three “buttons.” These buttons are labeled “Apply,” “Revert,” and “Quit.” When you have made your auditing selections, use the left mouse button to press the “Apply” button on the screen to activate the auditing selections. If you change your mind while making audit selections, you can use the “Revert” button to reset the individual event type buttons to the selections currently in use. The third button is labeled “Quit” and closes the *satconfig* window. If you have made selections that have not been applied, *satconfig* asks you if you really want to quit and discard the changes you have made without applying them.

Using sat_select

The *sat_select*(1M) utility is a character-based program that modifies your audit event type selections. Additionally, you can use the *sat_select* utility to change your local default auditing environment or to read in a preselected set of event type choices from a file. In this way, you can have several preset auditing environments ready in files for various situations and switch between them conveniently. If you have a graphical system, *satconfig* is the suggested utility for administering your auditing event type selections. *sat_select* exists for non-graphics systems and for making large-scale, file-oriented changes.

For complete information on using *sat_select*, consult the *sat_select*(1M) reference page, but in general, the syntax most often used is:

sat_select **-on** *event*

and

sat_select **-off** *event*

sat_select **-on** *event* directs the system audit trail to collect records describing the given event. If ''all'' is given as the **event** string, all event types will be collected.

sat_select **-off** *event* directs the system to stop collecting information on that event type. If ''all'' is given as the **event** string, all event types will be ignored.

sat_select issued with no arguments lists the audit events currently being collected. The effect of subsequent *sat_select* programs is cumulative. Help is available through the **-h** option.

Saving and Retrieving Your Auditing Environment

From time to time you may wish to change your auditing environment. You do this with the *sat_select* command. If you are making a temporary change, you may wish to save your current auditing environment for easy replacement. To do this, use the command:

```
sat_select -out > /etc/config/sat_select.options
```

Then, to restore auditing to the saved state, use the command:

```
sat_select 'cat /etc/config/sat_select.options'
```

The single quotation marks in the above example are crucial and must not be omitted.

You may save as many different audit states as you wish, in different filenames. Simply insert the filename of the state you wish to use in the above example. The */etc/config/sat_select.options* file is the default audit state file that is read at boot time. The */etc/config/sat_select.options* file must be labeled *dblow* if you are running Trusted IRIX/B, and you should restrict DAC file permissions to *root* only regardless of your operating system type.

Placing the Audit Files

The location of your audit record files is also configurable. You can direct your audit records to be saved to any location you desire, including magnetic tape. *satd* saves its input data in the directories or files named in its **path** arguments.

The **-f** option to *satd* specifies an output path, which may be a directory or a file. If the output path is a directory, *satd* creates and fills uniquely named files under that directory. (Files are named for the time of their creation. For instance, file *sat_9101231636* was created in 1991, on January 23, at 4:36 pm.) If the output path is a specific filename, *satd* writes to that file.

You can specify several output paths in the *satd* command line. To do so, you must precede each path with a **-f** or put commas (but no blank space) between each path name. Taken together, all of the output paths specified in the command line are known as the path list. Here are a pair of examples of command lines that contain path lists:

```
satd -f /sat1 -f /sat2 -f /sat3 -f /dev/null
satd -f /sat1,/sat2,/sat3,/dev/null
```

If no output paths are specified after the **-f** flag, the audit trail records are not saved anywhere, and the system halts. If a path given as a command line parameter is invalid for any reason, a warning is printed, that path is

omitted from the path list, and *satd* continues operating with whatever specified paths are valid. If the specified path does not already exist, *satd* creates a file with that name.

A file or directory is full when the file system on which it resides has no more available space. If a directory is specified as an output path, an audit file is constructed under that directory. When the audit file is filled to an internally specified maximum size, it is closed and a new audit file is created under that directory.

When one output path becomes full, *satd* replaces the current output path with a path that is not full. The method of replacement is configurable with the **-r** option. The output path is also replaced if *satd* receives a SIGHUP signal, for instance one sent with a *kill*(1) command.

If an output path becomes nearly full, warnings are displayed to the system console to notify the administrator to move the audit trail to tape. If all of the output paths become completely full, the system state moves to single-user mode with a very short grace period.

In order to protect against the loss of data due to sudden system state changes, when satd begins operations, it creates a file called */satd.reserve*, which is exactly 250,000 bytes long. If *satd* runs out of space, it immediately removes the *satd.reserve* file to free the 250,000 bytes for use to store audit records while the system moves to single-user mode. While the system is coming down, *satd* stores audit records in a series of files named */satd.reserve-n*, where *n* starts as 0. While *satd* is doing this, it issues a warning via *wall*(1) to all users that they have ten seconds before system shutdown.

If the file */satd.emergency-0* already exists, *satd* immediately moves to the first available filename, typically */satd.emergency-1*. To guard against this happening, a warning is issued at boot time if any */satd.emergency* files exist.

For complete information on customizing the location of your audit record files, see the *satd*(1M) reference page.

Understanding the Audit Data

The audit trail for an active system with full auditing can be too large for a single person to read and understand, and the entries in the trail that alert you to trouble are small and rare. If you were to read the raw audit trail to find an instance of policy violation, it would be like trying to find a needle in a haystack. Therefore, several utilities exist to help you reduce and interpret the raw audit data. The *sat_reduce*, *sat_interpret*, and *sat_summarize* commands can be used to remove superfluous information and format the audit history in succinct packages. See the reference pages for these commands for specific information on their usage.

After your raw data has been reduced and interpreted, an individual record looks something like this:

```
Event type = sat_ae_identity
Outcome = Failure
Sequence number = 5
Time of event = Mon Mar 11 12:46:13.33 PST 1991
System call = syssgi,SGI_SATWRITE
Error status = 0 (No error)
SAT ID = anamaria
Identity event = LOGIN|-|/dev/ttyq4|anamaria|That user gave
an invalid label.
```

The *sat_summarize* command provides a short listing of what types of records are in the audit trail and how many there are of each type. It's a useful tool for scanning the records quickly and identifying trends in system usage or consistent problems.

Remember that file pathnames within audit records are not the same as those in common usage through the shell on your system. Since the audit record is an exact log for security purposes, many attributes of the pathname that are designed to be transparent in normal usage are explicit in the audit log. For example, the double slash (//) means a directory level crossing (ordinarily represented through the shell with a single slash (/). A slash followed by an exclamation point (/!) indicates crossing a file system mount point. The slash and ampersand construction (/@) indicates that the path is following a symbolic link. If you are running Trusted IRIX/B, you may also see a slash followed by a right angle bracket (/>), which indicates that the directory level being crossed into is a multilevel directory. The *egrep*(1) utility supports this notation, so it is possible to specify this form of pathname notation in

regular expression searches. Below are two examples of audit record pathnames:

```
/usr/!orange2/@/fri//usr//src//lib//libmls//libmls.a
/u sr/!tmp/>L_e//sat//sat_9012280805
```

The system places the audit data in files on your system. Each file begins with the starting date and time of the file, the machine name, and the host ID and ends with the stopping date and time. If your system is interrupted (for example, by a power failure), the audit file being used at that time will have no ending entry. The audit daemon automatically closes a file when it reaches a certain manageable size and opens another. A new file is always started when the system is brought up. For information on these files and their format, see the *satd* reference page.

How to Audit a Specific User

At times, you may wish to examine the audit record of a particular user. For example, the user may have a history of violations of system security or may simply be leaving the project and an accounting of activity may be required.

If the user in question is being audited to determine if attempted security violations are taking place, use the command line:

`sat_reduce -P satfile | sat_summarize -u username`

This command line selects only the audit records that represent attempted violations. The *-P* flag to *sat_reduce* selects for attempted violations. The *-u* flag to the *sat_summarize* command lists the number of records generated by the user.

It is vitally important to remember that not every record of an attempted violation really represents malicious intent on the part of the user! Most of these records are generated in the course of normal work. The auditor should be looking for a trend, such as repeated attempts to access information unnecessary in the course of normal work (for example, a programmer attempting to access salary or hiring information).

In the second scenario, where the employee is leaving the project, the auditor is looking for a comprehensive list of files used by that employee so that the

correct files and directories may be assigned a new owner who is remaining on the project.

The above listed command line provides a basic look at the user's activity. Next, to more closely examine the user's activities, issue the following command line :

```
sat_reduce -u username satfile | sat_interpret | more
```

The *sat_reduce* command selects all of the audit records generated by the user. Then, the *sat_interpret* command puts the records into human readable form. The output of *sat_interpret* is very large. If it is impractical to direct this output to a file, you should direct the output to your screen and view it with a screen paging program such as *more*(1).

Using these two command lines, you should be able to view a user's activities and come to a reasonable knowledge of the types of actions the user is taking on the system. You can also generate a specific record, in human-readable form, of all security violations or files and resources accessed.

How to Audit a File

At times, you may wish to examine all audit records pertaining to an individual file. Perhaps some changes have been made to an important file and the user who made those changes must be identified. Or perhaps an accounting of all access to a sensitive file is needed. To obtain a record for each time the file was opened, you must first make certain that the audit daemon is recording *sat_open* and *sat_open_ro* events. Use the *sat_select* command to ensure that these events are logged. To search the audit log for these events, use the following command line:

```
sat_reduce -e sat_open -e sat_open_ro satfile |
sat_interpret | grep filename
```

How to Audit a Label Under Trusted IRIX/B

If you are using Trusted IRIX/B, your system supports Mandatory Access Control labels on all files and processes. This section explains how to check the audit trail of a given security label.

If you are using standard IRIX, your system does not support MAC labels and attempts to read the audit trail for events relating to such labels will be futile.

Since the number of configurable labels in Trusted IRIX/B is great enough for each project or portion of a project at your site to have its own label, you may sometimes need to audit a specific label to generate a record of activity on that label. Use the following command to generate a log of activity on a label:

`sat_reduce -l label satfile`

The above command chooses only audit records that pertain to the given label. The following command syntax allows you to select more than one label for your report:

`sat_reduce -l label -l label2 satfile`

Once you have obtained output from *sat_reduce*, use the other auditing utilities, such as *sat_interpret* or *sat_summarize*, to view it according to your needs.

Potential Security Violations

The overwhelming majority of records in an audit trail are the result of the normal actions of users doing their jobs. No automated tool exists to locate records that signify the actions of abusers trying to violate system security. Nonetheless, an administrator can apply some general rules to detect abuse or violation of security policy. This list of tips is neither complete nor universal. Each administrator must customize the list to meet the particular needs of each site.

Use and Abuse by Outsiders

Intrusion by outsiders is among the most feared of abuses. Fortunately, this kind of abuse produces distinctive audit record patterns and is easily detected. Below, we have identified several different subcategories of outsider abuse that can be detected by the audit system. Note though, that these kinds of patterns can also be generated by an authorized user who makes a mistake or is misinformed. Patterns of this type are described below.

Attempts at Unauthorized Entry

All attempts at unauthorized entry generate audit records of the *sat_ae_identity* event type. (Use *sat_select*, *sat_reduce*, and *sat_interpret* to collect and view these records) The interpreted output of these events contains a text string that describes the attempt at entry. Intruders from outside your organization have a much higher instance of failed login attempts than your authorized users.

Three interesting text strings reveal attempts at unauthorized entry:

- Unsuccessful login attempt
- That user gave an invalid label
- Could not set the label connection for device

Here is an example of an interpreted audit record of an unsuccessful login attempt:

```
Event type = sat_ae_identity
Outcome = Failure
Sequence number = 1
Time of event = Mon Mar 11 12:45:40.34 PST 1991
System call = syssgi,SGI_SATWRITE
Error status = 0 (No error)
SAT ID = anamaria
Identity event = LOGIN|-|/dev/ttyq4|guest|Unsuccessful login
attempt.
```

System Usage at Unusual Hours or From Unusual Locations

Usage of your system outside of normal working hours or, if your system maintains physical security of terminals, from unusual locations, is a matter

of interest. In most cases, the usage of the system is legitimate, but each instance certainly bears notation and examination. Many potential violations of security from outside your user community happen during

nonpeak hours, and rarely from within your physical site. To observe activity at odd hours, issue the following commands in order:

1. **sat_reduce -a start_time satfile > /usr/tmp/early+late**
2. **sat_reduce -A end_time satfile >> /usr/tmp/early+late**
3. **sat_reduce -U root -U sys -U daemon -U adm -U lp /usr/tmp/early+late > /usr/tmp/e+l_ordusers**
4. sat_interpret /usr/tmp/e+l_ordusers | more

If your site assigns a terminal to each user and maintains reasonable physical security for each terminal, you can monitor logins from unusual locations. For example, if a user normally working in a group computer lab makes a login attempt from a private office, this event may be cause for interest. To dget a list of login events, enter the following command:

```
sat_reduce -e sat_ae_identity sat_file | sat_interpret | grep LOGIN
```

Bear in mind that it does not necessarily represent a violation of security if a user is working at an unusual terminal or even if a user is logged on at two or more terminals at once. For instance, the user may be correcting a mistake and may have logged in elsewhere explicitly for the purpose of terminating unwanted processes. You should be looking for instances where the user is not genuinely logged in twice, but where one instance of the login is an intruder.

Connections with Machines Outside the Local Network

Whenever a user connects to a machine outside your trusted local network, an audit record should be generated. A connection to a host outside of the local network is worthy of notice but not necessarily a violation of security. You should be on the lookout for trojan horse programs that cause your system to make the outward connection at a later time. You can identify outward network connections with the following command sequence:

1. **`sat_reduce -e sat_bsdipc_addr`** *satfile* **`>`** */usr/tmp/connect*
2. **`sat_interpret`** */usr/tmp/connect* **`>`** */usr/tmp/connect.int*
3. **`grep -n "Remote host"`** */usr/tmp/connect.int*

The above command sequence is dependent on the specific implementation of your networking software. You may need to modify your command line to reflect your networking situation. For example, if the software you are using does not generate the *sat_bsdipc_addr* auditing event type, you should search for another event type that is generated.

Use and Abuse by Insiders

Beyond use and abuse by intruders, unfortunately, the possibility arises of abuse from within your organization. The following types of events are the most common instances of security violations. It is extremely counterproductive to assume that a security violation on the part of an authorized user indicates that the user is not trustworthy or is involved in some attempt to break security for malicious purposes. Most violations of system security by users involve a failure on the part of the Administrator to adequately prepare the working environment. Users are most concerned with accomplishing their work tasks, not with fixing the computer system to provide themselves with the correct tools. Therefore, you should not be suspicious of the user who violates security unless a clear pattern of a specific and unnecessary security violation is apparent.

File Permission Violations

Although the system records each instance where access to a file or resource is denied, the information contained in these audit records is rarely indicative of a security violation. Many applications and utilities operate on a principle of access denial as part of normal operation. These events are always logged, but only in rare cases do they indicate a violation. For example, the library function *getutent*(3) always tries to open */etc/utmp* for read-write access. If this action fails, *getutent* immediately tries again, but requesting read-only access. Permissions on */etc/utmp* prohibit all users except **root** from opening this file for reading and writing. When an unprivileged user runs a program that calls *getutent()*, a *sat_access_denied* record is generated, and it is immediately followed in the audit trail by a

sat_open_ro record, indicating that access was granted. The lesson in this example is that access denial is usually not indicative of a security violation.

The *sat_access_failed* event is often confused with the denial event. The event type is completely different and is even more rarely a cause for concern than access denial. When a user enters a command to an interactive shell (such as */bin/csh*), the shell tries to execute the command in each directory in the user's search path, failing at each attempt until it finds a directory that actually contains the command. Suppose a user enters **xterm** and his or her path variable contains:

```
/bin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/bin/X11:~/bin
```

A *sat_access_failed* record is generated for each directory in the path until the command is found and executed. In this scenario, a record of failed access is generated for each of the following nonexistent programs: */bin/xterm*, */usr/bin/xterm*, */usr/sbin/xterm*, */usr/local/bin/xterm* and a successful *sat_file_exec* record for the real program: */usr/bin/X11/xterm*.

Unexpected Use of Root Privilege

Every interpreted audit record contains a line beginning with the keyword "Outcome." The field following this keyword can be equal to one of "Success," "Failure," or "Success due to privilege." The last case indicates that the user made a system call that would have failed except that Superuser privilege was invoked to assure its successful completion. This is not necessarily a security violation or an unexpected use of **root** privilege. It is perfectly normal to see these outcomes. Any time an ordinary user runs a program that contains code that uses **root** privilege, "Success due to privilege" outcomes are generated. A good example of this kind of program is *passwd*(1M). An ordinary user generates a record of this type simply by changing the password on his or her account.

What you should be looking for is an instance where the "SAT ID" or "Effective ID" field is different from the "User ID" field. This occurs when a user executes */bin/su* to gain **root** privileges or otherwise promotes the privilege level of a session. In most cases, this is not a security violation, since the **root** password is necessary to successfully complete the */bin/su* command.

An instance of using Superuser privilege, though, is always worth examination in the audit trail. When you encounter an instance where a user has promoted his or her login session to **root**, you should check to see that the user is authorized to know the **root** password. If not, check whether the user indeed executed the */bin/su* command, or if he or she promoted the privilege of the session by some other means, such as a trojan horse *setuid* shell command.

Whenever a user runs */bin/su* and thereby promotes the privilege of his or her login session, the **auditor** should also make a routine check of what actions the user took while the privilege was promoted.

Activity by Particularly Interesting Users

Sometimes a particular user is under official scrutiny by the management of a site. He or she may be on probation or may have just left employment under less than ideal circumstances. The **auditor** can choose to look at the records describing that user's behavior just by directing the audit trail through the *sat_reduce* command as follows:

1. **`sat_reduce -u` jeff `<` `satfile` `>` /tmp/sat.jeff**
2. **`sat_interpret` /tmp/sat.jeff `|` `more`**

Rarely should any user be subjected to this kind of accounting, and this feature should be used carefully and with consideration of the individuals involved.

Access to Particularly Interesting Files or Resources

Sometimes a particular file or resource is of special interest. An information leak may have occurred and an investigation is proceeding into how the leak took place. Or a special file or resource may have been created as bait to trap browsing intruders. In either case, the file or resource is going to be closely accounted by the **auditor**.

`sat_reduce -n` interesting_file `-e sat_open -e sat_open_ro` sat_fi*lename*
`|` `sat_interpret`

Proper and Improper Management

Frequently, actions taken by the Administrator or **root** result in unusual audit records. With the enhanced privilege of these accounts, it is not unusual for more audit records of potential concern to be generated. Again, it is rare for a record to be generated that cannot be explained by the normal usage of the system or by simple human error.

Modifications of System Data Files

Every modification of system data files is of interest to the **auditor**. Since these data files are not only under system security but in fact define system security, any unauthorized access can result in a total breach of security.

Each site has individual policies on how users are added to or removed from the system, how access control of files and hardware is administered, how network connectivity is maintained and administered, and a host of other issues. It is the responsibility of the **auditor** at each site to enforce the policies of the site and to use the auditing tool effectively to exercise that responsibility.

If you are running Trusted IRIX/B, system data files should be modified only with the dedicated editing tool, *dbedit*(1), and never with general-purpose text editors. Only privileged users can use the *dbedit* tool, and only privileged users have permission to alter the contents of the system data files. Any use of any other editor on a system data file is a violation of security policy and should be noticed by the **auditor**. If your interpreted audit trail contains *sat_open* records where the "Actual name" field contains the string "/secadm", check that the "Process ID" field (which gives both the PID and the name of the program being executed) does not contain "vi", "ex", "emacs" or any other commonly available text editor. This field should contain only the name "dbedit".

Modifications of Attributes of System Programs

The Administrator should never modify permissions, ownership, or labels of system programs. If your audit trail contains evidence that the administrator has attempted to change attributes of system programs, you should investigate and find the reason for the change. Again, the explanation given is likely to be valid, and this is not good cause to suspect

your Administrator of subterfuge; however, you may want to examine your system's security policies and make certain that neither the users nor the administrators take a cavalier attitude toward the security policies.

The following command searches your audit trail for the type of records that can indicate this problem:

```
sat_reduce -e sat_file_attr_write -e sat_fd_attr_write <
satfile
```

In the interpreted output, look for lines with the "Actual name" field. Any audit records showing modified attributes for resources in */bin*, */sbin*, */etc*, */lib*, */var*, */usr/bin*, */usr/lib*, */usr/share*, */usr/bsd*, */usr/sbin*, or */usr/bin/X11* is an audit record deserving follow-up.

Manipulation of the Audit Trail

The **auditor** should be the only person to access the audit trail. No other users should read from it, write to it, remove files, or modify file attributes. Look at all records generated by people other than the one who knows the **auditor** account password, and check that none of those records refer to files in */var/adm/sat* or in any other directory you use to store audit trail information.

Archiving Audit Data

Since the audit trail is stored in ordinary system files, archiving your audit data is as easy as making a backup tape. Archive your audit data to conserve disk space but do keep copies of your audit trail; evidence of intrusion and damage to your system may not always be apparent immediately, and the ability to research your audit trail over time can be very valuable in tracking down a security breach. You can use the *compress*(1) utility to reduce the size of your old audit files by up to 80 per cent.

Removing Audit Data

Since the audit trail is stored in ordinary system files, once it has been archived, audit trail files can be safely removed. If you enter the *df* command (disk free) and determine that the file system containing your audit trail is more than 90 per cent full, you should remove old audit files. If your audit files are kept in */var/adm/sat*, give the command:

```
df -k /var/adm/sat
```

The output should be similar to this:

```
Filesystem  Type blocks  use   avail %use Mounted on
/dev/root   efs  245916 218694 27222  89%    /
```

In this example, the file system is 89 per cent full, and the **auditor** should archive and remove audit trail files.

Recovering from Audit File Overflow

Do not allow your audit files to grow too large. Oversized audit files can use up your available disk space and cause the system to refuse new records and immediately cease operations. This can result in lost work and lost audit records. Maintain at least 10 per cent free space in your audit file system at all times.

The audit daemon, *satd*(1M), must always be running on your system. The daemon eventually becomes unable to write to the audit file if free disk space drops to 0 per cent. When it can no longer write to the audit file, the daemon exits with an error, and the system changes the run level to single-user mode. You must then archive and remove the audit files to free disk space before bringing the system back to multi-user mode. If the *satd* daemon is somehow killed or interrupted on your system, the system changes the run level to single user mode immediately. The daemon will be respawned when the system is brought back up.

To make space on the disk for your audit trail, first boot the system into single-user mode. No audit records are generated in this mode. Once in single user mode, archive your audit files and remove them from the disk. Once at least 10 per cent of the file system is free, you may boot into multiuser mode without difficulty.

If your auditing system directs the audit files to the / (root) file system or the */usr* file system and either file system becomes full, you will not be able to bring the system to single-user mode to archive and remove your old audit files. If you find yourself in this situation, perform the following procedures to remove old audit files:

1. Boot the system from the original distribution media, and allow the *inst* utility to start up.
2. At the *inst* main menu, select the *admin* menu, and then select the *shell* option from the *admin* menu. You see a shell prompt.

   From the shell, you must archive and remove the old audit files. Remember that when your system is running the *inst* (also called *miniroot*) shell, your system's root directory appears as:

   ```
   /root/
   ```

   rather than:

   ```
    /
   ```

   and your */usr* file system appears as:

   ```
   /root/usr
   ```

   because your system's file systems are mounted on the *inst* file system.
3. Once you have created free disk space on your / (root) and */usr* file systems, you should be able to boot your system normally. If this is a recurring problem, you should refer to the *satd*(1M) reference page for information on changing the location of your audit files.

Summary

This chapter explained the concepts and procedures for administering the System Audit Trail. The types of auditable events were listed and explained, and the procedure for customizing your system's auditing was covered. The default auditing environment was also listed and discussed.

The procedure for reducing and interpreting the audit data was explained, and several scenarios for specific auditing were given. Guidelines for rational interpretation of audit data and pointers for discerning actual abuse of the system from normal use were outlined. The usage of the commands *sat_reduce*(1), *sat_summarize*(1), and *sat_interpret*(1) was also explained.

Chapter 14

System Accounting

Chapter 14 describes the system accounting facilities available with IRIX. With these utilities, you can keep track of system usage on a user-by-user basis. Topics covered here include:

- *Setting up process accounting.*
- *Maintaining process accounting.*

Chapter 14

System Accounting

This chapter deals with

- Using the accounting utilities to keep track of system use. See "Process (System) Accounting" on page 485.

Process (System) Accounting

IRIX provides utilities to log certain types of system activity. These utilities perform *process accounting*.

The IRIX process accounting system can provide the following information:

- the number of programs a user runs
- the size and duration of user programs
- data throughput (I/O)

Using this information, you can:

- Determine how system resources are used and if a particular user is using more than a reasonable share.
- Trace significant system events, such as security breaches, by examining the list of all processes invoked by a particular user at a particular time.
- Set up billing systems to charge login accounts for using system resources.

The next sections describe the parts of process accounting, how to turn on and off process accounting, and how to look at the various log files.

Parts of the Process Accounting System

The IRIX process accounting system has several parts:

- The IRIX kernel writes a record of each process on the system that terminates into the file */var/adm/pacct*. The file contains one record per terminated process, organized according to the format defined in */usr/include/sys/acct.h.*

You must specifically turn on this function. See "Turning on Process Accounting" on page 487.

- Once process accounting is turned on, the *cron* program executes several accounting commands, as specified in */var/spool/cron/crontabs/adm* and */var/spool/crontabs/root.* The commands in *adm* perform monthly accounting (*monacct*), check the size of the *pacct* file (*ckpacct*), and provide a daily accounting of processes and connect time (*runacct*). The *root* crontab file runs the *dodisk* program, which provides a report on current disk usage. These commands run automatically when process accounting is turned on.
- The *login* and *init* programs record connect sessions by writing records into */etc/wtmp*. This happens by default, as long as the *wtmp* file exists.
- Records of date changes, reboots, and shutdowns are copied from */etc/utmp* to */etc/wtmp* by the *acctwtmp* command.

- The *acctwtmp* utility is automatically called by *runacct*, */usr/lib/acct/startacct*, and */usr/lib/shutacct*, once process accounting is turned on.
- The disk utilization programs *acctdusg* and *diskusg* break down disk usage by login and prepare reports. For more information on disk usage quotas, see "The quotas(4) Subsystem" in Chapter 3. These programs are run by the *dodisk* script.

Turning on Process Accounting

To turn on process accounting:

1. Log in to the system as **root**.
2. Enter this command:

   ```
   chkconfig acct on
   ```

3. Enter this command:

   ```
   /usr/lib/acct/startup
   ```

   This starts the kernel writing information into the file */var/adm/pacct*.

Process accounting is started every time you boot the system, and every time the system boots, you should see a message similar to this:

```
System accounting started
```

Note that process accounting files, especially */var/adm/pacct*, can grow very large. If you turn on process accounting, especially on a server, you should watch the amount of free disk space carefully. See "Controlling Accounting File Size" on page 488

Turning Off Process Accounting

To turn off process accounting, follow these steps:

1. Log in as **root**.
2. Enter this command:

```
chkconfig acct off
```

3. Enter this command:

   ```
   /usr/lib/acct/shutacct
   ```

   This stops the kernel from writing accounting information into the file */var/adm/pacct*.

Process accounting is now turned off.

Controlling Accounting File Size

Process and disk accounting files can grow very large. On a busy system, they can grow quite rapidly.

To help keep the size of the file */var/adm/pacct* under control, the *cron* command runs */usr/lib/acct/ckpacct* to check the size of the file and the available disk space on the file system.

If the size of the *pacct* file exceeds 1000 blocks (by default), it runs the *turnacct* command with argument "**switch**." The "**switch**" argument causes *turnacct* to back up the *pacct* file (removing any existing backup copy) and start a new, empty *pacct* file. This means that at any time, no more than 2000 blocks of disk space are taken by *pacct* file information.

If the amount of free space in the file system falls below 500 blocks, *ckpacct* automatically turns off process accounting by running the *turnacct* command with the "off" argument. When at least 500 blocks of disk space are free, accounting is activated again the next time *cron* runs *ckpacct*.

Accounting Files and Directories

The directory */usr/lib/acct* contains the programs and shell scripts necessary to run the accounting system. Process accounting uses a login (*/var/adm*) to perform certain tasks. */var/adm* contains active data collection files used by the process accounting. Here is a description of the primary subdirectories in */var/adm*:

/var/adm/acct/nite contains files that are reused daily by *runacct*.

/var/adm/acct/sum contains the cumulative summary files updated by *runacct*.

/var/adm/acct/fiscal contains periodic summary files created by *monacct*.

Daily Operation

When IRIX enters multiuser mode, */usr/lib/acct/startup* is executed as follows:

- 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 by *turnacct*, which, in turn, executes *acct* on */var/adm/pacct*.
- *remove* is executed to clean up the saved *pacct* and *wtmp* files left in the *sum* directory by *runacct*.

The *ckpacct* procedure is run through *cron* every hour of the day to check the size of */var/adm/pacct*. If the file grows past 1000 blocks (default), the *turnacct* switch is executed. The advantage of having several smaller *pacct* files becomes apparent when you try to restart *runacct* after a failure processing these records.

The *chargefee* program can be used to bill users for file restores, etc. It adds records to */var/adm/fee* that are picked up and processed by the next execution of *runacct* and merged into the total accounting records. *runacct* is executed through *cron* each night. It processes the active accounting files, */var/adm/pacct*, */etc/wtmp*, */var/adm/acct/nite/disktacct*, and */var/adm/fee.* It produces command summaries and usage summaries by login name.

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 administrator should execute */usr/lib/acct/prdaily* to print the previous day's accounting report.

Setting Up the Accounting System

If you have installed the system accounting option, all the files and command lines for implementation have been set up properly. You may wish to verify that the entries in the system configuration files are correct. In order to automate the operation of the accounting system, you should check that the following have been done:

1. The file */etc/init.d/acct* should contain the following lines (among others):

   ```
   /usr/lib/acct/startup
   /usr/lib/acct/shutacct
   ```

   The first line starts process accounting during the system startup process; the second stops it before the system is brought down.

2. For most installations, the following entries should be in */var/spool/cron/crontabs/adm* so that *cron* automatically runs the daily accounting. These lines should already exist:

   ```
   0 4 * * 1-6 if /etc/chkconfig acct; then
   /usr/lib/acct/runacct 2> /var/adm/acct/nite/fd2log; fi
   ```

   ```
   5 * * * 1-6 if /etc/chkconfig acct; then
   /usr/lib/acct/ckpacct; fi
   ```

   Note that the above *cron* commands appear on one line in the source file. The following command, which is also all on one line in the source file, should be in */var/spool/cron/crontabs/root*:

   ```
   0 2 * * 4 if /etc/chkconfig acct; then
   /usr/lib/acct/dodisk > /var/adm/acct/nite/disklog; fi
   ```

3. To facilitate monthly merging of accounting data, the following entry in */var/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:

   ```
   0 5 1 * * if /etc/chkconfig acct; then
   /usr/lib/acct/monacct; fi
   ```

   The above command is all on one line in the source file, and 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 when *runacct* has sufficient time to complete. This will, on the first day of each month, create monthly accounting files with the entire month's data.

4. You may wish to verify that an account exists for *adm*. Also, verify that the PATH shell variable is set in */var/adm/.profile* to:

   ```
   PATH=/usr/lib/acct:/bin:/usr/bin
   ```

5. To start up system accounting, simply type the commands:

   chkconfig acct on

   and

   /usr/lib/acct/startup

The next time the system is booted, accounting will start.

runacct

runacct is the main daily accounting shell procedure. It is normally initiated by *cron* during nonpeak 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*, reads the *wtmp* file and produces usage statistics for each terminal line on the system. This report is especially useful for detecting bad lines. If the ratio between the number of logoffs to logins exceeds about 3/1, it is quite possible that the line is failing.
nite/daytacct	The total accounting file for the previous day in *tacct.h* format.
sum/tacct	The accumulation of each day's *nite/daytacct* 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 last login shell procedure. It maintains a record of the last time each *login* name was used.

sum/rprtMMDD
Each execution of *runacct* saves a copy of the daily report that can be printed by *prdaily.*

runacct takes care not to damage files in the event of errors. A series of protection mechanisms are 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* are written into *fd2log. runacct* will complain if the files *lock* and *lockl* 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*, 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. A file is used to remember the last state completed. When each state completes, *statefile* is updated to reflect the next state. After processing for the state is complete, *statefile* is read and the next state is processed. 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, */var/adm/pacct?,* are moved to */var/adm/Spacct?.MMDD*. The */etc/wtmp* file is moved to */var/adm/acct/nite/wtmp.MMDD* with the current time added on the end.

WTMPFIX The *wtmpfix* program checks the *wtmp* file in the *nite* directory for correctness. 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 of *ctmp.h. T*he lineuse file is created, and the reboots file is created showing all of the boot records found in the *wtmp* file.

ctmp is converted to *ctacct.MMDD*, which are connect accounting records. (Accounting records are in *tacct.h* format.)

The *acctprc1* and *acctprc2* programs are used to convert the process accounting files, */var/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 will not occur. One precaution should be noted: when restarting *runacct* in this state, remove the last *ptacct* file, because it will not be 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 the *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/tacctMMDD*, so that *sum/tacct* can be recreated in case it is 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/rprtMMDD*, remove the locks, then exit.

Recovering from a Failure

The *runacct* procedure can fail for a variety of reasons—usually due to a system crash, */usr* running out of space, or a corrupted *wtmp* file. If the *activeMMDD* 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 */var/adm/acct/nite/lock* and */var/adm/acct/nite/lock1* were found. These files must be removed before *runacct* can restart.

- ERROR: acctg already run for date: check /var/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: /var/adm/acct/nite/wtmp.MMDD already exists, run setup manually

  Self-explanatory.

- ERROR: wtmpfix detected a corrupted wtmp file. Use fwtmp to correct the corrupted file.

  Self-explanatory.

- ERROR: connect acctg failed: check /var/adm/acct/nite/log

  The *acctcon1* program encountered a bad *wtmp* file. Use *fwtmp* to correct the bad file.

- ERROR: Invalid state, check /var/adm/acct/nite/active

  The file *statefile* is probably corrupted. Check *statefile* for irregularities and read *active* before restarting.

Restarting runacct

The *runacct* program, 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* will rerun 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*, use the command:

```
nohup runacct 2 /var/adm/acct/nite/fd2log &
```

To restart *runacct*:

```
nohup runacct 0601 2 /var/adm/acct/nite/fd2log &
```

To restart runacct at a specific state:

```
nohup runacct 0601 WTMPFIX 2 /var/adm/acct/nite/fd2log &
```

Fixing Corrupted Files

Sometimes, errors occur in the accounting system, and a file is corrupted or lost. You can ignore some of these errors, or simply restore lost or corrupted files from a backup. However, certain files must be fixed in order to maintain the integrity of the accounting system.

Fixing wtmp Errors

The *wtmp* files are the most delicate part of the accounting system. When the date is changed and the IRIX system is in multiuser 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. However, some combinations of date changes and reboots will slip through *wtmpfix* and cause *acctcon1* to fail.

The following steps show how to fix a *wtmp* file:

1. `cd /var/adm/acct/nite`
2. `fwtmp < wtmp.MMDD > xwtmp`
3. `ed xwtmp`
4. Delete any corrupted records or delete all records from beginning up to the date change.
5. `fwtmp -ic <wtmp> wtmp.MMDD`

If the *wtmp* file is beyond repair, remove the file and create an empty *wtmp* file:

6. `rm /etc/wtmp`

7. **`touch /etc/wtmp`**

This prevents any charging of connect time. *acctprc1* cannot determine which *login* owned a particular process, but it will be 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 looks all right, the latest *sum/tacct.MMDD* should be patched up, then *sum/tacct* recreated. A simple patchup procedure would be:

1. Give the command:

   `cd /var/adm/acct/sum`

2. Give the command:

   `acctmerg -v < tacct.MMDD > xtacct`

3. Give the command:

   `ed xtacct`

4. Remove the bad records.
5. Write duplicate UID records to another file.
6. Give the command:

   `acctmerg -i < xtacc t > tacct.MMDD`

7. Give the command:

   `acctmerg tacctprev <tacct.MMDD> tacct`

Remember that the *monacct* procedure removes all the *tacct.MMDD* files; therefore, you can recreate *sum/tacct* by merging these files.

Updating Holidays

The file */usr/lib/acct/holidays* contains the prime/nonprime 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, which may appear anywhere in the file as long as the first character in the line is an asterisk.
- Year Designation Line, which should be the first data line (noncomment line) in the file and must appear only once. The line consists of three fields of four digits each (leading white space is ignored). For example, to specify the year as 1992, prime time at 9:00 a.m., and nonprime time at 4:30 p.m., the following entry is appropriate:

  ```
  1992 0900 1630
  ```

  A special condition allowed for in the time field is that the time 2400 is automatically converted to 0000.
- Company Holidays Lines, which follow the year designation line and have the following general format:

  ```
  day-of-year Month Day Description of Holiday
  ```

  The day-of-year field is a number in the range of 1 through 366, indicating the day for the corresponding holiday (leading white space is ignored). The other three fields are actually commentary and are not currently used by other programs.

Daily 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 users were logged in. The following paragraphs describe the reports and the meanings of their tabulated data.

In the first part of the report, the from/to banner should alert the administrator to 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. See the *acct*(1M) reference page for more information.

The second part of the report is a breakdown of line utilization. The TOTAL DURATION field tells how long the system was in multiuser state (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 of the accounting period.
# SESS	The number of times this port was accessed for a *login*(1) session.
# ON	This column has little significance. It previously gave the number of times that the port was used to log a user on; but since *login*(1) can no longer be executed explicitly to log in a new user, this column should be identical with SESS.
# OFF	The number of times a user logged off and also any interrupts that occur on that line. Generally, interrupts occur on a port when the *getty*(1M) is first invoked after the system is brought to multiuser state. This column comes into play when the # OFF exceeds the # ON by a large factor. This 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* should be monitored, since this is the file from which connect accounting is geared. If it grows rapidly, execute *acctcon1* to see which line is the noisiest. If the interrupting is occurring at a furious rate, general system performance will be affected.

Daily Usage Report

The daily usage report gives a by-user breakdown of system resource utilization. Its data consists of:

UID	The user ID.

LOGIN NAME
: The *login* name of the user; more than one *login* name can exist for a single user ID, and this entry identifies which login name used the resource.

CPU (MINS)
: The amount of time the user's process used the central processing unit. This category is broken down into PRIME and NPRIME (nonprime) utilization. The accounting system's idea of this breakdown is located in the */usr/lib/acct/holidays* file. As delivered, prime time is defined to be 0900 through 1700 hours.

KCORE-MINS
: 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 broken down into PRIME and NPRIME amounts.

CONNECT (MINS)
: The amount of time that a user was logged into the system. If this time is high and # OF PROCS is low, this indicates that the user was logged in for a long period of time without actually using the system. This column is also subdivided into PRIME and NPRIME utilization.

DISK BLOCKS
: When the disk accounting programs have been run, the output is merged into the total accounting record (*tacct.h*) and shows up in this column. This disk accounting is accomplished by the program *acctdusg*.

OF PROCS
: The number of processes invoked by the user. Large numbers in this column indicate that a user may have had a shell running out of control.

O SESS
: Number of times the user logged onto the system.

DISK SAMPLES
: Number of times 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 field represents the total accumulation of widgets charged against the user by the *chargefee* shell procedure.

See *acctsh(1M).* The *chargefee* procedure is used to levy charges against a user for special services performed such as file restores, and so on.

Daily Command and Monthly Total Command Summaries

These two reports are virtually the same except that the Daily Command Summary reports only on the current accounting period, while the Monthly Total Command Summary tells the story for 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 tells an administrator which commands are used most heavily. Based on those commands' characteristics of system resource utilization, the administrator can decide what to weigh more heavily when system tuning.

These reports are sorted by TOTAL KCOREMIN, which is an arbitrary yardstick but often a good one for calculating ''drain'' on a system.

COMMAND NAME
: The name of the command. Unfortunately, all shell procedures are lumped together 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* or *core* or any other name that does not seem quite right. Often people like to work on their favorite version of a personal program, but they do not want everyone to know about it. *acctcom* is also a good tool for determining who executed a suspiciously named command and also to see if superuser privileges were abused.

NUMBER CMDS
: The total number of invocations of this particular command.

TOTAL KCOREMIN
: The total cumulative measurement of the amount of kilobyte segments of memory used by a process per minute of run time.

TOTAL CPU-MIN
: The total processing time this program has accumulated.

TOTAL REAL-MIN
: The total real-time (wall-clock) minutes this program has accumulated. This total is the actual "waited for" time as opposed to kicking off a process in the background.

MEAN SIZE-K
: The mean of the TOTAL KCOREMIN over the number of invocations reflected by NUMBER CMDS.

MEAN CPU-MIN
: The mean derived between the NUMBER CMDS and TOTAL CPU-MIN.

HOG FACTOR
: This gives a relative measure of the total available CPU time consumed by the process during its execution. It is a measurement of the ratio of system availability to system utilization. It is computed by the formula:

  total CPU time / elapsed time

CHARS TRNSFD
: This column, which may contain a negative value, is a total count of the number of characters pushed around by the *read*(2) and *write*(2) system calls.

BLOCKS READ
: A total count of the physical block reads and writes that a process performed.

Files in the /var/adm Directory

The files listed here are located in the */var/adm* directory:

diskdiag	diagnostic output during the execution of disk accounting programs
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 by *turnacct*
Spact?.MMDD	process accounting files for MMDD during execution of *runacct*

Files in the /var/adm/acct/nite Directory

The following files are located in the */var/adm/acct/nite* directory:

active	used by *runacct* to record progress and print warning and error messages. *activeMMDD* is the same as *active* after *runacct* detects an error
cms	ASCII total command summary used by *prdaily*
ctacct.MMDD	connect accounting records in *tawcct.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* (see *cron* entry)
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*
logMMDD	same as log after *runacct* detects an error
reboots	contains beginning and ending dates from *wtmp* and contains 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

wtmperrorMMDD
: same as *wtmperror* after *runacct* detects an error

wtmp.MMDD
: previous day's *wtmp* file

Files in the /var/adm/acct/sum Directory

The following files are located in the */var/adm/acct/sum* directory:

cms
: total command summary file for current fiscal period 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*

pact.MMDD
: concatenated version of all *pacct* files for **MMDD**, removed by remove procedure after reboot

rprtMMDD
: saved output of *prdaily* programs

tacct
: cumulative total accounting file for current fiscal period

tacctprev
: same as *tacct* without latest update

tacctMMDD
: total accounting file for **MMDD**

wtmp.MMDD
: saved copy of *wtmp* file for **MMDD**, removed by remove procedure after reboot

Files in the /var/adm/acct/fiscal Directory

The following files are located in the */var/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?**

Summary of IRIX Accounting

The IRIX accounting system is designed to work as smoothly as possible. However, it is a complex system. If you plan to use the accounting system, it is a good idea to study carefully the reference pages for the various accounting programs. Also, keep accurate records in your system log books of how you set up the accounting system and how you changed it.

Chapter 15

Understanding Silicon Graphics Networking Products

Chapter 15 introduces the networking software shipped with IRIX. The optional networking products are also mentioned. Topics covered in this chapter are:

- *An overview of networking hardware and software.*
- *A list of optional networking software products.*
- *An overview of the standard network configuration.*
- *Instructions on starting and stopping your network.*

Chapter 15

Understanding Silicon Graphics' Networking Products

This chapter provides information about the standard hardware and software networking products provided with Silicon Graphics systems. It explains the physical connection of an IRIS system to an Ethernet and serial network and describes network hardware options and interface names for network devices. This chapter describes the standard networking files, directories, and daemons, and provides an overview of the network startup and shutdown processes. It also supplies a brief description of Silicon Graphic's optional networking products.

Topics covered in the remaining chapters of this guide require an understanding of the fundamentals of network theory and operation. If you need information on networking fundamentals, refer to the bibliography in the introduction to this guide for additional reading. Topics in this chapter include:

- An overview of networking hardware. See "Networking Hardware" on page 508.
- An overview of networking software. See "Networking Software" on page 510.
- A list of optional networking software products. See "Optional Networking Products" on page 511.
- A look at the standard network configuration. See "Standard Software Configuration" on page 513.
- Starting and stopping your network. See "Network Startup and Shutdown" on page 519.

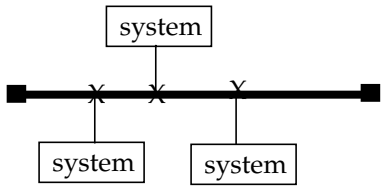
Networking Hardware

The networking hardware that comes standard on every Silicon Graphics system is an Ethernet controller and 2 serial ports. (Some hardware products may have more ports than this, including an ISDN port.) The Ethernet controller may be an entire board or an integrated chip. Controllers interface between the networking software and the network medium.

To connect your Ethernet controller to a network, you must have this hardware:

- an Attachment Unit Interface (AUI) cable, also referred to as a dropline or even simply as a cable
- a transceiver
- access to an active Ethernet cable

Figure 15-1 shows how systems (termed "stations" on the network) might be connected to an Ethernet network.



■ = terminator
—— = AUI
X = transceiver

Figure 15-1 Ethernet Network Attachment

The serial ports on an IRIS allows it to connect to serial networks. Serial line networks are systems connected by serial lines and modems. You do not need special hardware installed in your computer to connect to a serial network.

Figure 15-2 shows systems connected to a serial network using modems.

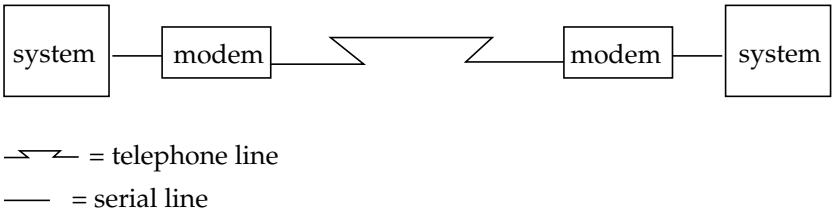


Figure 15-2 Serial Line Network

Networking Hardware Options

In addition to Ethernet and serial-line hardware, other types of controllers can be installed in Silicon Graphics systems as options. Some optional hardware products are user installable, while others require installation by a System Support Engineer certified by Silicon Graphics.

Some optional networking hardware is listed below.

- FDDIXPress
- EFast
- Token Ring
- X.25 Option Board
- Coaxial Emulator Optional
- 3270 Board
- 5080 Gateway
- IRIS SNA Option Board

Controller Interface Names

The network controller is the physical board or chip. The interface is software's interpreter and handler of the controller. The interface name is the name most evident to the user. For example, network management tools

refer to the interface name when providing information about the physical controller.

To configure a controller, each network controller on a system must have a valid interface name. A single system may have multiple controllers; each controller must have a unique interface name. Several different types of controllers are available. Each type has its own special interface name. Most network software supports a maximum of four network interfaces by default.

Some standard and optional interface names are listed in Table 15-1, where "*" is 0, 1, 2, or 3.

Table 15-1 Controller Interface Names

Controller Type	Interface Name
Ethernet	ec*, et*, or enp*
Efast	fxp*
FDDI	ipg* or xpi*
Token Ring	tr*

Networking Software

The standard networking software shipped with all IRIS systems adheres to the Internet Model standards and protocols. It is derived from the networking software in the 4.3BSD UNIX release from the University of California at Berkeley and the RPC (remote procedure call) system from Sun Microsystems. The IRIX® operating system implements the Internet Protocol suite and UNIX domain sockets using the 4.3BSD UNIX socket mechanism. The system also supports access to the underlying network media by means of raw sockets.

All standard networking software is supplied on the Execution Only Environment media (eoe1, eoe2, and netls_eoe). See Table 15-2 for a list of

standard networking software for IRIS systems. See Table 15-3 for a list of the optional networking products for IRIS systems.

Table 15-2 Standard Networking Software

Standard Networking Software	Description
TCP/IP	Transmission Control Protocol/ Internet Protocol support
UUCP	UNIX-to-UNIX Copy Programs
Sendmail	Electronic mail program
SLIP	Serial Line Internet Protocol
BIND	Berkeley Internet Name Domain
NETLS	Network License Server
NCS	Network Computing System (supports NETLS only)
RPC	Remote Procedure Call

Optional Networking Products

Silicon Graphics Inc. supplies a variety of optional networking products, both hardware and software, to provide interconnectivity between various vendors and mediums. Table 15-3 lists and briefly describes some of the available optional networking products. See your sales representative for detailed product information.

Table 15-3 Optional Networking Products

Optional Networking	Product Description
NFS	Includes software for Network File System (NFS), Network Information System (NIS, formerly YP), and Diskless system support.
4DDN	Enables IRIS 4D systems or servers to function as a Phase IV DECnet end node.

Table 15-3 (continued) Optional Networking Products

Optional Networking	Product Description
4DLT	Provides DECnet terminal service. (LAT)
Network License Server Developers Option	Consists of the License Server Lock (LSLOCK) and the Network License Server (LSSERVER). The LSLOCK allows software developers to license software products and LSSERVER is used to administer products licensed with LSLOCK.
NetVisualyzer	Offers a set of graphical traffic monitoring, diagnostic, planning, and performance analysis tools that provide network information and statistics in a visually intuitive form.
FDDIVisualyzer	Provides a graphical interface to the FDDI environment.
Efast	A high-performance Ethernet network interface board for POWER and Professional Series graphics systems and servers.
IRIS NetWorker	Application that automatically backs up systems over the network. Keeps online indices of all backed up files.
4D TCP 3270	Enables IRIS systems to emulate an IBM 3270-type terminal and open multiple sessions on an IBM mainframe.
IRIS 5080 Emulator	Provides IBM 5080 and 3270 terminal emulation. Delivers direct access to models, applications, and data residing on an IBM mainframe using the IRIS system.

Table 15-3 (continued) Optional Networking Products

Optional Networking	Product Description
4D Coax Connectivity	The 4D CUT 3270 and 4D DFT 3270 coax products provide your system with a cost-effective way to emulate an IBM 3270-type terminal.
4D SNA Connectivity	Allows access to the IBM SNA environment. Provides access to mainframe applications, utilizes multiple windows and file transfer programs.

Standard Software Configuration

The standard software configuration can be broken into two main categories:

- Files and directories
- Daemons

Note: See Chapter 21, "UUCP," for information regarding UUCP files, daemons, and tools. See Chapter 22, "SLIP and PPP," for information regarding SLIP files, daemons, and tools. NetLS and NCS are covered in the *NetLS Administration Guide*.

Files and Directories

Most of the standard software configuration files reside in */etc*, */etc/config*, and */etc/init.d*. A brief description of each file is provided. Usage examples are provided throughout the guide where applicable. See the online reference page for more details.

The following are the network configuration files that reside in the */etc* directory:

ethers(4) This file is used by the Reverse Address Resolution Protocol (RARP) or *bootp* for mapping physical (MAC) addresses to logical (Internet) addresses. It must be built and maintained

	by the administrator. If this file is required, it should be updated when a system's network controller board is replaced. The applications discussed in this guide do not depend on the *ethers* file; however, other common network controllers, such as FDDI, are dependent upon this file.
hosts(4)	This file contains the Internet or logical address database. It is used by all network programs based on the Internet Protocol. There must be an Internet address for all systems with which this system communicates. The Network Information Service (NIS) and/or the Berkeley Internet Name Daemon (BIND), implementing the Domain Name System) can significantly impact the function of this file. See Chapter 16, "Planning a Network" for more details.
hosts.equiv(4)	This file contains a list of "trusted users," where trusted users can be system names or system and user names. Programs like *rlogin*, *rcp*, *rdist*, and *rsh* use this file to determine the amount of validity checking to be done. This file is used in conjunction with the */etc/passwd+* and *.rhosts+* files.
networks(4)	This file contains the network name database. Administrators typically modify the file to contain information on local site networks (network addresses and names). The database is used by *netstat*(1M) to provide network number-to-names translation.
protocols(4)	This file contains the protocol name database. Specifically, it lists all known Internet Protocols by official name, number, and any aliases. This database is used by *inetd* to determine protocol support.
rpc(4)	This file lists all Remote Procedure Call (RPC) programs (*portmapper*, *ypserv*, *mountd*, etc.) and their respective program numbers. The *rpcinfo*(1M) and *inetd*(1M) programs use the information contained in this file.
services(4)	This file lists the ports associated with well-known (TCP/UDP). It is used by *inetd* to determine a server's port number. (RPC based applications get their port number assignments from the *portmap* program and */etc/rpc* database.)

sys_id(4) This file contains the system's host name. This file is accessed by various network applications to determine the identity of the system.

This list contains the standard networking configuration files located in the */etc* directory:

inetd.conf The configuration file used by the Internet super server, *inetd*, to determine which servers and daemons to start. You can modify this file to disallow access to various network services. See the *inetd*(1M) online reference page for additional information.

resolv.conf(4) This file determines hostname-address resolution order. It can be used to override default system lookup services, such as the resolve order and the domain. This file must be configured on systems that want to be BIND or DNS clients. See the *resolver*(3N) and *named*(1M) online reference pages for additional information.

mrouted.conf This is the configuration file for the *mrouted* daemon. Use this file to override the default multicast routing configuration. This file is also used to add multicast tunnel links between a local and remote network. See the *mrouted*(1M) online reference page for additional information.

ipfilterd.conf This file contains the configuration information for the *ipfilterd* daemon. It contains macro and filter definitions. Some example macros are provided in the file. See the *ipfilterd*(1M) online reference page for additional information.

snmpd.auth This file is used by the *snmpd* daemon to authenticate each incoming *snmp* request. It specifies get and set privileges for sets of systems and SNMP communities. The default configuration provides all systems from all communities with get privileges. See the *snmpd*(1M) online reference page for additional information.

gated.conf Configuration file for the *gated* daemon. The configuration file can be used to configure *gated* to handle up to three routing protocols: Routing Information Protocol (RIP),

Exterior Gateway Protocol (EGP), and HELLO. The default configuration supports RIP and HELLO. See the *gated*(1M) online reference page for additional information.

Daemons

There are three general network-related daemons that should always be running to support basic TCP/IP communications. Obviously, if you have optional network software loaded, other daemons are present. The general daemons are started by the master network configuration script, */etc/init.d/network*, at boot time. All daemons are found in the */usr/etc* directory. The three general network daemons are:

inetd(1M) Also known as the super server. It listens for requests to certain network services. When it gets a request, it checks its configuration file, */etc/inetd.conf*, to determine how and on which port to start the server. Three other files, */etc/services*, */etc/rpc*, and */etc/protocols* are also addressed by *inetd* to get the appropriate port number and protocol information.

portmap(1M) The daemon that converts RPC program numbers into TCP/UDP port numbers for Sun RPC. It must be running to support any RPC based network applications (NIS, NFS, *rstatd*, etc.). When *inetd* detects an RPC based connection, it contacts the portmap and gets the port number for the RPC connection. *inetd* starts the server on the port number supplied by the portmapper.

routed(1M) Manages (adds, updates, deletes, propagates) the kernel routing tables on and between systems on a network. It is based on the Routing Information Protocol (RIP).

Other common network daemons are:

named(1M) The Internet domain name server, also known as BIND. This daemon implements the Domain Name System (DNS). It loads the BIND database from the designated boot file, reads in the initial data, and listens and responds to host name or address queries. Site-dependent options and arguments reside in the */etc/config/named.options* file.

mrouted(1M)	The Internet Protocol multicast routing daemon. *mrouted* forwards a multicast datagram to all networks reachable by a cooperating set of *mrouted* routers. If the intermediary routers do not support multicast routes, *mrouted* supports "tunnels," which provide a virtual point-to-point link between pairs of *mrouted* systems located anywhere on the network. *mrouted* runs on any interface capable of multicasting. To override the default configuration or to add tunnel links, modify the */etc/mrouted.conf* file.
timed(1M)	The time server daemon. It averages a system's time with the time of other systems on a network running *timed*. It was designed for small and homogeneous networks. A *timed* master must be designated with a list of selected trusted systems (slaves) from which clock averaging is performed. One of the slaves serves as a backup if the master becomes unavailable. See */etc/config/timed.options* to override defaults.
timeslave(1M)	The server daemon to slave a local clock to a common clock. *timeslave* is more efficient than *timed*. It consumes less network bandwidth, fewer CPU cycles, and less memory on both the local and remote systems. It was designed for large, heterogeneous networks and should be used in conjunction with *timed*. The */etc/config/timeslave.options* file is not optional, it is required.
rarpd(1M)	The DARPA Reverse Address Resolution Protocol daemon. It responds to RARP requests. This daemon maps physical addresses (Ethernet) to logical addresses (Internet). For the *rarpd* daemon to answer requests, there must be a valid entry for the client in the server's */etc/hosts* and */etc/ethers* files. To override default configurations, see the file */etc/config/rarpd.options*.
rwhod(1M)	The system status server. It maintains a database used by the *rwho* and *ruptime* applications. It must be running to support *rwho* and *ruptime*. Site-dependent options and arguments should go in the */etc/config/rwhod.options* file. Do not run this daemon on a net with lots of systems; it could saturate the net. By default, *rwhod* is off.

gated(1M) The gateway routing daemon that handles multiple routing protocols and replaces *routed*, *egpup*, and any other routing daemon that speaks the HELLO routing protocol. *gated* runs on the router that interfaces with an exterior router running HELLO. It handles the Routing Information Protocol (RIP), the Exterior Gateway Protocol (EGP), and HELLO. It can be configured to support any combination of the three protocols by modifying the */etc/gated.conf* file. Default configuration supports RIP only.

rtnetd(1M) The daemon that allows pre-emptable network packet processing. This daemon allows higher-priority real-time processes to preempt processing of incoming network packets, thus allowing better response for real-time processes. It should be run on multiprocessor systems for best throughput. To override defaults, see the */etc/config/rtnetd.options* file.

snmpd(1M) The daemon for the Simple Network Management Protocol (SNMP). *snmpd*, also called the SNMP agent, listens for SNMP services queries. Each SNMP request is checked for privileges in the file */etc/snmpd.auth*.

Daemon Option Files

Site-dependent options for daemons are set in their respective configuration files in the directory */etc/config*. Table 15-4 lists the required configuration flag, the daemon's function, and the options file. Consult the */etc/init.d/network* script and each daemon's reference page for details.

Table 15-4 Network Configuration Option Files

Options File	*chkconfig* Flag	Function
gated.options	gated	Cornell Internet super-routing daemon
mrouted.options	mrouted	Stanford IP multicast routing daemon
named.options	named	4.3BSD Internet domain name server
routed.options	routed	RIP routing daemon

Table 15-4 (continued) Network Configuration Option Files

Options File	*chkconfig* Flag	Function
rtnetd.options	rtnetd	Pre-emptable networking process for real-time use
rwhod.options	rwhod	4.3BSD system status daemon
timed.options	timed	4.3BSD time synchronization daemon
timeslave.options	timeslave	SGI time synchronization daemon (this file is required for timeslave)
rarpd.options	none	Reverse Address Resolution Protocol
inetd.options	none	*inetd* options file
netif.options	none	Site-dependent interface options
portmap.options	none	Sun RPC portmap options
ifconfig-.options* (*=1, 2, 3, 4, or hy)	none	Interface configuration options

Network Startup and Shutdown

The main network script is */etc/init.d/network*. Other scripts for other network applications (UUCP, mail, etc.) also reside in this directory, but are covered in their appropriate chapter in this guide. A brief description of the *network* script is provided:

The *network* master script is called during system startup and shutdown. It defines the system name and host ID, ensures that the system has a valid Internet address, starts networking daemons, and initializes the network interfaces. Site-dependent configuration commands to start and stop local daemons, add static routes, and publish arp entries should be put in a separate shell script called */etc/init.d/network.local*. Make symbolic links from */etc/rc0.d* and */etc/rc2.d* to */etc/init.d/network.local* so the *network.local* file is called at system startup and shutdown (see "Creating a Local Network Script" on page 580 for setup procedure).

The *network* master script is linked to */etc/rc0.d/K40network*, which is invoked from */etc/rc0* during shutdown, and to */etc/rc2.d/S30network*, which is

invoked from */etc/rc2* during startup. The script understands two arguments: **start** and **stop**. It can be run manually for testing and troubleshooting network-related problems without having to reboot the system.

Network Initialization Process

During system initialization, the shell script */etc/init.d/network* is called. These are the actions performed by the script at start up:

1. Checks host name and Internet address to determine if system should be configured as standalone or networked. Checks *sys_id* and *hosts* files. If the **network** configuration flag is off, the system is configured for *standalone operation*.
2. Determines names and addresses or primary and router interfaces for typical configurations.
3. Obtains any site-dependent information for interfaces from the *netif.options* file.
4. If system is not diskless, the shell script flushes all old routes.
5. Configures all interfaces, including loopback, using the *ifconfig* command.
6. If configured for IP packet filtering, the shell script starts the IP packet filtering daemon (*/usr/etc/ipfilterd*). The *ipfiltered* daemon must be started before gateway interface initialization.
7. Initializes gateway interface.
8. Initializes additional interfaces specified in the *netif.options* file.
9. If specified, initializes the Hypernet interface according to the *ifconfig-hy.options* file.
10. Initializes the loopback interface.
11. Using the *chkconfig* command, determines daemon configuration and reads relevant daemon configuration files (**.options*).
12. Sets default route for all IP multicast packets to the primary interface.
13. If NIS software is configured, defines and sets NIS domain name.
14. If NIS software is configured, starts appropriate NIS daemons.

15. If NFS software is configured, starts appropriate NFS daemons and mounts any NFS file systems listed in the */etc/fstab*.
16. If configured on with *chkconfig*, it starts standard daemons (*inetd*, *timed*, *timeslave*, *rarpd*, *rwhod*, *snmpd*, etc.).

Network Shutdown Process

During system shutdown, */etc/init.d/network* stops the daemons and disables the network devices. These are the actions the script performs at system shutdown:

1. Kills all network services that may be associated with a shell (*rlogind, rexecd, rshd, ftpd, telnetd,* etc.).
2. Kills some network daemons immediately (*inetd, bootp, tftpd, snmpd,* etc.).
3. If NFS is running, unmounts remote file systems.
4. Kills all remote daemons.
5. If NFS is running, unexports exported file systems. See the *NFS Administration Guide* and the *NIS Administration Guide* for complete information about the optional NFS software.
6. Kills daemons that must be kept alive until the last minute (*portmap, slip, ipfiltered*)
7. Gracefully takes the system off the FDDI ring, if it is on the ring.
8. Stops the *ypbind* process of NIS.

Chapter 16

Planning a Network

Chapter 16 is designed to help you plan your network before you begin. Networks are complex, and it pays to have a plan before you begin. Read through this chapter before you set up your network. Topics covered include:

- *The physical layout of your network.*
- *Internet addresses.*
- *A list of common network applications and their uses.*
- *Instructions for creating subnetworks at your site.*
- *Network security information.*

Chapter 16

Planning a Network

This chapter contains common sense approaches to planning the physical and logical aspects of your network environment. The information contained in this chapter should be read prior to setting up a new network or integrating into an existing network (homogeneous or heterogeneous). The following topics are covered:

- Planning the physical layout of your network. See “Planning the Physical Network” on page 525.
- Helpful information for connecting to the Internet is provided in “The Internet” on page 528.
- A complete description of Internet addresses and how to obtain one. See “Internet Addresses” on page 531.
- A listing of common network applications and their uses. See “Format of Internet Protocol (IP) addresses” on page 532.
- Instructions for creating subnetworks within your site. See “Planning to Subnet Local Networks” on page 539.
- Information on maintaining network security. See “Planning for Network Security” on page 541.

Planning the Physical Network

Planning the physical network requires that you first answer the question, "What network media and topology configuration would best suit the needs of my users?" A review of the MAC (Medium Access Control) level and application-level performance information about the products you are considering will help you determine the appropriate choice of media for your environment. In your review, consider the size (number of stations) of your network. Your network size will influence the media type and topology you choose for your network. If your network requires different types of

media, determine whether you have the correct equipment for integrating the various media types.

The subsections that follow will help you answer this list of planning questions:

- What will my physical network look like?
- Do I have a map of my network?
- Will I need a repeater, bridge, router, and/or gateway?
- Will this network configuration meet my user's needs?
- Where are my performance bottlenecks? Can I reduce or avoid them?

Repeaters, Bridges, Routers, and Gateways

Your choice of media and the number of stations, networks, and protocols in your network may require the use of a repeater, bridge, router, or gateway. This section suggests the type of device required for certain network functions.

Repeater	A device that regenerates and amplifies electrical signals. Its purpose is to extend the physical length of a network.
Bridge	A device that decodes MAC-layer frames transmitted between different hardware and media. Its purpose is to resolve network media differences; it allows a network to be composed of various media types (Ethernet, fiber, serial, etc.). It can also be used to segment similar media types.
Router	A device that decodes and passes network-layer packets between different networks. Its purpose is to provide the physical and logical route from one network to another.
Gateway	A device that translates protocols from one station to another. Its purpose is to allow stations with different networking protocols to communicate successfully.

Note: The terms *router* and *gateway* are sometimes used interchangeably. Be sure you know the function of the device you are considering, as the term may be technically inaccurate.

Note that each device may not be limited to a single function. For example, a gateway may also perform router functions if it is configured as a router. Table 16-1 summarizes the characteristics of each network device.

Table 16-1 Network Device Characteristics

Device Name	Media	Protocol	LAN	Purpose
repeater	same	same	same	extends physical length of the network
bridge	different/same	different/same	same/different	bridge network media differences
router	same/different	same	different	provides physical and logical route between networks
gateway	same/different	different	same/different	communication between stations with different networking protocols

Performance Planning

You can circumvent some performance bottlenecks with appropriate planning. These bottlenecks might occur as a result of your choice of media, topology, number of network devices, controller boards, or your network design.

Choice of Media

Be sure the capacity of the medium you have selected is adequate for the network size and data transmission type (large or small volumes of data, sporadic or steady traffic). For example, Ethernet has a range of capacities depending on the specific type of Ethernet cable used (10base5, 10base2, or 10baseT). Media type is also a factor in data degradation. For example, 10baseT is unshielded twisted pair and is more sensitive to environmental conditions than 10base5. This should be a consideration if you are planning a manufacturing network that has a high degree of electrostatic discharge.

Number of Devices
: Network devices can cause degradation to the network performance. Use repeaters only when necessary. Each additional devices introduces additional resistance onto the network.

Choice of Controller
: Choose the most efficient controller for your media. For example, Silicon Graphics supplies a standard Ethernet controller. An optional Efast card will handle more of the protocol processing in hardware and frees the station's CPU for other processing.

Design of Network
: Think about the design of your network before you begin setting it up. If possible, put departments that interact heavily on the same network to decrease router traffic. Use dedicated routers to handle heavy traffic between networks.

The Internet

Chances are, you will want to connect your system or network to the Internet. Wherever you may be, there is likely an Internet gateway available within your local calling range. The following sections offer some information that should help you get set up and running. Obviously, each situation is somewhat different, and your local service provider will have variations in service and equipment. Some research and experimentation is usually required before everything works smoothly.

Helpful Information about Connecting to the Internet

Before you sign up for an internet connection, consider what level of service you need. For example, if you are an individual looking for basic E-mail, news and file transfer capabilities, it probably wouldn't make sense to install a dedicated network cable in your home for economic reasons. A better choice for single user access might be to subscribe to a network provider who establishes an account for you on their system (One who is currently

connected to Internet). Typically access to their system is through a modem connection.

If you are trying to establish a connection to the Internet for a corporation, you will likely need the bandwidth of a network cable, and all the required hardware that goes with it, including an IP router, CSU/DSU (In effect a digital modem to interface the cable to the router). You will have to take into consideration the many administrative issues of running a site. These issues include, but are not limited to:

- Cost Analyses/Budget
- Establishing a domain
- Applying for IP addresses
- Establishing site policy
- Establishing site security
- Administration of network services (i.e.Domain Name Services, NIS, Mail, etc.)

There are providers of network connectivity that can provide varying levels of service. You must investigate the providers, and decide who provides the level of service you need, at the appropriate cost.

If you are going for an individual account on a provider's machine, the service provider deals with most, if not all the administrative tasks, and you simply enjoy access to the Internet.

If you would like a somewhat broader range of services, most providers will set you up with a dedicated modem and phone line for your exclusive use, or they can provide a network-only service (using SLIP, PPP, or UUCP) either through modems or other network connections.

If you are trying to set up internet access for a company, or corporation, you should research the issues listed above. Based on the information you obtain, formulate a plan for your site based on the needs and expectations of your organization. One of the best sources of information is the Internet itself. You should first obtain an individual account from a local provider. With the individual account, you can gain access to a large amount of information pertaining to establishing a site on the Internet.

Information Sources Available On-line

With an individual account, or access to the Internet you can get the information you need to provide access to your own site.

Usually, the provider of an individual account will also provide new user documentation that describes the basics of using the Internet. There are FTP (File Transfer Protocol) sites available on the Internet that have a wealth of information on many subjects, including Internet connectivity.

If you have never used *ftp*, here is a brief summary on how to use the ftp command. Anonymous FTP is a conventional way of allowing you to sign onto a computer on the Internet in order to obtain copies of files that are made available to the public. Some sites offer anonymous FTP accounts to distribute software and various kinds of information. Many systems will allow any password and request that the password you choose is your userid. If this fails, the generic password is usually "guest".

Note: In order to obtain a file through an ftp connection the command you will use is *get*. The *get* command copies one file from the remote system to your local system. If you wish to obtain multiple files from the remote system use the *mget* command.

On FTP.RS.INTERNIC.NET, in the *rfc* directory, you will find *fyi16.txt*, entitled *Connecting to the Internet - What Connecting Institutions Should Anticipate*. This paper describes the issues and processes of establishing a site. There is no better documentation than this for establishing a site.

Another important set of problems you will face when connecting a site to the net is that of security. On `moink.nmsu.edu`, there is a directory called *firewalls* that contains two files in postscript format that give an overview of firewall configurations.

On FTP.RS.INTERNIC.NET, in the *rfc* directory, you will find *fyi8.txt*, entitled *Site Security Handbook*. This handbook is a guide to setting computer security policies and procedures for sites that have systems on the Internet.

Other places to look are on FTP.RS.INTERNIC.NET:

ddn-news	DDN Management Bulletins
domain	Root Domain Zone Files

iesg	IETF Steering Group
ietf	Internet Engineering Task Force
internet-drafts	Internet Drafts
netinfo	NIC Information Files
netprog	Guest Software (ex. whois.c)
protocols	TCP-IP & OSI Documents
rfc	RFC Repository
scc	DDN Security Bulletins
std	Internet Protocol Standards

Internet Addresses

An Internet address is required if your network is attached to the Internet. The Internet Network Information Center (InterNIC) is responsible for assigning the network portion of an Internet address for each site. For example, if Company A applies for an Internet address, the InterNIC provides the network portion of the Internet address for the entire Company A site. A centralized organization within Company A is responsible for assigning and managing the station ID portion of the Internet address.

Internet addresses are maintained on each station or in a centralized network database such as NIS or BIND. See "Format of Internet Protocol (IP) addresses" on page 532. Each station that wishes to communicate must have a valid Internet address registered in the appropriate database. The standard hosts name-address database on IRIX stations is the */etc/hosts* file.

The subsections that follow will help you answer this list of planning questions:

- How do I get a valid Internet address for my site?
- Do I understand the purpose of the */etc/hosts* file?
- Do I have valid Internet addresses ready for all required stations?

Format of Internet Protocol (IP) addresses

An IP address is a 32-bit number that network software uses to identify a system on a network. The format is ###.###.###.###. Every system on an IP network must have its own unique IP address for the network to function properly.

Note: Unlike a system's Ethernet address, a system's IP address is determined by the network and network system administrators.

Conceptually, each IP 32-bit IP address is a pair of numbers where one number is representing the network and the other the system itself. There are 5 classes of addresses determined by the highest bits of the network.

The Format of Internet Protocol (IP) addresses:

CLASS A (128 networks with 16,777,214 systems each):

```
Bit #s 0 1-----7 8---------------------31
      |-|-------|------------------------|
      |-|-------|------------------------|
IP Address 0 NETWORK HOST
```

CLASS B (16,384 networks with 65,534 systems each):

```
Bit #s 01 2-----------15 16------------31
      |--|--------------|----------------|
      |--|--------------|----------------|
IP Address 10 NETWORK HOST
```

CLASS C (2,097,152 networks with 254 systems each):

```
Bit #s 012 3-----------------23 24----31
      |---|---------------------|--------|
      |---|---------------------|--------|
IP Address 110 NETWORK HOST
```

CLASS D (multicast group address within a network site):

```
Bit #s 0123 4-------------------------31
      |----|----------------------------|
      |----|----------------------------|
IP Address 1110 MULTICAST ADDRESS
```

CLASS E - Not implemented.

To simplify Internet addressing, dotted decimal notation is used to break the 32-bit number into 4 decimal numbers separated by dots.

Example: The IP address 128.74.41.123 in binary is:

```
10000000| 01001010| 00101001| 01111011
```

or

```
128 | 74 | 41 | 123
```

IP addresses of different classes in valid dot notation conform to the following specifications:

```
Class A -- 001.hhh.hhh.hhh through 126.hhh.hhh.hhh*
Class B -- 128.001.hhh.hhh through 191.254.hhh.hhh*
Class C -- 192.000.001.hhh through 223.255.254.hhh*
```

*where `hhh` is the local system and the leading numbers are the network.

Classes D and E are special cases:

```
Class D -- multicast within a network site.
Class E -- reserved for future use.
```

If a system is to be on a network that is part of the Internet, then the IP address for the system must be obtained according to the information provided in “Obtaining an Internet Address” on page 533.

However, coordinate with your local network administrator as he or she may already have an allotment of IP addresses available for your site.

Obtaining an Internet Address

You should obtain an Internet network address from the InterNIC before you begin setting up your network.

InterNIC Required Information

To request an Internet network address, supply the following information to the InterNIC:

1. Your administrative point of contact (POC). The administrative POC is the person responsible for answering administrative and policy questions about the network. You need to know his/her name, title, mailing address, and phone number.
2. Your technical point of contact (POC). The technical POC is responsible for the technical support of the network. You need to know his/her name, title, mailing address, and phone number.
3. Your network name (up to 12 characters).
4. Your network's geographic location and organization name.
5. The name and location of the network document plan.
6. Gateway information (connectivity, hardware, software, address).
7. The approximate size of your network, initially and within five years.
8. Justification for other than a Class C network number.

Contacting the InterNIC

You may send your request for an Internet Number by electronic mail to:

hostmaster@INTERNIC.NET

Or you can transfer the files:

templates/domain-template.txt

templates/in-addr-template.txt

using anonymous FTP from the site:

FTP.RS.INTERNIC.NET

Or you may mail your hardcopy request through the public mails or phone the InterNIC at:

Network Solutions
Attn: InterNIC Registration Services
505 Huntmar Park Drive
Herndon, VA 22070
Phone: 1-800-444-4345 or 1-703-742-4777

Internet Addresses and the Hosts Database

This section discusses the *hosts* database. Procedures for setting up a *hosts* file are described in Chapter 17, "Setting Up a Network."

The host name-address database on IRIX stations correlates an Internet address with the name of each station on the network. This information must exist on each station on a network. When a host name is referenced by an application program, the program accesses the */etc/hosts* database, or equivalent database, with the *gethostbyname*(3N) routine to find the internet address of the station.

/etc/hosts

The */etc/hosts* database is an ASCII file that you can modify with any text editor. The file contains lines of text that specify a station's address, its "official" name, and any aliases. The "official" name should be the fully qualified domain name. The address and name(s) are separated by blanks, tabs, or both. Comments begin with a pound sign (#) and continue to the end of the line.

An */etc/hosts* database is shown in this sample */etc/hosts* file:

```
# This is a comment
127.0.0.1   localhost
119.0.3.20  mint.spices.com mint  # mint is an alias
119.0.3.21  ginger.spices.com  ginger
119.0.3.22  sassafras.spices.com  sassafras sas
```

Each IRIS must have a copy of */etc/hosts* that contains entries for **localhost** and all of its network interfaces. As shipped, the */etc/hosts* database contains two entries. The first entry is a name you can use to test the local network software:

```
127.0.0.1 localhost
```

When you reference *localhost*, the message is looped back internally; it is never transmitted across the network.

Caution: Many important programs depend on the **localhost** entry—*do NOT remove or modify it.* If the master copy of */etc/hosts* is not maintained on an IRIS station or if you are using BIND or NIS, make sure that the host database contains the **localhost** entry.

The second entry in the *hosts* file is the default address and name for your IRIS. To enable the IRIS to access the network, change this entry to contain a newly assigned IP address and the name in */etc/sys_id.* The entry *must* contain the *sys_id* name, either as the official host name or as an alias.

Using the example */etc/hosts* file above, the */etc/sys_id* file for the host *ginger* should contain either "ginger" or "ginger.spices.com".

If you change the IRIS's name in */etc/sys_id*, make sure to update the entry in */etc/hosts;* otherwise the network software will not initialize properly. If the following message appears during station startup, then the */etc/hosts* and */etc/sys_id* files are inconsistent and must be fixed:

```
*** Can't find hostname's Internet address in /etc/hosts
```

If your IRIS is a gateway, each network interface must be assigned an Internet address and have an entry in */etc/hosts,* as described in "Setting Up a Router" on page 560.

It is important that each station have a consistent version of the *host* database. The proper method for maintaining the consistency depends on the size of your network and whether the network is connected to the Internet.

Alternatives to the Local Hosts Database

For a small network of stations under the same administrative control, maintaining a consistent */etc/hosts* database is straightforward. Establish a master copy on one station and make additions or deletions from its file. Then use *rcp*(1C) or *rdist*(1C) to copy the file to the other stations in the network.

Maintaining consistent versions of */etc/hosts* on every station in a large network is troublesome. NIS and/or the BIND name server make maintenance easier by providing a centralized version of the host database. See "Format of Internet Protocol (IP) addresses" on page 532 for additional information about BIND and NIS.

Using Common Network Applications

This guide is written specifically to support the standard network hardware and software – Internet protocols over Ethernet. However, when discussing networking in general, it is difficult, if not impossible, to ignore network applications that are not standard, but are common to most network environments. This section presents a brief overview of some of the common network applications that you should consider when planning your network.

The subsections that follow will help you answer this planning question:

- What network applications do I need on my network?

Electronic Mail

Electronic mail is a group of programs (*sendmail*) used to send and receive messages to and from users on the same local station or between remote stations. Mail can be sent using UUCP or TCP/IP protocols. IRIX supports both System V (*/bin/mail)* and 4.3BSD (*/usr/sbin/Mail*) mail programs.

UNIX-to-UNIX Copy Program (UUCP)

UUCP, also called the Basic Networking Utilities, is a set of utilities that lets stations using a version of the UNIX operating system (such as IRIX) communicate with each other over serial lines. The utilities provided range from those used to copy files between computers to those used for remote login and command execution.

You may consider setting up UUCP for long haul communications using modems and telephone lines. It is usually used to distribute electronic mail and network news.

Serial Line Internet Protocol (SLIP)

SLIP provides simultaneous operation of multiple processes on a serial cable or telephone line. It allows network users the freedom to use TCP/IP based applications over a serial cable or modem connection.

You might consider setting up a SLIP network when cost and distance are large factors in your network planning.

Point to Point Protocol (PPP)

The Point to Point Protocol is similar in nature to SLIP. PPP provides a network connection as if your system were connected to the remote host by a LAN connection. Multiple processes and TCP/IP based applications are supported.

Network Information Service (NIS)

NIS is a network-based information service and an administrative tool. It allows centralized database administration and a distributed lookup service. NIS supports multiple databases based on regular text files. For example, NIS databases can be generated from the *hosts*, *passwd*, *group*, and *aliases* files on the NIS master.

NIS is best suited for a moderate-sized network (one containing approximately 1000 stations, or a small collection of interconnected networks). NIS is part of the NFS optional software and is detailed in the *NFS and NIS Administration Guide*.

Berkeley Internet Name Domain (BIND)

BIND is the Internet's Domain Name System (DNS). BIND is used by various network applications and programs to map station names to internet addresses and vice-versa. If your network interfaces with the Internet, it should run DNS.

BIND is best suited for large networks, or networks connected directly or indirectly to the Internet. BIND provides access to a much larger set of stations than is provided in the */etc/hosts* database. A drawback of BIND is its complicated setup. BIND is described in more detail in Chapter 19, "The BIND Name Server"

Network File System (NFS)

NFS is a network program that can access a remote station's file system and attach it and its data to the local station's file system. On the local station, the remote file system is accessed as if it were a local file system.

NFS should be considered in a network when you want to share files between stations. With NFS, software or data used by a group is put on an NFS server. Authorized NFS clients access the data over the network when needed. This approach ensures consistent information, frees up disk space on client stations, and simplifies the backup procedure. NFS is an optional software product and is described in the *NFS and NIS Administration Guide*.

Planning to Subnet Local Networks

Subnetting allows multiple local networks to appear as a single internetwork to off-site stations. Subnetworks are useful because they allow a site to hide its local topology. The standard describing subnetting of Internet addresses is RFC 950.

Subnetting should be considered when the class limits are unrealistic for your network. For example, a Class B network gives you approximately 64,000 stations per network. This far exceeds the maximum number of stations allowed on most networks. Subnetting allows the local organization to designate some of the host id bits to form a subnet. Subnetting generates a realistic number of stations per network. All changes are made at the local site by the site administration group and are transparent to off-site stations.

Planning is required for subnetting a network (see "Configuring an IRIS for a Network" on page 551 for subnetting procedure). Primarily, you must determine how to partition the host part of the 32-bit Internet address. To define local subnetworks, use bits from the host number sequence to extend

the network portion of the Internet address. This reinterpretation of internet addresses is done only for local networks. It is not visible to off-site stations.

Sites with a Class A network number have 24 bits of host part with which to work; sites with a Class B network number, 16 bits; and sites with a Class C network number, 8 bits. For example, if your site has a Class B network number, each station on the network has an Internet address that contains 16 bits for the network number and 16 bits for the host number. To define 254 local subnetworks, each possessing at most 254 stations, you can use 8 bits from the host portion of the address. Construct new network numbers by concatenating the original 16-bit network number with the extra 8 bits containing the local subnetwork number.

Figure 16-1 shows what happens to the bit assignments in a Class B Internet address that is subnetted.

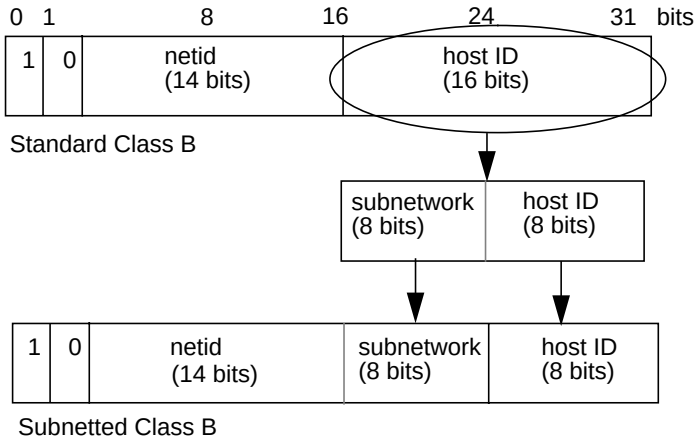


Figure 16-1 Subnetted Class B Address

For example, the Class B Internet address for an entire site as seen from other sites might be 128.50. If subnetting is enabled within the site, the site might be composed of several subnets with network IDs like 128.50.20, 128.50.21, 128.50.22, and so on. A station that resides on the subnet 128.50.21 might have the Internet address 128.50.21.5.

Note: The numbers 0 and 1 are reserved for broadcast addresses. Do not use subnetwork numbers with all 0s or all 1s.

Planning for Network Security

Securing a network is a difficult if not impossible task. If you can discourage potential intruders and quickly isolate or pinpoint successful intruders, you can consider your network secure. Some security approaches are basic UNIX procedures focusing on local and remote stations; for example, the */etc/hosts.equiv* and *.rhosts* files. There are additional security approaches that deal specifically with the physical network (network layout, firewalls) and those that are application based. For additional information on network security and system security in general, see Chapter 12, "System Security."

The subsections that follow will help you answer this list of planning questions:

- Do I understand the purpose of the */etc/hosts.equiv* and *.rhosts* files?
- Do I have any security holes in my physical network?
- Should I consider a network security application?
- Are my *ftp* accounts secure?
- Should I use remote access logging?

The /etc/hosts.equiv File

The authentication scheme used by remote login and shell servers such as *rcp*, *rdist*, and *rsh* is based on the concept of trusted stations. Trusted stations are placed in the file */etc/hosts.equiv*. Users on these stations are allowed to remotely log in or to perform various shell server commands with minimum verification. Minimum verification means that the remote user is equivalent to a local user (password not required and shell servers executed as if locally initiated).

For the */etc/hosts.equiv* file to function properly, the following criteria must be met:

- It must be owned by *root*.
- Permissions must be set to *644*, writable by *root* only.
- The name used in the */etc/hosts.equiv* file must be the remote station's *fully-qualified* name.

- If the remote station is a router, *all fully-qualified* names for *all* interfaces must be included in the */etc/hosts.equiv* file.

Note: The fully-qualified name is the name returned by *gethostbyname*(3). The *ping*(1M) command provides a station's fully-qualified name as part of its output.

The */etc/hosts.equiv* file format consists of the trusted station's name and user's name (optional). Table 16-2 provides some of the common */etc/hosts.equiv* entries. Explanations of the entries are also provided. The remote station is *friend.trust.com* and *safe.trust.com* is the local station.

Table 16-2 Sample Entries for the */etc/hosts.equiv* File

***hosts.equiv* Entry**	**Meaning**
friend.trust.com	Allows any user on *friend.trust.com* to perform as that user on *safe.trust.com*.
friend.trust.com bob	Allows the user *bob* on the station *friend.trust.com* to perform as *any* user (except *root*) on *safe.trust.com*.
+	Allows *any* user (except *root*) from *any* station to perform as that user on *safe.trust.com*.
-friend.trust.com	Denies access to any user from *friend.trust.com*.
friend.trust.com +	Allows any user (except *root*) on *friend.trust.com* to perform as a very trusted user (except *root*) on *safe.trust.com*.
friend.trust.com -bob	Denies access to the user *bob* from the station *friend.trust.com*

Note: If you are using NIS, **netgroup** names in the */etc/hosts.equiv* file are supported. See the *NFS and NIS Administration Guide* as well as the *hosts.equiv*(4) reference page for more details.

The .rhosts File

The *.rhosts* file is an extension of the trusted users provided with the */etc/hosts.equiv* file. The *.rhosts* files are not default files, but are created by the user as needed. Remote login and server shells rely on both the */etc/hosts.equiv* and *.rhosts* (if available) files for reliable authentication. The *.rhosts*

files are used to provide access to remote *root* or regular users without divulging or removing existing passwords. The *.rhosts* file can be created by regular users and *root*.

Entry format for the *.rhosts* file is the same as for the */etc/hosts.equiv* file (see Table 16-2). For the *.rhosts* file to be recognized and to function properly, the following criteria must be met:

- It must be owned by the user.
- Permissions must be set to *644*; writable by *user* only.
- The name used in the *.rhosts* file must be the remote station's *fully-qualified* name.
- If the remote station is a router, *all fully-qualified* names for *all* interfaces must be included in the *.rhosts* file.

Note: The fully-qualified name is the name returned by *gethostbyname*(3). The *ping*(1M) command provides a stations fully-qualified name as part of its output.

Regular Users and the .rhosts File

If the */etc/hosts.equiv* file does not give access to a particular user, a local user can set up a *.rhosts* file in his/her home directory. If the remote user is specified in a local users *.rhosts* file, the user is trusted and allowed access to the local users account. A local user's *.rhosts* file overrides the station's */etc/hosts.equiv* file entries.

root and the .rhosts File

The local */etc/hosts.equiv* file is never checked when authenticating remote root access. The */.rhosts* file is checked to authenticate remote root access. Entries in this file allow users to have local root privileges. You should be very selective about setting up the */.rhosts* file and, in particular, about the stations you trust with root access to your station. Be very careful about putting a "+" in the */.rhosts* file; it gives the *root* user on any remote station *root* privileges on the local station.

Firewalls

A firewall is a network security measure that slows or eliminates network trespassing. Firewalls make it *physically* difficult to jump from one network to another. Firewalls are implemented in two ways:

- Externally
- Internally

External Firewalls

External firewalls put a wall between your local site and the outside world. A solid external firewall includes a *choke* and a *gate*. The choke and the gate may be two separate stations or the same station with multiple interfaces and forwarding turned off.

The choke is the bridge between the inside network and the outside network. It blocks all packets, except those destined for the gate or sent from the gate. You can also screen packets based on protocols, for example, the choke may pass mail related packets, but not *telnet* packets.

The gate is the station that enforces security, authenticates users, and sanitizes data. Typically, the gate stations serves as the mail server, the Usenet server (for news), and the FTP repository. The gate station should meet the following criteria:

- Minimum user accounts; no regular users.
- No NIS or NFS services without *portmap* access controls.
- No unnecessary program or development tools.
- 711 permissions on many system directories.
- Full logging on.
- Process and file quotas on.
- No unnecessary network services.
- No */etc/hosts.equiv* file.

Internal Firewalls

Internal firewalls provide a second layer of protection to network intruders. If a network intruder manipulates their way through your external firewall, the internal firewalls make it difficult for them to move around on your network. This slowdown provides you with more time to track and possibly isolate the intruder. Implementing internal firewalls is simple; separate your site into groups of networks which communicate through routers and/or gateways with high levels of security. Be careful not to create so many local networks that it creates a network bottleneck. The purpose of internal firewalls is to slow down an intruder, not your network performance.

Security Applications

In general, most networks are susceptible to two kinds of security leaks: eavesdropping and imposters. Eavesdroppers are users who record network packets intended for another destination. Imposters actually take the identity of another station and receive information destined for that station. In either case, this susceptibility is primarily due to the promiscuous nature of certain types of media (Ethernet, token ring, and so on.).

Many network applications transmit authentication information (user id and passwords) when requesting service, making eavesdropping and imposter scenarios profitable. Encryption based applications can help minimize security leaks. Encryption scrambles the data making it more difficult to decipher if intercepted by an eavesdropper or imposter. The intruder must know the encryption system used and the key to decipher the data. Kerberos is one such encryption system that is gaining popularity.

Kerberos is an authentication system that uses the Data Encryption Standard (DES) cryptology to pass sensitive information, such as passwords, on an open network. It is an add-on system and can be used with any network protocol. Kerberos was developed by the Massachusetts Institute of Technology (MIT) Athena project. Silicon Graphics Inc. does not supply or support Kerberos, but you can obtain Kerberos source code and papers using anonymous FTP from the computer *ATHENA-DIST.MIT.EDU* or from the following address:

MIT Software Center
W32-300

20 Carlton Street
Cambridge, MA 02139
(617) 253-7686

Anonymous and Restricted FTP Access

The File Transfer Protocol (FTP) allows remote users to transfer files between stations. *ftp* requires passwords and keeps a record of all *ftp* requests in the */var/adm/wtmp* log and in */var/adm/SYSLOG*, if configured correctly. The *ftp* server included in the system provides support for anonymous *ftp*. Because of the inherent security problems with such a facility, read this section carefully if you are considering providing such a service.

The /etc/ftpusers File

For security reasons, the *ftpusers* file should be created and should list the names of any nontrusted *ftp* users. It is checked on each *ftp* connection. If the user's name is located in the file, the request for service is denied. Accounts with nonstandard shells should be listed in this file. Accounts without passwords need not be listed in this file; the *ftp* server does not service these users.

Restricted ftp Accounts

A restricted account has limited access to files on the station. When a client uses the restricted account, the server performs a *chroot*(2) system call to restrict the client from moving outside that part of the file system where the account's home directory is located. Because the server uses a *chroot* call, certain programs and files used by the server process must be placed in the account's home directory. Furthermore, all directories and executable images must be unwritable.

A restricted account must be listed in */etc/ftpusers* with a line containing the account name followed by the word r*estricted*. For example, this command specifies that the "support" account allows restricted *ftp* access:

```
support restricted
```

A client must specify a password to access a restricted account. When the account logs in, the file called *README* in the account's home directory is

printed, if it exists, before the client can execute commands. The *README* file is a good place for station notices and account usage policy.

Anonymous ftp

An "anonymous" account is a special type of restricted account that does not require a password and does not need to be listed in the *ftpusers* file. See the *ftpd*(1M) reference page for complete information. You can enable an anonymous account by creating an entry in */etc/passwd* for the user *ftp:*

```
ftp:*:997:999:FTP anonymous account:/usr/people/ftp:/dev/null
```

Using an asterisk (*) for the password prevents anyone from logging into the account by any other means. See "Setting Up Anonymous ftp" on page 569 for instructions on setting up anonymous *ftp*.

Chapter 17

Setting Up a Network

Chapter 17 covers the actual configuration process of your network. Instructions are provided to set up a variety of network services. Topics include:

- *Configuring an IRIS for network connectivity.*
- *Setting up routers.*
- *Creating Subnets.*
- *Setting up anonymous ftp services.*
- *Modifying the network interface configuration.*
- *Creating a custom network script.*

Chapter 17

Setting Up a Network

This chapter contains procedures for:

- Configuring an IRIS for network connectivity. See "Configuring an IRIS for a Network" on page 551.
- Connecting to an Ethernet Network and checking your Ethernet connection. See "Attaching Your Station to an Ethernet Network" on page 552.
- Setting up routers. See "Setting Up a Router" on page 560.
- Setting up an anonymous *ftp* account. See "Setting Up Anonymous ftp" on page 569.
- Setting up a system as the InSight file server. See "Setting Up an InSight File Server" on page 571.
- Modifying the network interface configuration. See "Modifying the Network Interface Configuration" on page 575.
- Changing network parameters. See "Changing Network Parameters" on page 578.
- Creating a custom network script. See "Creating a Local Network Script" on page 580.
- Setting up remote logging. See "Turning On Remote Access Logging" on page 581.

Configuring an IRIS for a Network

The procedure in this section explains how to configure an IRIS station, with one interface, for an Ethernet network using its local */etc/hosts* file (without BIND or NIS). Configuring the station takes six steps:

1. bringing the station down
2. attaching the station to the network
3. checking the station's network configuration
4. modifying the */etc/hosts* database
5. naming the station
6. testing the connection

Each of these steps is explained in the sections that follow.

Attaching Your Station to an Ethernet Network

Attach your station to the network by connecting one end of the Attachment Unit Interface (AUI) cable to the Ethernet I/O port and the other end to the network transceiver or Medium Access Unit (MAU).

If your transceiver model has status lights, make sure that the power light on the transceiver comes on when you attach the AUI cable to your workstation and the transceiver. This light indicates that the Ethernet card or the integral Ethernet controller is alive. Your station must be powered on to activate the power light on the transceiver.You may also see another light which indicates that your link to the network is activated.

If the power light on the transceiver is lit or if your transceiver or MAU has no lights, bring your station back up into multi-user mode now.

Checking Your Ethernet Connection

To make sure your ethernet connection is functional, perform the steps outlined in this section. You will use *ping -r* to check the connection. This command ensures that you can connect with at least one other system on the ethernet network. Perform the following steps:

1. Obtain the hostname of at least one reliable station on the local area network to which your system is connected. If possible, get the fully qualified hostname and the IP address. (For example, a hostname might be *hancock*, and the fully qualified hostname might be *hancock.corp.gen.com*, while the IP address might be *192.70.0.356*.) It is important that the station you select have a reliable ethernet connection and that it is up and running.
2. Once you have obtained a hostname and IP address, give the command:

   `ping -r` *hostname*

   You should see a series of records indicating the returned packets from the remote host. For example (using our example system):

   ```
   PING hancock (192.70.0.356): 56 data bytes

   64 bytes from 192.70.0.356: icmp_seq=0 ttl=255 time=2 ms
   64 bytes from 192.70.0.356: icmp_seq=1 ttl=255 time=2 ms
   64 bytes from 192.70.0.356: icmp_seq=2 ttl=255 time=2 ms
   64 bytes from 192.70.0.356: icmp_seq=3 ttl=255 time=2 ms
   64 bytes from 192.70.0.356: icmp_seq=4 ttl=255 time=2 ms
   ```

3. Press **`<Ctrl>-C`** or your **`Delete`** key to stop the *ping* command. You will see the tallied results of the *ping* command. For example:

   ```
   ----hancock PING Statistics----

   5 packets transmitted, 5 packets received, 0% packet loss

   round-trip min/avg/max = 2/2/2 ms
   ```

4. If your network connection is working, you should see results comparable to those above. Your *ping* results should show 0% packet loss and an equal number of packets transmitted and received. If some packets are being lost, the first thing you should check is the tightness and quality of the cable connections. Loose cables are frequently the cause of lost packets. The round-trip time factors are a function of the size and general load of your network, and not necessarily an indication of a problem with your ethernet connection.

If your ping command is not successful, there are several things you should check. Perform these steps:

1. Try to ping the station by its IP address. For example, using our sample host *hancock*, use the command:

   ```
   ping -r 192.70.0.356
   ```

2. Try to ping a different station on your local network.

3. Check the network configuration on your system with the *netstat -in* command. You should see information similar to this:

   ```
   Name Mtu  Network  Address    Ipkts Ierrs Opkts Oerrs
   ec0  1500 192.70.0 192.70.0.9 18    0     18    0
   ```

   The `ec0` entry indicates your primary ethernet connection. The `Ipkts` and `Opkts` fields indicate the number of inbound and outbound packets the network interface has processed. The `Ierrs` and `Oerrs` fields indicate the number of errors in input and output, respectively.

   For the purposes of this troubleshooting session, though, check that the portion of the IP address shown under the `Network` heading match the IP address of the hostname that you attempted to *ping*. If the network addresses are not the same, the station is on a different network and the *ping* likely failed for that reason. Find a system to *ping* that is on your immediate local network.

4. Check the */var/adm/SYSLOG* file for ethernet error messages.

5. Check to ensure that other stations are operating normally on the local network.

6. Check to ensure that the correct software (*eoe2.sw.tcp*) package has been installed on your system.

7. Check the physical connections to the ethernet cables and transceivers for tightness and connection. A good indicator is the status light display on your transceiver (if your transceiver has these lights.). If your ethernet hardware is loose or disconnected, the */var/adm/SYSLOG* file and your system console should both show messages such as:

   ```
   ec0: no carrier: check Ethernet cable
   ```

8. If all connections are tight and you still receive errors, replace the pieces of the ethernet connection outside your system. (The cable and transceiver.) Different transceivers require different wire connections, and sometimes the wrong cables are used.

9. If you receive a message indicating that another host has the same IP address, find out which host has the same address and determine which host has set their address incorrectly. (It is more likely that the same address was accidentally assigned to a second system, or that the new system being tested incorrectly set the address.)

Troubleshooting Your Ethernet Connection

This section addresses some of the common ethernet related problems.

Cable Problems

Unplugged or loose cables are the most common problem that causes Ethernet related error messages. For example

```
unix: <ethernet_device>: no carrier: check Ethernet cable
```

This message means that carrier was not detected when attempting to transmit. Some recommendations are:

- Check to see if the Ethernet cable is unplugged from the back of the machine. For detailed instructions on connecting the cable, see your owner's guide. You do not need to *shut down* the system to connect or disconnect the cable. If you re-connected the cable, test the network connection.

After determining that the transceiver cable is plugged into the back of the machine and also into the transceiver box, look for other possible reasons for failure.

- Check the machine's transceiver.
- Check the transceiver cable, and try swapping it out with one that is working on another machine.
- Check for a problem with a 10baseT hub.
- If you try all the troubleshooting techniques and you still cannot access the network, contact your service provider; the network itself may be temporarily out of order.

For more information on this error, see the reference page for *ethernet(7)*. See the "Troubleshooting" chapter in the *Personal System Administration Guide* for detailed instructions on troubleshooting network problems.

Packet Size

An error message regarding the packet size is explained here.

```
unix: <ethernet_device>: packet too small (length = <packet
size>)
```

The ethernet controller received a packet that was smaller than the minimum Ethernet packet size of 60 bytes. This problem is caused by another machine with a bad Ethernet controller or transceiver. Some recommendations are:

- Check other machines on the network for bad Ethernet controllers.
- Check other machines on the network for bad transceivers.
- Check to see if removing a certain machine from the network makes the problem change or disappear.

For more information on this error, see the reference page for *ethernet(7).*

Unable to Contact Server System

When your system cannot contact a network system, an error message similar to this is displayed:

```
portmapper not responding: giving up
```

This problem occurs in one of these situations:

- The system is not running.

  Physically go to the system or call the system's Primary User or Administrator to check to see if the system is powered up and running.
- The network is not running.

  To check, try to access another system on the network:
- The network administrator changed some information about the system or about the system's logical location on the network.

Check whether this is the case, and get the appropriate information to fix the problem.

- The system has too many users or systems using its resources. It cannot provide services to you at this time.

  Contact the system's Primary User or Administrator to check on this.

Checking Additional Network Interfaces

If your system has more than one network interface (additional interfaces are usually fiber-optic (FDDI) links or SLIP connections or other ethernet boards) you can easily perform the above checks on each network interface.

To check your other network interfaces, give the *netstat -in* command. You see output similar to the following:

```
Name Mtu   Network    Address    Ipkts Ierrs Opkts Oerrs Coll
ec0  1500  192.70.0   192.70.0.9  15    0     15    0     24
ec1  1500  192.70.2   192.70.2.5  15    0     15    0     24
sl0  1006  (pt-to-pt) 192.70.0.9  0     0     0     0     0
lo0  8304  loopback   localhost   8101  0     8101  0     0
```

The second ethernet connection is to the network 192.70.2, a different LAN from the first ethernet connection. The address of the local station on the second LAN is 192.70.2.5. To check that connection, use the *ping* command to test the connection to another station on that network.

There is also a SLIP link running in this example. The SLIP link extends the same LAN as `ec0` to another system in a different location. To test this link, find the hostname or IP address of the station at the other end of the SLIP link and use the *ping* command to test connectivity.

The `lo0` interface is the loopback network interface on the local host.

Checking the Network Software Configuration

When the station comes up, verify the station's software configuration. Log into the station as *root*.

Issue the *chkconfig*(1M) command to check that your station's standard networking software is correctly configured:

```
/etc/chkconfig
```

You see information similar to this:

```
named      off
network    on
rwhod      off
timed      on
timeslave  off
gated      off
mrouted    off
rtnetd     off
snmpd      on
routed     on
```

Note: Your output will vary from the output above depending upon installed software and configuration flag settings.

If you are familiar with the network-related daemons, you can customize your configuration flags to suit your network needs. If you are not familiar with the network-related daemons, set your network-related flags as shown above. In particular, make sure that the *network* variable is configured *on*.

Modifying the hosts database

Before you modify the hosts database, you should have a list of station names and valid Internet addresses of all stations in your network. If the network has routers (stations with multiple network interfaces), there must be a valid Internet address and name for each interface. See "Internet Addresses" on page 531 for addressing information.

The *hosts* file is the host name database. It contains information regarding known stations, from the perspective of the local station. This example assumes that you are not using NIS or BIND. If you are using NIS, refer to the *NFS* and *NIS* administration guides for more information. If you are using BIND, refer to Chapter 19, "The BIND Name Server," for more information.

Edit the */etc/hosts* file and add the host names and Internet addresses for all stations on your network, including this station. Each station on the network must have all station names in the local */etc/hosts* file. You can use the *rcp* or *rdist* programs by means of *cron* to ensure that the *hosts* files stay in sync.

Caution: Do not remove or modify in any way the *loopback host address* 127.0.0.1. This is a special Internet address that must be present for testing and loopback facilities.

Modify the */etc/hosts* file. For this example, the local station is *setup1* and the network includes three other stations, *setup2*, *setup3*, and *setup4*. The network is a Class C network, 192.60.31.

```
vi /etc/hosts
```

```
127.0.0.1       localhost loghost
192.60.31.1     setup1
192.60.31.2     setup2
192.60.31.3     setup3
192.60.31.4     setup4
```

Save and exit the */etc/hosts* file.

Naming Your Station

Once you have determined your station's name, edit the */etc/sys_id* file to give your station its new identity.

1. Remove the default station name (IRIS) and replace it with the new station name (*setup1* for this example). You can use this command:

   ```
   echo setup1 > /etc/sys_id
   ```

2. For the change to take affect, reboot your station with this command:

   ```
   reboot
   ```

When your station comes back up, it should have the new station name as the login prompt.

Testing Your Network Connectivity

Two network management tools, *rup* and *ping*, provide quick information about network connectivity. *rup* indicates if there is a physical problem with the network, such as your station being unable to contact the other stations. Since *rup* uses broadcasts as a default, it does not go through routers. If your station can see the other stations on the network, use *ping* to test communication abilities. *ping* uses the Internet Control Message Protocol (ICMP), which requests an echo from the designated station. It lets you know if you can transmit and receive packets to and from specific stations.

1. Issue the *rup* command to determine if your station can contact the other stations on the network:

   ```
   /usr/bin/rup
   ```

   You should get output on each of the stations on your network. The other stations on your network must be up for your station to get a user-friendly response. If the other stations are powered on and attached to the network but not up in user mode, the information comes back in hexadecimal.

2. Issue the *ping* command to see if your station can communicate with the other stations on the network:

   `/usr/etc/ping` *station_name*

   Let the output run a few seconds, then use **`<Ctrl-C>`** to break it. Pay particular attention to the ping statistics. *ping* gives you the number of packets transmitted, number of packets received, percentage packet loss, and round trip time (min/avg/max). These are all good indicators as to the general condition of your network. Obviously, you want 0% packet loss and a fast round trip time.

Setting Up a Router

A router is a station with multiple network connections that forwards packets between networks. This section provides the configuration procedure for a router with two and a router with more than two interfaces. A station can have multiple interfaces and not act as a router. The procedure for turning forwarding off on a station with multiple interfaces is also provided in this section.

Configuring a Router with Two Interfaces

The */etc/init.d/network* script is designed to automatically detect and configure a router with two interfaces *if* the default naming scheme for the interfaces is used. By default, the Internet addresses of the primary and secondary interfaces are derived from the */etc/sys_id* file. The primary interface uses the name in the *sys_id* file. The secondary interface prefixes *gate-* to the name specified in the *sys_id* file.

To set up a router with two interfaces using the default naming scheme, follow this procedure:

1. Log in as **root**.
2. Assign valid Internet names and addresses to both interfaces in the */etc/hosts* file. For example, the */etc/hosts* file entries for the primary and secondary interfaces on the station *biway* might look like this:

   ```
   192.26.75.2     biway
   192.26.80.3     gate-biway
   ```

3. Ensure that the router has the appropriate name in its */etc/sys_id* file. Following this example, the */etc/sys_id* file should look like this:

   ```
   biway
   ```

4. Reconfigure the kernel and reboot the station to initialize your changes and interfaces. Some systems will prompt you for permission, as in the following example. Others will simply return a shell prompt. In either case, you must explicitly give the command to reboot the system.

   ```
   /etc/autoconfig
   Automatically reconfigure the operating system? (y/n)y
   reboot
   ```

Note: If you do not want to use the standard router naming scheme, you must modify the */etc/config/netif.options* file. “Modifying the Network Interface Configuration” on page 575 details the procedure for changing an interface name.

Configuring a Router with More Than Two Interfaces

If the router contains more than two interfaces, you must modify the */etc/config/netif.options* file in addition to the */etc/hosts* and */etc/sys_id* files. In the *netif.options* file, you must define the interface type (enp1, ipg0, and so on.). By default, the names for the third and fourth interfaces are *gate2-$HOSTNAME* and *gate3-$HOSTNAME*, where $HOSTNAME is the value returned when you issue the *hostname* command. If you want to modify the interface names, see "Modifying the Network Interface Configuration" on page 575 for the detailed procedure.

To set up a router with more than two interfaces and using the default naming scheme, follow this procedure:

1. Log in as **root**.
2. Assign valid Internet names and addresses to all interfaces in the */etc/hosts* file. For example, the */etc/hosts* file entries for the router *freeway*, with four interfaces, would look like this:

   ```
   192.26.30.1     freeway
   192.26.32.4     gate-freeway
   192.26.41.5     gate2-freeway
   192.26.59.6     gate3-freeway
   ```

3. Ensure that the router has the appropriate name in its */etc/sys_id* file. Following this example, the */etc/sys_id* file should look like this:

   ```
   freeway
   ```

4. Modify the *netif.options* file to define your interface types. For this example, the third and fourth interfaces are FDDI (ipg*). Change the *if3name* and *if4name* variables from:

   ```
   if3name=
   if4name=
   ```

   to

   ```
   if3name=ipg0
   if4name=ipg1
   ```

   Save your changes to the *netif.options* file.

5. Reconfigure the kernel and reboot the station to initialize your changes and interfaces. Some systems will prompt you for permission, as in the following example. Others will simply return a shell prompt. In either case, you must explicitly give the command to reboot the system.

```
/etc/autoconfig
Automatically reconfigure the operating system? (y/n)y
reboot
```

Turning Forwarding Off

By default, when a station has two or more network interfaces, it automatically forwards (routes) packets between the two interfaces. If you do not want the station to forward (route) packets between the networks, you must disable supplied routing information and IP forwarding.

1. Modify the */etc/config/routed.options* file so the router will not supply routing information about general network routes (**-q**) or local interface routes (**-h**). Type:

```
echo -qh > /etc/config/routed.options
```

2. Using the *chkconfig*(1M) command, turn the network off momentarily, then start it again.
3. When the network comes up, verify that it is not forwarding packets with the *netstat* tool. Type:

```
netstat -s -p ip |grep forward
```

To change forwarding dynamics, create or edit the */etc/config/routed.options* file to contain the desired options. Some of the behaviors that can be altered by means of the */etc/config/routed.options* file are listed below:

- Suppress or force publicizing of routing information
- Enable tracing of all received packets
- Filter packets destined for specific networks

See the *routed*(1M) online reference page for more details.

Turning On Multicast Routing

If you're using Silicon Graphics workstations as routers, you can turn on multicast routing by doing the following:

1. Identify the routers on each network that need to support multicasting. Make sure that the routers you select are running IRIX Version 5.2 or later.
1. If you haven't already done so, install on each router the *eoe2.sw.ipgate* subsystem from your IRIX Version 5.2 distribution source. Run *autoconfig(1M)* if necessary.
2. As **root**, enter the command:

   ```
   chkconfig mrouted on
   ```

3. Restart the system with the *reboot* command.

Understanding Where Multicast Packets are Forwarded

Figure 17-1 shows an example network with three routers. Each bubble labeled *Host* represents a system on the network.

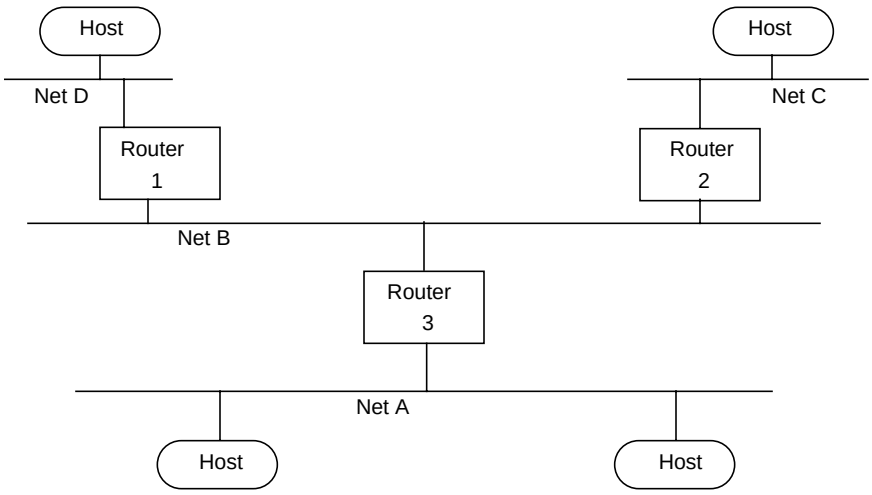


Figure 17-1 A network with multicast routers.

- If people on networks A, C, and D are listening for packets, as shown in the figure, all four networks receive the multicast packets.
- If people on networks A and C are listening for packets, networks A, B, and C receive the multicast packets.
- If three or more people on network A are listening for packets, networks A and B receive the multicast packets. The multicast routing protocol prevents packets from being sent to "leafs" of the network, such as networks C and D; however, it doesn't prevent packets from being sent to interior networks, such as network B.

Setting Up Tunnels to Support Multicast Packets

If your routers do not support multicast routing, you can support multicast packets by creating "tunnels" between the networks. Any Silicon Graphics workstation running IRIX Version 5.2 or later can be configured as the endpoint of a *tunnel*. With tunneling, the multicast packets are encapsulated inside unicast packets, and then are sent to the other end of the tunnel. They are converted back into multicast packets when they are received.

Figure 17-2 shows an example of a setup with tunnels.

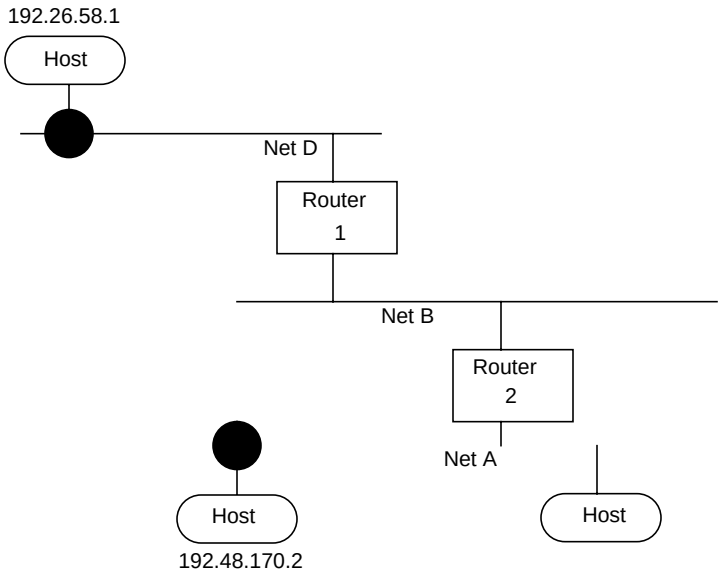


Figure 17-2 Diagram showing tunnels between networks.

To create the tunnel, you edit the file */etc/mrouted.conf*. Step-by-step instructions follow:

1. Select the systems on each network that you will use for the sending and receiving end of the tunnel.

   Choose a system that's running IRIX Version 5.2 or later, is fast, and is not used extensively. The audio and video data may be intermittent if the system you select is slow or overloaded.

2. If you haven't already done so, install the *eoe2.sw.ipgate* subsystem from your IRIX distribution source.
3. As *root*, edit the file */etc/mrouted.conf* on the sending and receiving end of the tunnel. Note that these endpoints can be separated by many routers; you can use any machine on the network that is running IRIX Version 5.2 or later.

   Add the following line for each network to which you want to establish a tunnel.

   tunnel <*local IP address*> <*remote IP address*>

   In the above example, the system on network D would have the following entry:

   ```
   tunnel 192.26.58.1 192.48.170.2
   ```

   The system on network A would have the following entry:

   ```
   tunnel 192.48.170.2 192.26.58.1
   ```

4. You can specify other optional settings for the tunnel. For details, see the *mrouted*(1M) man page.
5. Restart the system.

Note: One copy of the multicast packets is sent for each tunnel entry in *mrouted.conf*. This results in extra network traffic. For example, suppose you have a workstation with one network interface and you set up tunnels to workstations on three other networks. The packets are replicated three times.

Update */etc/rpc* for NIS Users

The IRIX Version 5.2 copy of the */etc/rpc* file contains additions that are essential if you're running the Network Information Service (NIS). If the NIS master is not running IRIX Version 5.2 or a later release, or is not a Silicon Graphics workstation, verify that the */etc/rpc* file on the NIS master includes these additions:

```
sgi_iphone 391010
sgi_videod 391011
```

Subnetting a Network

Implementing the subnet address scheme is a very simple procedure. The following must be done to implement subnetting:

1. Setting the netmask
2. Rebooting the station

Note: If you have not done so already, read Chapter 16, "Planning a Network." It contains information about partitioning Internet addresses for subnetting.

Setting the Netmask

The **netmask** option of the *ifconfig* command is used to interpret and define the network portion (subnets included) of the Internet address. A *network mask* defines the portion of the Internet address that is to be considered the network part. This mask normally contains the bits corresponding to the standard network part as well as the portion of the host part that has been assigned to subnetworks. If no network mask is specified when the address is set, it will be set according to the class of the network.

To configure a station's primary interface to recognize a subnetted Class B address that is using 8 bits of the host ID for defining the subnets, create or modify the */etc/config/ifconfig-1.options* file and insert the following line:

```
netmask 0xffffff00
```

This netmask value indicates that for the primary interface, the upper 24 bits of the Internet address represent the network portion of the address, and the remaining 8 bits represent the host ID. The nonsubnetted netmask for a Class B address (in hexadecimal) would be 0xffff0000. The netmask value can be set using hexadecimal, dot-notation Internet address, or pseudo-network name formats. This entry must be made on all stations on the subnet.

Note: For stations with multiple interfaces, the network mask should be set for each interface in the appropriate *ifconfig-*.options* file.

Rebooting the Station

When the **netmask** value is set, the station must be reconfigured and rebooted to incorporate the new network address into the stations routing tables. Always reboot routers before regular stations, since they provide routing information to other stations and networks.

Reconfigure the kernel and reboot the station to initialize your changes and interfaces. Some systems will prompt for permission before reconfiguring the kernel, while others will simply return a shell prompt. In either case, you must still give the *reboot* command explicitly.

```
/etc/autoconfig
Automatically reconfigure the operating system? (y/n)y
reboot
```

Setting Up Anonymous ftp

This section summarizes how to set up an anonymous *ftp* account. If you have not done so already, read "Planning for Network Security" on page 541. It contains security information pertaining to *ftp*.

Follow these procedures to set up an anonymous *ftp* account:

1. Create the anonymous *ftp* password entry. You may choose the location of the home directory. This example uses */usr/local/ftp* as the home directory. The example entry would look like this:

   ```
   ftp:*:997:999:Anon FTP Account:/usr/local/ftp:/dev/null
   ```

2. Make a home directory for the anonymous *ftp* account:

   ```
   mkdir -p /usr/local/ftp
   ```

3. Set permissions, owner, and group for *ftp* directory with the following command:

   ```
   cd /usr/local; chmod 555 ftp; chown ftp.other ftp
   ```

4. Create the mini file system with the following commands:

   ```
   cd /usr/local/ftp
   mkdir bin etc pub lib dev
   ```

5. Set permissions, owner, and group for *bin*, *lib*, *dev,* and *etc* directories with the following commands.

```
chown root.sys bin etc lib dev
chmod 555 bin etc lib dev
```

6. Set permissions, owner, and group for *pub* directory with the following commands.

```
chown ftp.other pub
chmod 777 pub
```

7. Copy the *ls* command from */bin* to *ftp*'s *bin* directory and make it executable only with the following commands.

```
cp /bin/ls bin
chmod 111 bin/ls
```

8. Copy the password and group files from */etc* to *ftp*'s *etc* directory and set restrictive permissions with the following commands (note the lack of a leading slash (/) on the second argument of the copy commands):

```
cp /etc/passwd etc/passwd
cp /etc/group etc/group
chmod 444 etc/*
```

9. Copy some library files to *ftp*'s *lib* directory and change their permissions:

```
cd /usr/public/ftp/lib
cp /lib/rld .
cp /lib/libc.so.1 .
chmod 555 *
```

   Please check to make sure that the *guest* account is listed in the */etc/passwd* file.

   To place files in the restricted/anonymous area, local users must place the files in the */usr/local/ftp/pub* subdirectory.

   Caution: An important issue to consider is the *passwd* file in *ftp*'s */etc* directory. This file can be copied by users who use the account. They may then try to break the passwords of users on your machine for further access. Therefore, remove any unnecessary users and change all passwords in this file to an invalid password such as "*". A good choice of users to include in this copy of the */etc/passwd* file might be *root*, *bin*, *sys*, and *ftp*.

10. Copy the run-time libraries from */lib* to the *ftp/lib* directory and create a special *zero* file in the *ftp/dev* directory with the following commands:

```
cp /lib/rld lib
cp /lib/libc.so.1 lib
chmod -R 555 lib dev
/etc/mknod dev/zero c 37 0
chmod 444 dev/zero
```

Setting Up an InSight File Server

The files and directories that make up the InSight system of on-line documentation take up a great deal of space. In your network, there is no reason for each system to maintain a separate copy of the InSight documents as long as all systems are at substantially the same software revision level. If the InSight software revision level is the same across several systems, you can designate one system to serve the others with the InSight files, thus freeing up disk space on the client systems.

You should be sure to choose a server system that is not called upon to carry a heavy workload, and you should also take your network performance into account before you begin. If your users use InSight a great deal and your network is already burdened, you may find that your users will not appreciate the decreased response time from both InSight and your network in general. Also, if a person is to use the InSight server as his or her workstation, that person must be prepared to accept a certain (possibly substantial) amount of disk, CPU, and network overhead as a result.

There are two convenient ways to set up an InSight server. These will be detailed in following subsections. Both methods require that you have the NFS software option installed. If you do not understand the terms and concepts behind NFS, you should read the *NFS Administration Guide* and the *NIS Administration Guide* before undertaking these projects. The second method described here also requires a dedicated CD-ROM drive to hold the InSight distribution media.

A Conventional InSight Server/Client System

To install the IRIS InSight Viewer and Document Library on a remote server and retrieve the information from a local client system, follow these steps:

On the server system:

1. Log in as **root** (The Superuser).
2. Bring up *inst*(1M) and install the complete IRIS InSight product image (from your CD-ROM drive or distribution directory) with the commands:

   ```
   Inst> install insight insight_gloss *.books.*
   Inst> go
   ```

   The total size of the viewer and document library is a little less than 23 Megabytes.
3. Export the */usr/share/Insight/library/SGI_bookshelves* directory using the System Manager or the *exportfs*(1M) command:

   ```
   exportfs -i /usr/share/Insight/library/SGI_bookshelves
   ```

   If the server does not have a graphical display, a warning is generated during the exit. This warning relates to updating the X server's font directory and can be ignored.

On the client system:

1. Log in as *root* (The Superuser).
2. Give the command:

   ```
   versions remove *.books.*
   ```

   to remove the books on your bookshelves.
3. Bring up *inst*(1M) on your client system and give the following commands to install the *insight.sw.client* subsystem (you may want to install the *insight.man.man* and the *insight.man.relnotes* subsystems as well).

   ```
   Inst> keep insight insight_gloss
   Inst> install insight.sw.client
   Inst> go
   ```

4. Mount the *SGI_bookshelves* directory from your server on your client system. In the example below, the name of the server machine is *capra*.

   ```
   mkdir /usr/share/Insight/library/SGI_bookshelves
   ```

   ```
   mount capra:/usr/share/Insight/library/SGI_bookshelves /
   usr/share/Insight/library/SGI_bookshelves
   ```

   Note that when you enter the *mount* command on your client system, the entire command line goes on a single line. The command is broken across two lines in this example due to formatting limitations.

Note: If you have remote-mounted your InSight books, the SGI *desktophelp* system will not operate correctly while the books are mounted. To view the *desktophelp*, unmount the books temporarily.

A CD-ROM InSight Server/Client System

With this method, you need not use up your disk space for the InSight files if you have a CD-ROM drive you can dedicate to InSight. You simply leave the CD with the InSight distribution in the drive and link the mounted distribution to the directory where InSight would have installed the files. The drawback of this system is that you must dedicate a CD-ROM drive to the purpose, but this method can be used with NFS to provide server access to your entire network.

Note: If you have a version of the IRIS InSight Document Library installed on your client disk but you want to mount the books from the CD-ROM, you must remove all the files in */usr/share/Insight/library* before creating the symbolic link described in step 2. Use the *versions remove* command to cleanly remove the books, then check the directory to be sure all files have been removed.

If you wish to use the IRIS InSight Viewer and Document Library from the CD-ROM and access the information from a local or remote system, follow these steps:

On the server system with the CD-ROM drive:

1. Log in as **root** (The Superuser).
2. Insert the IRIS InSight CD into the CD-ROM drive. If the drive is not mounted, use the System Manager or the *mount*(1M) command to mount the drive. The most common mount point is */CDROM*. If the drive is mounted correctly, you should be able to change directories to */CDROM* and see all the files in the IRIS InSight CD. To see the files use the command:

   ```
   cd /CDROM/insight
   ```

3. As *root*, create a symbolic link from */usr/Insight/library/SGI_bookshelves* to the CD *insight* directory. You may need to create the directory */usr/Insight/library* if it doesn't exist. Use the commands:

   ```
   mkdir -p /usr/share/Insight/library
   ```

   ```
   ln -s /CDROM/insight/SGI_bookshelves /usr/share/Insight/
   library
   ```

   Note that when you enter the link command on your client system, the entire command line goes on a single line. The command is broken across two lines in this example due to formatting limitations.

   At this point, the InSight bookshelves are mounted on your server and available for use on that system. To allow users on other systems on your network to use the bookshelves, proceed to step 4.

4. If you want to share the bookshelves with other users on your network, export the */CDROM* directory using the System Manager or the *exportfs*(1M) command:

   ```
   exportfs -i /CDROM
   ```

On each client system, perform the following steps:

1. Log in as **root** (The Superuser).
2. Bring up *inst*(1M) and give the following commands to install the *insight.sw.client* subsystem (you may want to install the *insight.man.man* and the *insight.man.relnotes* subsystems as well). You can get the installation software from the CD in the */CDROM/dist* directory on the server.

   ```
   Inst> keep *
   Inst> install insight.sw.client
   Inst> go
   ```

3. Mount the bookshelves from the server. In the example below, the name of the server machine is *capra*. Use the following command.

   ```
   mount capra:/CDROM/insight/SGI_bookshelves /usr/share/
   Insight/library/SGI_bookshelves
   ```

   Note that when you enter the *mount* command on your client system, the entire command line goes on a single line. The command is broken across two lines in this example due to formatting limitations.

Using Remote InSight

After the software has been installed, you should force your shell to remake its list of available programs and commands with the command:

```
rehash
```

You can now invoke the IRIS InSight Viewer. The command to invoke the viewer is *iiv*. The name *iiv* is an acronym for IRIS InSight Viewer. The Viewer process does not automatically put itself in the background because it sends some error messages to the console. You can place it in the background by typing the command with the backgrounding operator (&):

```
iiv &
```

Modifying the Network Interface Configuration

You do not always need to modify (configure) a station's network interface; in most situations the default configuration suits the site's needs. Modifying the network interface configuration requires that you modify the */etc/config/netif.options* file. You modify this file if:

- The station has more than two interfaces.
- You don't like or use the default naming conventions.
- The default order is incorrect.

There are two configurable variables in the *netif.options* file: **interface name** and **interface address**. The interface name variable designates the order (primary, secondary, third, or fourth) and type of interface involved. The interface address variable assigns a valid Internet address to each interface.

There must be a valid Internet address for each interface in the */etc/hosts* file. Table 17-1 summarizes the interface name and interface address variables.

Table 17-1 Variables for the *netif.options* File

Variable Name	Variable	Examples
Interface Name	if*x*name=	if1name=enp0
	where:	if2name=ipg0
	x = 1, 2, 3, or 4	if3name=enp1
	name=ec0, et0, enp0, enp1, fxp1, ipg0, ipg1, etc.	if4name=enp2
Interface Address	if*x*address=	if1address=$HOSTNAME
	where:	if2address=fddi-$HOSTNAME
	x = 1, 2, 3, or 4	if3address=gate-goofy
	address=$HOSTNAME, station name, or Internet address	if4address=192.30.28.2

You can modify either or both variables. Instructions for modifying both variables are provided below.

Modifying the Interface Name

When a station has more than two network interfaces, you must modify the name entries in the */etc/config/netif.options* file to assign the interface order. Default order is only assigned to the first two interfaces. Additionally, if you want to change the order (primary, secondary, etc.) or interface type assigned to a network interface, you must modify the */etc/config/netif.options* file. This example makes the first FDDI interface secondary.

1. Using the *netstat* command, verify the network interface's name:

   ```
   /usr/etc/netstat -ina
   ```

2. Using vi or any editor, open the *netif.options* file for editing:

   ```
   vi /etc/config/netif.options
   ```

3. Locate and modify the appropriate interface name variable. For this example, search for the secondary interface name variable (*if2name*) and change it from the default configuration to a configuration that supports the first FDDI interface as secondary:

   Change from this:

   ```
   : if2name =
   ```

   to this:

   ```
   if2name=ipg0
   ```

   Caution: Note that all *default* variables (primary and secondary) start with a leading colon (:). You must remove the leading colon(:) *and* enter the interface type to change the default interface name.

4. Save and exit the file.

If you have no other changes, autoconfigure and reboot the station. Otherwise, repeat this procedure for each interface name change.

Note: When you alter the order of one network interface, the other interfaces in the station remain in their default relative ordering. For example, on a three interface station (a=primary, b=secondary, and c=third), if you were to make the default secondary the primary (b=primary), the remaining interfaces would be configured a=secondary and c=third.

Modifying the Interface Address

To change the default interface address, you must modify the */etc/config/netif.options* file. All interface names require valid Internet addresses as found in the */etc/hosts* file. The $HOSTNAME variable pulls the station name from the */etc/sys_id* file. This example changes the secondary, third, and fourth interface addresses as follows:

- secondary address: fddi-$HOSTNAME
- third interface address: gate-goofy
- fourth interface address: 192.30.28.26
- Follow these instructions to modify your network interface address according to the above example:

1. Verify or assign a valid entry for each interface in the */etc/hosts* file. Note down the name and address for each interface.
2. Using *vi* or any editor, open the *netif.options* file for editing:

   ```
   vi /etc/config/netif.options
   ```

3. Locate and modify the appropriate interface address variable. For this example, we want to modify the secondary, third, and fourth interface address variables. Find and modify each variable as follows:

   Change from this:

   ```
   : if2addr=gate-$HOSTNAME
   if3addr=gate2-$HOSTNAME
   if4addr=gate3-$HOSTNAME
   ```

   to this:

   ```
   if2addr=fddi-$HOSTNAME
   if3addr=gate-goofy
   if4addr=192.30.28.26
   ```

   Caution: Note that all *default* variables (primary and secondary) start with a leading colon (:). You must remove the leading colon(:) *and* enter the interface address to change the default interface address.

4. Save and exit the file.

Repeat this procedure for each interface address change. If you have no other changes, reconfigure and reboot the station.

Changing Network Parameters

Network parameters are those settings that determine how an interface processes or supplies certain network information. Modifying network parameters requires that you create and modify the appropriate */etc/config/ifconfig-#.options* file (where "#" can be 1, 2, 3, or 4, based on the interface name).

The default parameter settings are known to function well and suit most site needs. Changing the settings can cause a network to become dysfunctional. It is recommended that these parameters be changed only by experienced or trained Network Managers.

Modifying the ifconfig-#.options File

There are four nondefault network parameters that can be set in the *ifconfig-#.options* file:

- Netmask
- Broadcast Address
- ARP
- Route Metric

These steps show how to set the nondefault network parameters:

1. Using the *netstat* command, determine the order assigned to the network interface at configuration time:

   ```
   /usr/etc/netstat -i
   ```

2. Using vi or any editor, open or create the file */etc/config/ifconfig-#.options*, where the pound sign (#) represents the network interface's name. For example, to configure a primary interface, open or create the file */etc/config/ifconfig-1.options*.

3. To change the network mask value for a network interface, enter a line with the word `netmask` followed by a space and either the 32-bit hexadecimal value, Internet address dot-notation, or the pseudo-network name.

   ```
   netmask 0xffffff00
   ```

   See "Turning On Multicast Routing" on page 564 for more details regarding netmasks.

4. To change the broadcast address for a network interface, enter a line with the word `broadcast` followed by a space and the dotted decimal IP broadcast address.

   ```
   broadcast 189.92.6.0
   ```

   To enable or disable the Address Resolution Protocol (ARP), enter a line with *arp* (to enable) or *-arp* (to disable).

   ```
   arp
   ```

   The ARP tables are used by some of the network management tools (*netstat*, *ifconfig*, etc.) and provide the administrator with invaluable network information.

5. To change the routing metric count for a network interface, enter a line with the word `metric` followed by a space and the count:

   ```
   metric 7
   ```

   The default metric count, also called *hops*, is 0. The *routed* daemon monitors the number of hops required to route data from one network to another. If you want to reduce network traffic on a particular route, increase the metric count on the appropriate router.

The interface configuration file for the secondary interface might look like the following:

```
cat /etc/config/ifconfig-2.options

netmask 255.255.255.0
broadcast 129.38.50.0
-arp
metric 4
```

The interface configuration file above indicates that the Class B network is subnetted using 8 bits of the host ID for the subnet. It uses 0 as the broadcast address instead of the default 1. ARP has been disabled and the metric (hop) count has been set to 4 to discourage router traffic.

Creating a Local Network Script

To start and terminate locally developed network daemons, or to publish ARP entries and set routes, create the separate shell script */etc/init.d/network.local*. Make symbolic links in */etc/rc0.d* and */etc/rc2.d* to this file to have it called during station start-up and shutdown:

```
ln -s /etc/init.d/network.local /etc/rc0.d/K39network
ln -s /etc/init.d/network.local /etc/rc2.d/S31network
```

See */etc/init.d/network* for the basic format of the script. Also refer to *network*(1M), *rc2*(1M), and *rc0*(1M) for more information.

Turning On Remote Access Logging

Several network daemons have an option that lets you log remote accesses to the station log file */var/adm/SYSLOG* by using *syslogd*(1M). Sites connected to the Internet should use this feature. To enable logging for *ftpd*(1M), *tftpd*(1M), and *rshd*(1M), edit */etc/inetd.conf* and add "-l " after the right-most instance of *ftpd* and *tftpd*. Add "-L" after the right-most instance of *rshd.* For additional *ftp* logging, add "-ll" to the *ftpd* entry. Signal *inetd* to reread its file after you have added your changes:

```
/etc/killall -HUP inetd
```

Remote logins by means of *rlogin*(1), *telnet*(1), and the 4DDN *sethost*(1) programs can be logged by *login*(1). Edit */etc/default/login* and add the keywords "SYSLOG=ALL" or "SYSLOG=FAIL" to it. For example, this command in the *login* file logs successful and failed local and remote login attempts to *syslogd*(1M):

```
syslog=all
```

See the *login*(1) reference page for details.

Chapter 18

Managing a Network

Chapter 18 covers the day-to-day management of your network. This is the chapter you will turn to most often for your networking questions. Topics include:

- *Tools for network management.*
- *Interpreting network statistics.*
- *Network performance.*
- *Network servers.*
- *Network packet sizes.*
- *Kernel settings that affect networking.*

Chapter 18

Managing a Network

This chapter provides information on how to manage a network after installation. Management includes maintenance, monitoring, and problem isolation. This chapter provides brief descriptions of the various network management tools, help in interpreting network statistics, a discussion on the factors that can impact network performance, and some of the network kernel configuration options. The following topics are covered:

- The available tools for network management. See "Network Management Tools" on page 585.
- How to interpret network statistics. See "Interpreting Network Statistics" on page 588.
- The various factors that affect network performance. See "Factors Affecting Network Performance" on page 591.
- Setting up network servers. See "Network Servers" on page 593.
- Information on network packet sizes. See "Packet Size" on page 593.
- Information on kernel settings that affect networking. See "Kernel Configuration" on page 594.

Network Management Tools

This section describes a number of standard and some optional networking tools at your disposal for managing the day to day operation of your network. Except as noted, the standard networking tools reside in the */usr/etc* directory. See the online reference pages for additional information.

ifconfig (1M) Configures the network interface devices. *ifconfig* is performed at boot time by the master network configuration script */etc/init.d/network*.

Interface configuration includes enabling and disabling of the interface and any interface options that should be set when the interface is configured. Options include support for the Address Resolution Protocol (ARP), routing metrics, netmask, broadcast addresses, etc. The *ifconfig* tool used with the station's interface name displays the current interface configuration. Some of the interface names for Silicon Graphics models include, *et0* - Power Series, *ec0* for the Personal IRIS and IRIS Indigo, *fxp0* for Professional series, *ipg0* and *xpi0* for FDDI, etc.

netstat (1M) Displays various network-related data structures that are useful for monitoring and troubleshooting a network. It has many options to show information about all or specific interfaces (**-i** or **-I**), routing tables (**-r** and **-M**), socket information (**-a** or **-A**), queue information (**-iq**), network memory (**-m**) and protocols (**-p**).

arp (1M) Displays and manipulates *arp* table entries located in cache. *arp* options include **-a** for all entries, **-d** to delete an entry, **-s** to publish an entry and act as a server for this entry, and **-f** to pull information from a specified file versus */dev/kmem. arp* is a good tool for troubleshooting physical address resolution on a network. *arp* does not display the local station's Ethernet address. To get a local station's Ethernet address, use the *netstat* command with the **-ia** options.

rpcinfo (1M) Provides information about Remote Procedure Call (RPC) based programs on local and remote stations. This is an excellent tool for isolating network problems related to the RPC service. The information provided by *rpcinfo* includes a list of *rpc*-based applications (*portmapper*, NIS, *rstatd*, etc.), the program number, version number, protocol (TCP/UDP), and associated port number. If you are running an RPC based network application and cannot get a response from the remote station, use the *rpcinfo* tool to ensure that the remote station supports the desired application.

ping (1M) Tests and measures general network performance. It is based on the Internet Control Management Protocol (ICMP) and sends an ECHO_REQUEST soliciting an ECHO_RESPONSE, thereby creating a two-way stream. It

provides general information about packet loss and round trip time. *ping* increases network load; this factor should be considered when testing a network with *ping*.

spray (1M) Sends a one-way stream of packets to a station using remote procedure calls. It reports information about the transfer rate and the number of packets received by the remote station. It provides very limited information about general network performance.

rtquery (1M) Sends a request to a designated station for information on the station's network routing tables (*routed* or *gated*). This tool is especially useful for troubleshooting routing problems.

traceroute (1M)

Tracks packets as they journey through the network. This tool is very useful for isolating network and router faults in a large heterogeneous network. It displays the names and addresses of all the intermediary routers that support the Internet Protocol "time-to-live" (TTL) field. It also displays the amount of time the packet spends traveling to the router, on the router, and leaving the router. *traceroute* increases network load; this factor should be considered when testing a network with *traceroute*.

route (1M) Manipulates the network routing tables. Typically, the routing tables are handled automatically by the *routed* or *gated* daemon. However, *route* can be used to create, maintain, and delete static routing tables, to flush routing tables, and to show metric information about routes. To have static routes incorporated at startup, modify the file */etc/gateways* and */etc/config/routed.options*.

/usr/bin/rup (1C)

Displays status information, including uptime and load average, about remote stations using Sun RPC broadcasts. If no specific station is specified, it uses broadcasting and returns information about stations on the local network; broadcasting does not go through routers. This tool is useful for isolating physical problems with a station or the network.

ttcp (1) Used to test Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) performance. This tool provides a more realistic measurement of performance than the standard tests (*spray, rup*, *ping*). It allows measurements to be taken at both the local and remote end of the transmission.

These network management tools are available as options for use on any Silicon Graphics 4D-IRIS station:

NetVisualyzer
A passive network management product. It offers a set of graphical traffic monitoring, diagnostics, planning, and performance analysis tools that provide network information and statistics for Ethernet or FDDI networks in a visually intuitive form. NetVisualyzer is comprised of six tools: NetLook, NetGraph, NetCPA, Analyzer, RouteQuery, and *TraceRoute*. NetVisualyzer allows you to view and monitor your network, collect network statistics and generate reports based on those statistics, and decode heterogeneous packets layer by layer.

SPECTRUM A UNIX based network management tool for large-scale, multi-LAN, multi-vendor networks. It provides you with graphical, real-time views of your network. These views represent network location layouts, network topology maps, and device front panels. The information provided by these views allows you to intelligently manage, monitor, and configure your network.

IRIS Networker
An automatic backup and recovery service for networked stations that performs backups of client systems to a centralized server. Backup schedules are specified by the network administrator. Stations in the network may be heterogeneous.

Interpreting Network Statistics

The network management tools provide the network administrator with valuable information about the network. However, the presentation of these

statistics can be overwhelming. This section illustrates how to use three of the most common management tools and how to interpret the network statistics generated by these tools.

The ping Tool

The *ping* tool tests and measures general network performance. It tells you when there is a problem with your network. The most important piece of information provided by *ping* is the *percentage packet loss*. Ideally, you want to see 0% packet loss; however, anything under .01% is acceptable. This low packet loss threshold is required because many network applications transmit large packets. If 0.1% of a packet is lost, the entire packet must be retransmitted. This can cause a network to be saturated by retransmissions.

The following example uses *ping* in its simplest form, but the information obtained is very useful. The *ping* tool is testing and measuring traffic between the local station and the station *testcase*. See the reference page for more details about the many *ping* options.

```
/usr/etc/ping testcase

PING testcase (192.55.43.4): 56 data bytes

64 bytes from 192.55.43.4: icmp_seq=0 ttl=255 time=0 ms
64 bytes from 192.55.43.4: icmp_seq=1 ttl=255 time=0 ms
64 bytes from 192.55.43.4: icmp_seq=2 ttl=255 time=0 ms
64 bytes from 192.55.43.4: icmp_seq=3 ttl=255 time=0 ms
64 bytes from 192.55.43.4: icmp_seq=4 ttl=255 time=0 ms

----testcase PING Statistics----

5 packets transmitted, 5 packets received, 0% packet loss

round-trip (ms)  min/avg/max = 0/0/0
```

The percentage packet loss is highlighted in bold. Again, 0% packet loss is the goal. Anything over 0.1% should be investigated further.

The ttcp Tool

The *ttcp* tool provides a realistic measurement of network performance between two stations as it allows measurements to be taken at both the local and remote end of the transmission. This tool measures the network

throughput. As with all network management tools, the statistics must be interpreted with the network configuration and applications in mind. For example, the statistics generated from a *ttcp* probe between two stations with routers in between results in lower throughput than if the stations were located on the same network. On the same note, users running applications that transmit large data structures see slower throughput than users running applications that transmit smaller data structures.

In any case, on a relatively quiet network, you should expect to see throughput in the 700 KB/sec or greater range. Throughput of 500 KB/sec or less is questionable, and 400 KB/sec or less may indicate a definite network problem.

The following example illustrates the statistics you might see if you ran a simple *ttcp* test between the stations *sheridan* and *longstreet* (two workstations) on a clean network. See the *ttcp* reference page for details about the many *ttcp* options.

On *sheridan*, give the command:

ttcp -r -s

You see the following output:

```
ttcp-r: buflen=8192, nbuf=2048, align=16384/0, port=5001 tcp
ttcp-r: socket ttcp-r: accept from 192.102.108.4
ttcp-r: 16777216 bytes in 19.99 real seconds = 819.64 KB/sec
+++
ttcp-r: 10288 I/O calls, msec/call = 1.99, calls/sec = 514.67
ttcp-r: 0.1user 3.4sys 0:19real 17%
```

On *longstreet*, give the following command:

```
ttcp -t -s sheridan
```

You see the following output:

```
ttcp-t: buflen=8192, nbuf=2048, align=16384/0, port=5001
tcp  -> sheridan
ttcp-t: socket
ttcp-t: connect
ttcp-t: 16777216 bytes in 19.98 real seconds = 820.02 KB/sec
```

```
+++
ttcp-t: 2048 I/O calls, msec/call = 9.99, calls/sec = 102.50
ttcp-t: 0.0user 2.3sys 0:19real 12%
```

The throughput statistics are highlighted in bold and are represented by KB/sec. The throughput on the station *sheridan* is 819.64 KB/sec and the throughput on the station *longstreet* is 820.02 KB/sec. Both throughput values indicate good network performance between the stations.

The *netstat* Tool

The *netstat* tool displays various network-related data structures that are useful for monitoring and troubleshooting a network. Detailed statistics about network collisions can be captured with the *netstat* tool. Collision rates anywhere between 5% and 20% indicate a problem with the network. The problem could be physical (bad tap, transceiver, loose terminator, and so on.) or there might be too much traffic on the network.

This example illustrates the statistics you might see on a station using *netstat*. See the reference page for details about the many *netstat* options:

```
netstat -i
Name Mtu  Network  Address  Ipkts   Ierrs  Opkts   Oerrs Coll
enp0 1500 b1-channels thumper 498690  937   1066135   3  4858
lo0  32880 loopback localhost 1678915 0     1678915   0   0
```

The collision rate is approximately 0.45%, well within the acceptable range.

Factors Affecting Network Performance

A variety of factors can impact network performance, including hardware problems, network configuration, network applications, and packet size.

Hardware Problems

Hardware problems can cause a network to be slow or inoperable. These problems are usually in the form of packet loss or corruption. Both of these

problems can cause increased network traffic to the point of unmanageable congestion. Items to check at the physical level are:

Controller board
: Even if the network media bandwidth is capable of handling the network traffic load, the individual station may not be able to handle the traffic. This is evidenced by a high degree of traffic on the network interface for no apparent reason. This traffic can be seen using the *gr_osview*(1M) tool (see the *gr_osview* online reference page for options to see network traffic statistics). If traffic is unusually heavy on the interface, then there may be a problem with the controller, or the controller may be too slow to handle the volume of traffic. You may need a high-speed controller like the Efast card.

Transmitter and controller
: Ensure that the Signal Quality Error (SQE), also called *heartbeat*, is disabled on both the transmitter and controller. SQE can cause unnecessary network traffic between the local station and the transceiver. See the installation guides for your network controller and transceiver for instructions on disabling SQE. By default, all Silicon Graphics' network controller boards are shipped with SQE disabled.

Physical problems with the media
: Cables, taps, and other hardware will periodically break or malfunction. A Time Domain Reflectometer (TDR) is essential for troubleshooting Ethernet cable problems. A good analyzer is also strongly recommended to assist in isolating network physical problems. Silicon Graphics' NetVisualyzer product supplies a visual network analyzer ideal for locating physical problems with the media.troubleshooting network media.

Network Configuration

The network configuration or topology can also adversely effect network performance. Check for the following conditions:

- Check network configuration to ensure that it is within the official guidelines for the media and topology. Pay special attention to the number and location of repeaters and bridges.
- Consider work group affiliations when determining which network is best suited for a particular station. Planning a network based on work group affiliations can reduce the amount of intranetwork traffic. Put stations that routinely share resources on the same net (NFS mounting, same NIS domain, electronic mail, financial databases).
- If connecting large subnets, use a dedicated router (station that performs routing function only) as the primary router. Ensure that there are at least two ways in and out of a network, because one of the routers will eventually fail. The router should also use appropriate filters to filter out unnecessary traffic.

Network Servers

Some network servers (*rwhod*, *rtnetd*, etc.) can have an undesirable affect on the network or network interface. For example, if a workstation is a multi-processor or is running real-time processes, the *rtnetd* daemon may be running on the station. This daemon is responsible for preempting incoming network packets to provide better response time for real-time processes. This is perfectly acceptable if the user is aware of the trade-offs between network processing and real-time processing. Some network servers should be evaluated individually.

Do not load *rtnetd* software on routers or other network intensive stations (mail servers, NIS and DNS servers, etc.).

Packet Size

The Maximum Transfer Unit (MTU) for data on the Ethernet is 1500 bytes. Network performance and efficiency increase with packet size up to the MTU for the medium. Packets that are larger than the media’s MTU must be broken into smaller packets (fragmented) to fit within the medium's MTU. Applications and services must be configured to transmit MTU size packets.

Kernel Configuration

You can change several parameters to customize network behavior for local configurations. The parameters listed Table 18-1 are in the */var/sysgen/master.d/bsd* configuration file. For details on reconfiguring the kernel after changing this file, see Chapter 5, "Tuning System Performance." Some of these listed options are also discussed in Appendix A, "IRIX Kernel Tunable Parameters."

Kernel Tunable Options

Table 18-1 Kernel Configuration Options

Parameter	Meaning
tcp_sendspace tcp_recvspace udp_sendspace udp_recvgrams	These parameters determine the default amount of buffer space used by TCP (SOCK_STREAM) and UDP (SOCK_DGRAM) sockets. The *tcp_sendspace* and *tcp_recvspace* parameters define the initial buffer space allocated to a socket. The *udp_sendspace* parameter defines the default maximum size of UDP datagrams that can be sent. The *udp_recvgrams* parameter determines the number of maximally sized UDP datagrams that can be buffered in a UDP socket. The total receive buffer size in bytes for each UDP socket is the product of *udp_sendspace* and *udp_recvgrams*. A program can increase or decrease the send buffer and receive buffer sizes for a socket with the SO_SNDBUF and SO_RCVBUF options to the *setsockopt*(2) system call. Many older TCP implementations have problems with large TCP *sendspace*/*recvspace* values. This should be decreased from 60 to 24 in environments where older stations have problems communicating.

For 4.2BSD compatibility, the IRIX system limits its initial TCP sequence numbers to positive numbers.

PC Connectivity

Many industry-standard personal computers with TCP/IP implementations experience difficulty connecting to Silicon Graphics workstations and

servers. This is because of the increased size of the *tcp_sendspace* and *tcp_recvspace* variables in the IRIX file */var/sysgen/master.d/bsd*.

To allow your personal computers to connect successfully, change the values of the above variables from the default (60 * 1024) to (24 * 1024) and reconfigure the kernel with the *lboot*(1M) command. For more information on reconfiguring these values, see Chapter 5, "Tuning System Performance."

Chapter 19

The BIND Name Server

Chapter 19 covers the BIND name server. This is the feature that allows you to address systems on your network by easily remembered hostnames rather than by numeric IP addresses. Topics covered in this chapter include:

- *Setting up BIND.*
- *Configuring BIND.*
- *Maintaining and Troubleshooting BIND.*

Chapter 19

The BIND Name Server

The Berkeley Internet Name Domain (BIND) server implements the Internet Domain Name Service (DNS) for the IRIX operating system. A name server is a network service that enables clients to name resources or objects in the network and share this information with other network objects. In effect, a name server is a distributed database system for objects in a computer network. All IRIX network programs can use BIND to store and retrieve station names and addresses. You can use BIND to replace the original *host* table lookup of information in the */etc/hosts* file.

BIND has two parts: the name server program, *named,* and a set of C library "resolver" routines that access the server. *named* is a daemon that runs in the background and responds to UDP and TCP queries on a well-known network port. The library routines reside in the standard C library, *libc.a.* The host-address lookup routines *gethostbyname*(3N), *gethostbyaddr*(3N), and *sethostent*(3N) use the resolver routines to query the name server. The resolver library routines described in *resolver*(3N) include routines that build query packets and exchange them with the name server.

The following topics are covered in detail in this chapter:

- The Domain Name Service. See "The Domain Name Server" on page 600.
- The organization of BIND. See "Organization of BIND" on page 601.
- BIND server and client information. See "BIND Servers and Clients" on page 603.
- The named server daemon. See "The named Server Daemon" on page 604.
- Registering your BIND domain name. See "Registering Your BIND Domain Name" on page 605.
- The BIND database files. See "The BIND Database Files" on page 606.

- Setting up a BIND configuration. See “Setting Up a BIND Configuration” on page 613.
- Managing the BIND environment. See “Managing the BIND Environment” on page 627.
- Troubleshooting and debugging BIND problems. See “Debugging named” on page 629.

The Domain Name Server

The basic function of the name server is to provide information about network objects by answering queries. The specifications for this name server are in RFCs 974, 1034, and 1035. You can obtain these documents in either of two ways:

- You can request them from the Internet Network Information Center (InterNIC) at the address listed in Chapter 16, “Obtaining an Internet Address”
- You can transfer the files:

  templates/domain-template.txt

  templates/in-addr-template.txt

  from the site:

  FTP.RS.INTERNIC.NET

  by using the FTP “anonymous” account. Also refer to the reference pages *named*(1M), *resolver*(3N), and *resolver*(4) for additional details.

When registering a domain for a connected network, be sure to register the reverse address domain (IN- ADDR.ARPA) for your networks.

Host table lookup routines, such as those using the */etc/hosts* file, require that the master file for the entire network be maintained at a central location by a few people. This approach works well for small networks where there are only a few stations and there is cooperation among the different organizations responsible for them. However, this approach does not work well for large networks where stations cross organizational boundaries.

The advantage of the name server over host table lookup is that it eliminates the need for a single, centralized clearinghouse for all names. The authority for this information can be delegated to the organizations on the network that are responsible for it.

Organization of BIND

The collection of all domains is considered the *domain space*. The name server perceives the network as a hierarchy of domains. The name space is organized as a tree according to organizational or administrative boundaries: Each node in the tree, called a *domain*, is given a label. The name of the domain is the concatenation of all the domain labels from the root to the current domain. The labels are listed from right to left and are separated by dots. A label needs to be unique only within its domain.

The whole space is partitioned into several non-overlapping areas of authority called *zones*. Information in each zone is handled by the zone's "authoritative" or "master" name server(s). Each zone starts at a domain and extends down to the leaf domains, or to domains where other zones start. Zones usually represent administrative boundaries.

The current top-level domains registered with the Network Information Center are:

arpa	a temporary domain for stations, also used as the top-level domain for address-to-name mapping
com	companies and businesses
edu	universities and other educational institutions
gov	government agencies
mil	military organizations
net	various network-type organizations, network management-related organizations, such as information centers and operations centers
org	technical support groups, professional societies, or similar organizations

There are also many national domains. For example, DE for Germany and FR for France.

An example of a domain name for a station at the University of California, Berkeley is:

```
monet.berkeley.edu.
```

The top-level domain for educational organizations is edu. Berkeley is a subdomain of edu, and *monet* is the name of the station.

Figure 19-1 shows a part of the current domain name space. Note that the tree is a very small subset of the actual name space.

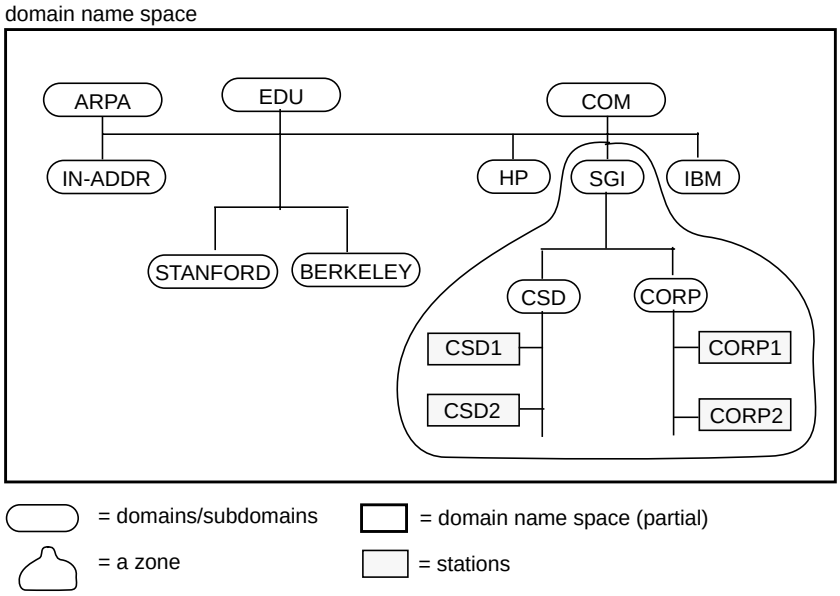


Figure 19-1 Domain Name Space (partial view)

BIND Servers and Clients

BIND is based on a server-client relationship. There are several server configurations and a client configuration. You can configure the BIND environment in several ways, depending on the degree of authority and network connectivity.

Master Servers

A master server for a domain is the authority for that domain. This server maintains all the data corresponding to its domain. Each domain should have at least two master servers: a primary master, and a secondary master to provide backup service if the primary is unavailable or overloaded. A server can be a master for multiple domains, serving as primary for some domains and secondary for others.

A primary master server is a server that loads its data from a file on disk. This server can also delegate authority to other servers in its domain. A secondary master server is a server that is delegated authority and receives its data for a domain from a primary master server. At boot time, the secondary server requests all the data for the given domain from the primary master server. This server then periodically checks with the primary server to see if it needs to update its data.

Root servers are the master servers for the root and top-level Internet domains. They are listed in the *root.cache* file described in "BIND's root.cache File" on page 612.

Slave and Forwarding Servers

A slave server always forwards queries it cannot satisfy locally to a fixed list of forwarding servers, instead of interacting with the master name server for the root and other domains. There may be one or more forwarding servers, and they are tried in turn until the list is exhausted.

A slave-and-forwarder configuration is useful when you do not want all the servers at a given site to interact with the rest of the Internet servers. The stations might be administratively prohibited from having Internet access.

To give the stations the appearance of access to the Internet domain system, the stations could be slave servers to the forwarding server on the gateway station. The gateway server would forward the queries and interact with other name servers on the Internet to resolve each query before returning the answer. A benefit of using the forwarding feature is that the central station develops a more complete cache of information which all the stations can take advantage of. The use of slave mode and forwarding is discussed further in "Setting Up a BIND Configuration" on page 613.

Caching-Only Server

A caching-only server is not authoritative for any domain. It services queries and asks other servers, who have the authority, for needed information.

Note that all servers cache data until the data expires, based on a *time-to-live* field attached to the data received from another server.

Clients

A BIND client accesses the name servers that run on other stations in the network. The *named* server does not run on the client station.

The named Server Daemon

The *named* server must be running on any station providing the Domain Name Service; it does not run on client stations. The server process for BIND is the daemon */usr/sbin/named*. The *named* daemon is started automatically during station startup if the configuration flag */etc/config/named* is "on." See the *chkconfig*(1M) online reference page for more details.

Table 19-1 summarizes the general characteristics for various BIND server configurations.

Table 19-1 BIND Server Configurations

Primary Server	Secondary Server	Caching-only Server	Forwarder Server	Slave Server
Authoritative server for the domain	"Delegated" authority from primary server	Non-authoritative	Non-authoritative	Non-authoritative
Loads data from local file	Loads data from primary server	Answers queries or forwards queries to authoritative servers	Answers recursive requests or interacts with other name servers before answering requests	Accesses data from specified list of servers (forwarders)

All server configurations must run the *named* server daemon. The client accesses data from the name servers specified in its *resolv.conf* file. It does not run the domain server, *named*.

Registering Your BIND Domain Name

To set up a domain on the Internet, contact the InterNIC Registration Services and request the appropriate domain registration form. You can reach the InterNIC by electronic mail at:

```
HOSTMASTER@INTERNIC.NET
```

Or you can use anonymous ftp to transfer the files:

templates/domain-template.txt

and

templates/in-addr-template.txt

from the Internet station FTP.RS.INTERNIC.NET.

When registering a domain for a connected network, be sure to register the reverse address domain (IN-ADDR.ARPA) for your networks.

The BIND Database Files

This section discusses the various BIND database files. Examples of these files are provided in "Setting Up a BIND Configuration" on page 613.

In IRIX, the *named* database files are stored in the */var/named* directory. A *README* file contains a short summary of the setup procedure and a list of official names for BIND clients and servers. Typically, servers are also clients. A file that may or may not be present on your station, but is mandatory if the station is going to be a BIND client, is the */etc/resolv.conf* file. In previous releases of IRIX, this file was located in the */usr/etc* directory. A symbolic link has been to the old filename has been created for backwards compatibility.

The */var/named/Examples* subdirectory contains sample *named* database files. The files in the *Examples* directory should be used and changed to reflect your setup. These files use the record format described in Appendix E, "BIND Standard Resource Record Format."The database files needed to set up your BIND environment are:

- *named.boot*
- *root.cache*
- *named.hosts*
- *named.rev*
- *localhost.rev*

Note: If your network has more than one domain, incorporate the domain name as part of the *named.hosts*, *named.rev*, and *localhosts.rev* file name when you create your versions of these files.

The number and configuration of the database files depend on the station configuration (server type or client).

Table 19-2 summarizes the relationship between station configuration and the database files.

Table 19-2 BIND Database FIles

File Name	Primary Server	Secondary Server	Caching-only Server	Forwarder Server	Slave Server
named.boot	required	required	required	required	required
localhosts.rev	required	required	required	required	required
named.hosts	required	N/A	N/A	N/A	N/A
named.rev	required	N/A	N/A	N/A	N/A
root.cache	required	required	required	required	required
resolv.conf	optional	optional	optional	optional	optional

The only configuration file required for a BIND client is the *resolv.conf* file.

BIND and /etc/resolv.conf and /etc/hosts

The file */etc/resolv.conf* is read the first time *gethostbyname*(3N) or *gethostbyaddr*(3N) is called. The *resolv.conf* file has several functions:

- It defines the default domain or the default domain search list.
- It specifies the ordering of host resolution services used by *gethostbyname*(3N) and *gethostbyaddr*(3N).
- It lists Internet addresses of name servers.

The first two items apply to both client and server stations. The last item is required only by client stations. The file's format is described in detail in *resolver*(4).

To set up a station as a client of remote servers, add **nameserver** entries for the Internet addresses of the name servers to */etc/resolv.conf*. For example:

```
nameserver 128.32.130.12
```

You can specify up to three *nameserver* entries. It is usually not necessary to create this file if you have a local server running. An entry for the local server should use an Internet address of 0 (meaning "this station").

On client and server stations, the name in */etc/sys_id* should be set to the fully qualified domain name. For example:

```
monet.Berkeley.EDU
```

However, if you choose not to use fully qualified domain names, add a line with the keyword *domain* and the station's domain to the *resolv.conf* file. For example:

```
domain berkeley.edu
```

The *gethostbyname*(3N) and *gethostbyaddr(*3N) library routines are normally configured to access station information in this order:

1. NIS
2. BIND
3. Local */etc/hosts* file

You can change this behavior with the *hostresorder* keyword in */etc/resolv.conf*. See *resolver*(4) for details.

To enable the system manager to copy files from another station when it is in single-user mode, the */etc/hosts* file should contain entries for important stations in addition to the entries for the local station's network interface(s) and *localhost*. See *hosts*(4) for more information about the format.

BIND's Boot File

The boot file is first read when *named* starts up. It tells the server what type of server it is, which zones it has authority over, and where to get its initial data. The default name of this file is */etc/named.boot.* The template for this file is called */var/named/Examples/named.boot.master* (for primary server) and *named.boot.slave* (for secondary server).

To use a different file, create or modify the */etc/config/named.options* file with this entry:

```
-b other-bootfile-name
```

The recognized boot file structures are described in the subsections that follow.

Directory

The directory line specifies the directory in which the name server should run, allowing the other file names in the boot file to use relative path names.

```
directory /var/named
```

This entry is required. It makes sure *named* is in the proper directory when you try to include files by relative path names with $INCLUDE. It also allows *named* to run in a location that is reasonable for dumping core if necessary.

Primary Master

The line in the boot file that designates a primary server for a zone looks like this:

```
primary   Berkeley.EDU   named.hosts
```

The first field specifies that the server is a primary one for the zone stated in the second field. The third field is the name of the file from which the data is read.

Secondary Master

The line for a secondary server is similar to that for the primary except that it lists addresses of other servers (usually primary servers) from which the zone data is obtained. For example:

```
secondary  Berkeley.EDU 128.32.0.10 128.32.0.4 ucbhosts.bak
```

The first field specifies that the server is a secondary master server for the zone stated in the second field. The two network addresses specify the name servers that are primary for the zone. The secondary server gets its data across the network from the listed servers. It tries each server in the order listed until it successfully receives the data from a listed server.

If a file name is present after the list of primary servers, data for the zone is saved in that file. When the server first starts, it loads the data from the backup file if possible, and consults a primary server to check that the zone information is still up to date.

Caching-Only Server

All servers should have a line like this one in the boot file to prime the name server's cache:

```
cache   .   root.cache
```

All listed cache files are read when *named* starts up. Valid values are reinstated in the cache, and the root name server information in the cache files is always used to handle initial queries.

The name server needs to know the servers that are the authoritative name servers for the root domain of the network. The *root.cache* file primes the server's cache with the addresses of these higher authorities. This file uses the Standard Resource Record format (or Master File format) described in detail in Appendix F.

You do not need a special line to designate that a server is a caching server. What denotes a caching-only server is the absence of authority lines, such as *secondary* or *primary*, in the boot file.

Forwarders

Any server can make use of forwarders. For example, a server capable of processing recursive queries may try resolving queries on behalf of other stations. The *forwarders* command specifies forwarders by Internet address as follows:

```
forwarders    128.32.0.10     128.32.0.4
```

There are two main reasons to use forwarders. First, if your station does not have full network access, it cannot send IP packets to the rest of the network. Therefore, it must rely on a forwarder with access to the network. Second, the forwarder can see all queries as they pass through the server and, therefore, builds up a more complete cache of data than the cache in a typical station name server. In effect, the forwarder becomes a meta-cache from which stations can benefit, thereby reducing the total number of queries from that site to the rest of the network.

Slave Mode

You can use slave mode if, because of limited network access, use of forwarders is the only way to resolve queries. You can also use slave mode if you wish to prevent the name server from using forwarders other than those listed. Slave mode is activated by the following command in the boot file:

```
slave
```

If you use *slave,* you must specify forwarders. In slave mode, the server forwards each query to each of the forwarders until an answer is found or the list of forwarders is exhausted.

BIND's named.hosts File

This file contains the host-address database for your domain. It is required for primary servers.

BIND's named.rev File

This file specifies the IN-ADDR.ARPA domain, which is used to translate IP addresses into names. Because Internet addresses do not fall within domain boundaries, this special domain was formed to allow inverse mapping. The IN-ADDR.ARPA domain for a station has four labels preceding it. These labels correspond to the four octets of an Internet address in reverse order. All four octets must be specified, even if an octet is zero.

For example, the Internet address 128.32.130.12 is located in the domain 12.130.32.128.IN-ADDR.ARPA. This reversal of the address allows for the natural grouping of stations in a network.

An IN-ADDR.ARPA domain can also represent a network. For example, if the ARPANET is network 10, there is a domain called 10.IN-ADDR.ARPA.

BIND's localhost.rev File

This file specifies the IN-ADDR.ARPA domain of the local loopback interface's network address, 127.0.0.1. The address is better known as the *localhost* address. Many important network programs depend on the information in this domain. This file is required on all servers.

BIND's root.cache File

This file, by default, contains the initial cache data for root domain servers. It is required, in one form or another, on all servers.

The /etc/config/named.options File

This file is optional. It is used during station startup and by the *named.restart* script. Specify command-line arguments for *named* in this file. See *named*(1M) for details on the options.

Setting Up a BIND Configuration

This section provides an example of how a BIND environment might be organized and the procedure for configuring the various servers and client stations. The example assumes you are connected to the Internet. When setting up your own environment, replace the variables in the example with your own BIND environment variables. The example is based on these variables:

- One domain is *fruit.com*, network address 192.35.10, and the network is attached to the Internet
- Primary server is *apples.fruit.com*, internet address 192.35.10.1
- Secondary server is *oranges.fruit.com*, internet address 192.35.10.2
- Forwarding server is *banana.fruit.com*, internet address 192.35.10.3
- Caching-only server is *guava.fruit.com*, internet address 192.35.10.4
- Slave servers are *pineapple1.fruit.com*, and *pineapple2.fruit.com*, Internet addresses 192.35.10.8 and 192.35.10.9
- Clients are *plum1.fruit.com, plum2.fruit.com*, and *plum3.fruit.com*, internet addresses 192.35.10.5, 192.35.10.6, and 192.35.10.7.

Figure 19-2 illustrates the example BIND environment described above. Station names in the figure are shortened for illustrative purposes only.

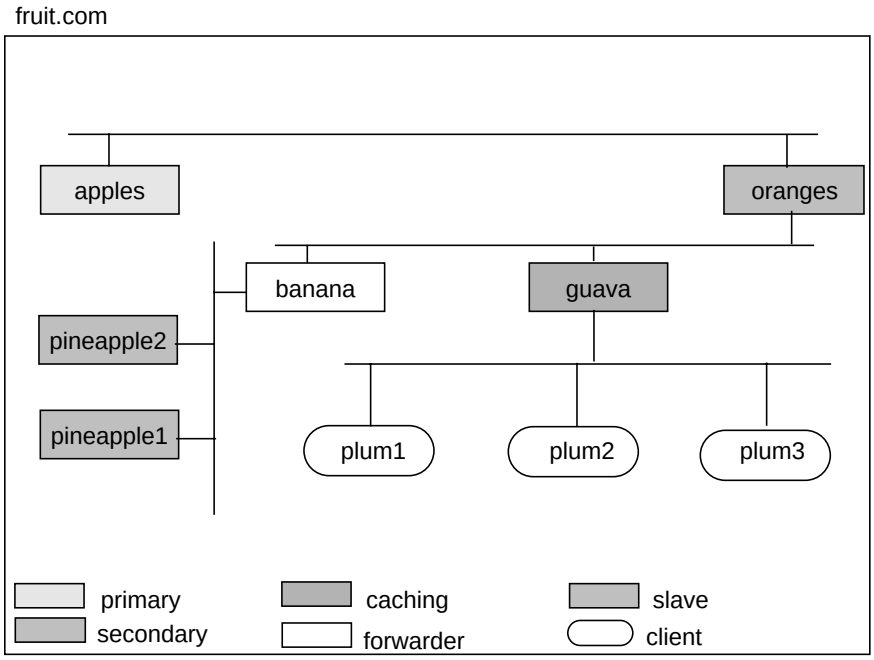


Figure 19-2 Example BIND Configuration

Configuring the Primary Server

Use this procedure to configure a primary server:

1. Log in as **root**.
2. Move to the named example directory.

   cd /var/named/Examples
3. Copy the template files to the */var/named* directory.

   ```
   cp named.boot.master root.cache named.hosts \
   named.rev localhost.rev /var/named
   ```

4. Move *named.boot.master* to default file name.

   cd ..

   mv named.boot.master named.boot

5. Modify *named.boot* with your editor of choice to resemble the following:

```
;
; Boot file for apple.fruit.com, primary for fruit.com
;
directory /var/named
;type    domain     source host/file     backup file
cache    .          root.cache
primary  fruit.com  fruit.named.hosts
```

6. Modify the *named.hosts* file, here called *fruit.named.hosts*, to resemble the following:

```
; Authoritative data for fruit.com
;
@   IN  SOA  apples.fruit.com.named-mgr.apples.fruit.com.
            (1994021501 ; Serial
             10800      ; Refresh 3 hours
             3600       ; Retry 1 hour
             3600000    ; Expire  1000 hours
             86400 )    ; Minimum 24 hours
; authoritative name servers for fruit.com
          IN    NS     apples.fruit.com.
          IN    NS     oranges.fruit.com.
; address records for all hosts on the net
          IN     A     192.35.10.1
apples    IN     A     192.35.10.1
oranges   IN     A     192.35.10.2
banana    IN     A     192.35.10.3
guava     IN     A     192.35.10.4
plum1     IN     A     192.35.10.6
plum3     IN     A     192.35.10.7
pineapple1IN     A     192.35.10.8
pineapple2IN     A     192.35.10.9
localhost IN     A     127.0.0.1
; canonical or alias name for localhost
loghostIN        CNAME localhost
```

7. Modify the *localhost.rev* file to resemble the following:

```
;localhost.rev  -- PTR record for 127.1
;
@  IN  SOA  apples.fruit.com. named-mgr.apples.fruit.com.
                            (1994021501 ;Serial
                             10800      ;Refresh 3 hours
                             3600       ;Retry   1 hour
                             3600000    ;Expire 1000 hours
                             86400 )    ;Minimum 24 hours
; authoritative name servers for fruit.com
   IN      NS      apples.fruit.com.
   IN      NS      oranges.fruit.com.
0  IN      PTR     loopback.fruit.com.
1  IN      PTR     localhost.
```

8. Modify the *named.rev file* (*fruitnamed.rev*) to resemble the following:

```
;
;       @(#)named.rev   1.1     (Berkeley)      86/02/05
;
@  IN  SOA  apples.fruit.com. named-mgr.apples.fruit.com.
                             (1994021501 ; Serial
                              10800 ; Refresh 3 hours
                              3600  ; Retry   1 hour
                              3600000 ; Expire  1000 hours
                              86400 ); Minimum 24 hours
;authoritative name servers for fruit.com
               IN      NS     apples.fruit.com.
               IN      NS     oranges.fruit.com.
;named.rev addresses, by default, are the last two numbers
;of the internet addresses in reverse order, if Class B
;address.  If Class C address, then it's the last number.
1              IN    PTR      apples.fruit.com.
2              IN    PTR      oranges.fruit.com.
3              IN    PTR      banana.fruit.com.
4              IN    PTR      guava.fruit.com.
5              IN    PTR      plum1.fruit.com.
6              IN    PTR      plum2.fruit.com.
7             IN    PTR    plum3.fruit.com.
8             IN    PTR    pineapple1.fruit.com.
9             IN    PTR    pineapple2.fruit.com.
```

9. Use the default *root.cache* file if the primary server is attached to the Internet. If practical, you should obtain the most up-to-date list from FTP.RS.INTERNIC.NET using anonymous FTP. The list is kept in the file *domain/named.root*.

```
;
;This file holds the information on root name servers
;needed to initialize cache of Internet domain name
;servers
;(e.g. reference this file in the "cache . <file>"
;configuration file of BIND domain name servers).
;
;This file is made available by InterNIC registration
;services under anonymous FTP as
;           file                /domain/named.root
;           on server           FTP.RS.INTERNIC.NET
;       -OR- under Gopher at RS.INTERNIC.NET
;           under menu          InterNIC Registration
;                               Services (NSI)
;              submenu          InterNIC Registration
;                               Archives
;           file                named.root
;
;       last update:    April 21, 1993
;       related version of root zone:   930421
;
.                   99999999 IN  NS    NS.INTERNIC.NET.
NS.INTERNIC.NET.    99999999     A     198.41.0.4
.                   99999999     NS    KAVA.NISC.SRI.COM.
KAVA.NISC.SRI.COM.  99999999     A     192.33.33.24
.                   99999999     NS    C.NYSER.NET.
C.NYSER.NET.        99999999     A     192.33.4.12
.                   99999999     NS    TERP.UMD.EDU.
TERP.UMD.EDU.       99999999     A     128.8.10.90
.                   99999999     NS    NS.NASA.GOV.
NS.NASA.GOV.        99999999     A     128.102.16.10
                    99999999     A     192.52.195.10
.                   99999999     NS    NS.NIC.DDN.MIL.
NS.NIC.DDN.MIL.     99999999     A     192.112.36.4
.                   99999999     NS    AOS.ARL.ARMY.MIL.
AOS.ARL.ARMY.MIL.    99999999     A     128.63.4.82
                    99999999     A     192.5.25.82
.                   99999999     NS    NIC.NORDU.NET.
NIC.NORDU.NET.       99999999     A     192.36.148.17
; End of File
```

10. Enable *named* and reboot the station with the following commands:

```
chkconfig named on
reboot
```

Configuring the Secondary Server

Use this procedure to configure a secondary server:

1. Login as **root**.
2. Move to the *named* example directory.

   cd /var/named/Examples
3. Copy the template files to the */var/named* directory.

   cp named.boot.slave root.cache localhost.rev /var/named
4. Move *named.boot.slave* to default file name.

   cd ..

   mv named.boot.slave named.boot
5. Modify *named.boot* to look like the following:

   more named.boot

```
;
; Boot file for orange.fruit.com, secondary for fruit.com
;
directory  /var/named
; type     domain     source host/file  backup file
cache      .          root.cache
secondary  fruit.com  192.35.10.1       fruithosts.bak
```

6. Modify the *localhost.rev* file to resemble the following:

```
;
;localhost.rev  -- PTR record for 127.1
;
@  IN   SOA  apples.fruit.com. named-mgr.apples.fruit.com
                          (1994021501     ; Serial
                           10800    ; Refresh 3 hours
                           3600     ; Retry   1 hour
                           3600000  ; Expire  1000 hours
                           86400 )  ; Minimum 24 hours
; authoritative name servers for fruit.com
                IN     NS    apples.fruit.com.
                IN     NS    oranges.fruit.com.
0               IN     PTR   loopback.fruit.com.
1               IN     PTR   localhost.
```

7. Use the default *root.cache* file if the primary server is attached to the Internet. Pull the most up-to-date list from FTP.RS.INTERNIC.NET using anonymous FTP. The list is kept in the directory *domain/named.root*.

```
;
;This file holds the information on root name servers
;needed to initialize cache of Internet domain name
;servers
;(e.g. reference this file in the "cache . <file>"
;configuration file of BIND domain name servers).
;
;This file is made available by InterNIC registration
;services under anonymous FTP as
;           file                /domain/named.root
;           on server           FTP.RS.INTERNIC.NET
;       -OR- under Gopher at RS.INTERNIC.NET
;           under menu          InterNIC Registration
;                               Services (NSI)
;              submenu          InterNIC Registration
;                               Archives
;           file                named.root
;
;       last update:    April 21, 1993
;       related version of root zone:   930421
;
.                  99999999 IN  NS    NS.INTERNIC.NET.
NS.INTERNIC.NET.   99999999     A     198.41.0.4
.                  99999999     NS    KAVA.NISC.SRI.COM.
```

```
KAVA.NISC.SRI.COM. 99999999     A     192.33.33.24
.                  99999999     NS    C.NYSER.NET.
C.NYSER.NET.        99999999     A     192.33.4.12
.                  99999999     NS    TERP.UMD.EDU.
TERP.UMD.EDU.       99999999     A     128.8.10.90
.                  99999999     NS    NS.NASA.GOV.
NS.NASA.GOV.        99999999     A     128.102.16.10
                    99999999     A     192.52.195.10
.                   99999999     NS    NS.NIC.DDN.MIL.
NS.NIC.DDN.MIL.     99999999     A     192.112.36.4
.                   99999999     NS    AOS.ARL.ARMY.MIL.
AOS.ARL.ARMY.MIL.    99999999     A     128.63.4.82
                    99999999     A     192.5.25.82
.                   99999999     NS    NIC.NORDU.NET.
NIC.NORDU.NET.       99999999     A     192.36.148.17
; End of File
```

8. Enable *named* and reboot the station with the following commands:

   ```
   chkconfig named on
   reboot
   ```

Configuring a Caching-Only Server

Use this procedure to set up a caching-only server:

1. Log in as **root**.
2. Move to the *named* example directory:

   cd /var/named/Examples
3. Copy the template files to the */var/named* directory.

   cp named.boot.master root.cache /var/named
4. Move *named.boot.master* to default file name.

   cd ..

   mv named.boot.master named.boot

5. Modify *named.boot* to look like the following:

   more named.boot

```
;
;Boot file for guava.fruit.com,caching-only server for
;fruit.com
;Note that there should be one primary entry for each SOA
;record.
;
;
directory          /var/named
;type    domain  source host/file  backup file
cache    .       root.cache
```

6. Use the default *root.cache* file if the primary server is attached to the Internet. Pull the most up-to-date list from FTP.RS.INTERNIC.NET using anonymous FTP. The list is kept in the directory *domain/named.root*.

```
;
;This file holds the information on root name servers
;needed to initialize cache of Internet domain name
;servers
;(e.g. reference this file in the "cache . <file>"
;configuration file of BIND domain name servers).
;
;This file is made available by InterNIC registration
;services under anonymous FTP as
;           file                /domain/named.root
;           on server           FTP.RS.INTERNIC.NET
;       -OR- under Gopher at RS.INTERNIC.NET
;           under menu          InterNIC Registration
;                               Services (NSI)
;              submenu          InterNIC Registration
;                               Archives
;           file                named.root
;
;       last update:    April 21, 1993
;       related version of root zone:   930421
;
.                 99999999 IN  NS    NS.INTERNIC.NET.
NS.INTERNIC.NET.   99999999     A     198.41.0.4
.                 99999999     NS    KAVA.NISC.SRI.COM.
KAVA.NISC.SRI.COM. 99999999     A     192.33.33.24
.                 99999999     NS    C.NYSER.NET.
```

```
C.NYSER.NET.        99999999     A     192.33.4.12
.                  99999999     NS    TERP.UMD.EDU.
TERP.UMD.EDU.       99999999     A     128.8.10.90
.                  99999999     NS    NS.NASA.GOV.
NS.NASA.GOV.        99999999     A     128.102.16.10
                    99999999     A     192.52.195.10
.                   99999999     NS    NS.NIC.DDN.MIL.
NS.NIC.DDN.MIL.     99999999     A     192.112.36.4
.                   99999999     NS    AOS.ARL.ARMY.MIL.
AOS.ARL.ARMY.MIL.    99999999     A     128.63.4.82
                    99999999     A     192.5.25.82
.                   99999999     NS    NIC.NORDU.NET.
NIC.NORDU.NET.       99999999     A     192.36.148.17
; End of File
```

7. Enable *named* and reboot the station with the following commands:

   ```
   chkconfig named on
   reboot
   ```

Configuring the Forwarding Server

Use this procedure to set up a forwarding server:

1. Log in as **root**.
2. Move to the *named* example directory:

   cd /var/named/Examples
3. Copy the template files to the */var/named* directory.

   cp named.boot.master root.cache localhost.rev /var/named
4. Move *named.boot.master* to default file name.

   cd ..

   mv named.boot.master named.boot

5. Modify *named.boot* to look like the following:

   more named.boot

```
;
;Boot file for banana.fruit.com, forwarder server
;for fruit.com
;Note that there should be one primary entry for each
;SOA record.
;
;
directory       /var/named

;type       domain        source host/file   backup file
cache       .             root.cache
forwarders  192.35.10.1   192.35.10.2
```

6. Modify the *localhost.rev* file to resemble the following:

```
;
;       localhost.rev  -- PTR record for 127.1
;
@   IN   SOA  apples.fruit.com. named-mgr.apples.fruit.com
                              (1994021501     ; Serial
                               10800   ; Refresh 3 hours
                               3600    ; Retry   1 hour
                               3600000 ; Expire  1000 hours
                               86400 ) ; Minimum 24 hours
; authoritative name servers for fruit.com
                IN      NS      apples.fruit.com.
                IN      NS      oranges.fruit.com.
0               IN      PTRloopback.fruit.com.
1  IN PTR localhost.
```

7. Use the default *root.cache* file if the primary server is attached to the Internet. Pull the most up-to-date list from FTP.RS.INTERNIC.NET using anonymous FTP. The list is kept in the directory *domain/named.root*.

```
;
;This file holds the information on root name servers
;needed to initialize cache of Internet domain name
;servers
;(e.g. reference this file in the "cache . <file>"
;configuration file of BIND domain name servers).
;
```

```
;This file is made available by InterNIC registration
;services under anonymous FTP as
;           file                /domain/named.root
;           on server           FTP.RS.INTERNIC.NET
;       -OR- under Gopher at RS.INTERNIC.NET
;           under menu          InterNIC Registration
;                               Services (NSI)
;              submenu          InterNIC Registration
;                               Archives
;           file                named.root
;
;       last update:    April 21, 1993
;       related version of root zone:   930421
;
.                  99999999 IN  NS    NS.INTERNIC.NET.
NS.INTERNIC.NET.    99999999     A     198.41.0.4
.                  99999999     NS    KAVA.NISC.SRI.COM.
KAVA.NISC.SRI.COM. 99999999     A     192.33.33.24
.                  99999999     NS    C.NYSER.NET.
C.NYSER.NET.        99999999     A     192.33.4.12
.                  99999999     NS    TERP.UMD.EDU.
TERP.UMD.EDU.       99999999     A     128.8.10.90
.                  99999999     NS    NS.NASA.GOV.
NS.NASA.GOV.        99999999     A     128.102.16.10
                    99999999     A     192.52.195.10
.                   99999999     NS    NS.NIC.DDN.MIL.
NS.NIC.DDN.MIL.     99999999     A     192.112.36.4
.                   99999999     NS    AOS.ARL.ARMY.MIL.
AOS.ARL.ARMY.MIL.    99999999     A     128.63.4.82
                    99999999     A     192.5.25.82
.                   99999999     NS    NIC.NORDU.NET.
NIC.NORDU.NET.       99999999     A     192.36.148.17
; End of File
```

8. Enable *named* and reboot the station with the following commands:

   chkconfig named on

   reboot

Configuring a Slave Server

1. Log in as **root**.
2. Move to the *named* example directory.

   cd /var/named/Examples
3. Copy the template files to the */var/named* directory.

   cp named.boot.slave root.cache localhost.rev /var/named
4. Move *named.boot.master* to default file name.

   cd ..

   mv named.boot.slave named.boot
5. Modify *named.boot* to look like the following:

```
;
;Boot file for pineapple1.fruit.com, slave server for
;fruit.com
;
directory       /var/named
;type       domain      source host/file   backup file
cache       .           root.cache
forwarders  192.35.10.3
slave
```

6. Modify the *localhost.rev* file to resemble the following:

```
;
;       localhost.rev  -- PTR record for 127.1
;
@  IN   SOA   apples.fruit.com. named-mgr.apples.fruit.com
                         (1994021501     ; Serial
                          10800   ; Refresh 3 hours
                          3600    ; Retry   1 hour
                          3600000 ; Expire  1000 hours
                          86400 ) ; Minimum 24 hours
; authoritative name servers for fruit.com
                IN     NS      apples.fruit.com.
                IN     NS      oranges.fruit.com.
0               IN     PTR     loopback.fruit.com.
1               IN     PTR     localhost.
------
```

7. Use the default *root.cache* file if the primary server is attached to the Internet. Pull the most up-to-date list from FTP.RS.INTERNIC.NET using anonymous FTP. The list is kept in the directory *domain/named.root.:*

```
;
;This file holds the information on root name servers
;needed to initialize cache of Internet domain name
;servers
;(e.g. reference this file in the "cache . <file>"
;configuration file of BIND domain name servers).
;
;This file is made available by InterNIC registration
;services under anonymous FTP as
;           file                /domain/named.root
;           on server           FTP.RS.INTERNIC.NET
;       -OR- under Gopher at RS.INTERNIC.NET
;           under menu          InterNIC Registration
;                               Services (NSI)
;              submenu          InterNIC Registration
;                               Archives
;           file                named.root
;
;       last update:    April 21, 1993
;       related version of root zone:   930421
;
.                   99999999 IN  NS    NS.INTERNIC.NET.
NS.INTERNIC.NET.    99999999     A     198.41.0.4
.                   99999999     NS    KAVA.NISC.SRI.COM.
KAVA.NISC.SRI.COM.  99999999     A     192.33.33.24
.                   99999999     NS    C.NYSER.NET.
C.NYSER.NET.        99999999     A     192.33.4.12
.                   99999999     NS    TERP.UMD.EDU.
TERP.UMD.EDU.       99999999     A     128.8.10.90
.                   99999999     NS    NS.NASA.GOV.
NS.NASA.GOV.        99999999     A     128.102.16.10
                    99999999     A     192.52.195.10
.                   99999999     NS    NS.NIC.DDN.MIL.
NS.NIC.DDN.MIL.     99999999     A     192.112.36.4
.                   99999999     NS    AOS.ARL.ARMY.MIL.
AOS.ARL.ARMY.MIL.    99999999     A     128.63.4.82
                    99999999     A     192.5.25.82
.                   99999999     NS    NIC.NORDU.NET.
NIC.NORDU.NET.      99999999     A     192.36.148.17
; End of File
```

8. Enable *named* and reboot the station with the following commands:

```
chkconfig named on
reboot
```

Configuring the Client

Use this procedure to set up a BIND client:

1. Log in as **root**.
2. Create or modify the *resolv.conf* file to include the default domain name, the host resolution order, and the list of name servers. It should look something like this:

```
domain fruit.com
nameserver 192.35.10.4
nameserver 192.35.10.2
nameserver 192.35.10.1
hostresorder bind local
```

3. Rebooting the client is suggested, but not required.

Managing the BIND Environment

This section describes the steps involved in maintaining the databases. It details how to add and delete a station from the domain and how to add a new subdomain. It also discusses some of the scripts that manage the BIND database.

Adding a New Station

To add a new station to your zone files:

1. Edit the appropriate zone file for the station's domain.
2. Add an *A* record for each address of the station.
3. Add *CNAME*, *HINFO*, *WKS*, *RP*, and *MX* records (optional).
4. Add the reverse IN-ADDR entry for each station address in the appropriate zone files for each network the station is on.

Deleting a Station

To delete a station from the zone files:

1. Remove all the station's resource records from the zone file of the station's domain.
2. Remove all the station's *PTR* records from the IN-ADDR zone files for each network the station was on.

Adding Another Domain

To add a new subdomain to your domain:

1. Set up the other domain server, the new zone file, or both.
2. For each server of the new domain, add an *NS* record to the zone file of the parent domain.
3. Add any necessary glue address records. See Appendix E, "BIND Standard Resource Record Format" for details about glue records.

Management Scripts

You can use two shell scripts to manage *named*: */usr/sbin/named.reload* and */usr/sbin/named.restart*.

The /usr/sbin/named.reload Script

This shell script sends the HUP signal to *named,* which causes it to read *named.boot* and reload the database. All previously cached data is lost. Use this script when *named* is running and you want the internal database for *named* to reflect any changes you have made.

The /usr/sbin/named.restart Script

This shell script terminates the running *named* and starts a new one. Use this script when you have made changes to the *named.boot* file, or whenever you need to place the server in a known state.

Debugging named

When *named* is running incorrectly, first check */var/adm/SYSLOG* for any messages. For additional information, send *named* one of the following signals, using *killall*(1M) and defining SIG as INT, ABRT, USR1, or USR2:

```
/etc/killall -SIG named
```

INT	Dumps the current database and cache to */var/tmp/named_dump.db*. This dumping should indicate whether the database was loaded correctly.
ABRT	Dumps statistics data into */var/tmp/named.stats*. Statistics data is appended to the file.
USR1	Turns on debugging. Each subsequent USR1 increases the debug level. There are 10 debug levels, and each prints more detailed information. A debug level of 5 is useful for debugging lookup requests. The output goes to */var/tmp/named.run*.
USR2	Turns off debugging completely.

SYSLOG Messages

Using *syslog*(3B), *named* logs certain errors to */var/adm/SYSLOG*. This section lists important error messages and their meanings.

- *dname* `has CNAME and other illegal data`

  An alias has more than just a *CNAME* record. For example, since only "monet" should have the *A* record, the following is wrong:

  ```
  ucbmonet IN CNAME monet
  ucbmonet IN A 128.32.0.1
  ```

- `Attempted to query myself on ipaddr as name server for` *dname*

  The station is listed incorrectly as a forwarder in the *named.boot* file.

- `zoneref: Masters for secondary zone` *dname* `unreachable`

This station is a secondary server for *dname*. The primary server returned an invalid data response or, because of a network problem, the station was not able to contact the primary server to obtain the current state of the zone information.

- `Lame delegation to` *dname1* `received from` *ipaddr*`(purported server for` *dname2*`) on query on name [`*dname3*`]`

  The message indicates that the remote server at the specified address is supposed to be authoritative for the *dname2* domain but has returned an indication that implies it is not. The remote server or the server for its parent domain is misconfigured. This error message can be disabled if *named* is started with the **-L lamedel** option.

- `MAXQUERIES exceeded, possible data loop in resolving` *dname*

  The name server has tried to query too many other servers for the specified record. This might happen if each of two remote servers reply that the other has the desired information for *dname*.

- `Malformed response from` *ipaddr*

  The remote DNS server at the specified address returned a malformed packet. This message typically indicates an error in the remote server.

- `Bogus root NS` *dname1* `received from` *ipaddr* `on query on name[`*dname2*`] -- rejected`

  The name server at the specified address improperly returned NS records for the root domain. The records have been ignored. You can disable this error message if *named* is started with **-L rootns.**

- `Root NS` *dname1* `received from` *ipaddr* `on query on name [`*dname2*`]`

  The name server at the specified address returned NS records for the root domain. You can disable this informational message if *named* is started with the **-L rootns** option.

The nslookup Command

The *nslookup*(1) command is a useful debugging tool for querying local and remote name servers. *nslookup* operates in either interactive or non-interactive mode. Interactive mode allows the user to query the name server for information about various stations and domains or print a list of stations

in the domain. Non-interactive mode is used to print just the name and requested information for a station or domain. The following example of *nslookup* output gets the address record for the station *monet.berkeley.edu*.

```
Default Server:  ucbvax.berkeley.edu
Address:  128.32.133.1
> monet
Server:  ucbvax.berkeley.edu
Address:  128.32.133.1
Name:    monet.berkeley.edu
Address:  128.32.130.6
```

To exit, press **<Ctrl-D>** or **exit**. The *help* command summarizes available commands. The complete set of commands is described in the *nslookup*(1) reference page.

Chapter 20

IRIX Sendmail

Chapter 20 discusses IRIX sendmail. This is the network feature that allows you to send and receive mail with users on remote systems. Mail is one of the chief reasons for creating a network, so an understanding of the mail system is crucial for effective system administration. Topics covered here include:

- *Planning for sendmail.*
- *Configuring sendmail.*
- *Maintaining sendmail.*
- *Troubleshooting sendmail.*

Chapter 20

IRIX sendmail

This chapter describes IRIX *sendmail*, a facility for routing mail across an internetwork. This chapter is for system administrators who set up and maintain the mail system on a station or network. It provides the information necessary for a straightforward implementation of *sendmail*. The following topics are covered:

- The overall mail system. See "The Mail System" on page 636 "An Overview of sendmail" on page 637 and "How sendmail Works" on page 640.
- The *aliases* database. See "How sendmail Works" on page 640.
- Network configurations for *sendmail*. See "sendmail Network Configurations" on page 650.
- User configurable macros. See "User Configurable Macros and Classes" on page 653.
- Planning your *sendmail* system. "A sendmail Planning Checklist" on page 655.
- Configuring your *sendmail* System. See "Configuring sendmail" on page 656.
- Managing your *sendmail* system. See "Managing sendmail" on page 669.
- Troubleshooting your *sendmail* system. See "Questions, Problems, and Troubleshooting" on page 675.

For sites already using *sendmail*, refer directly to "Notes to Current sendmail Users" on page 676.

The complete *IRIX sendmail Reference* appears as Appendix E of this guide.

The Mail System

The mail system is a group of programs that you can use to send messages to and receive messages from other users on the network. You can send mail through either UUCP or TCP/IP. The IRIX operating system uses Media Mail, System V */bin/mail,* 4.3BSD */usr/sbin/Mail,* and *sendmail* for its mail implementation.

The process of delivering mail involves four elements:

User Interface
: The user interface creates new messages and reads, removes, and/or archives received messages. Media Mail, System V */bin/mail*, and 4.3BSD */usr/sbin/Mail* are the user interfaces provided with IRIX. Reference pages are available to fully describe the features of these interfaces, and Media Mail has an extensive online help system.

Mail Routing
: A mail router examines each message and routes it through the network to the appropriate station. The *sendmail*(1M) program not only routes messages, but also formats them appropriately for their recipient stations.

Mail Transfer
: A mail transfer program transmits messages from one station to another. *sendmail* implements the Simple Mail Transfer Protocol (SMTP) over TCP/IP. For TCP/IP mail, *sendmail* acts as an integrated routing and transfer program. In all cases, mail transfer has a counterpart: mail reception. In most cases, a single program provides both functions. UUCP is a mail transfer program that uses its own protocols and runs over serial lines.

Mail Delivery
: A mail delivery program deposits mail into a data file for later perusal by a user or another program. The */bin/mail -d* program delivers local mail.

After you compose a message by using Media Mail, */bin/mail*, or */usr/sbin/Mail*, the message is sent to *sendmail,* which attempts to determine the destination of the message. *sendmail* either calls */bin/mail* (for mail to a user on the local station) or passes the message to the appropriate mail transfer program (for mail to a user on a remote station).

When *sendmail* receives a message from another station, it analyzes the recipient address; then, it either calls */bin/mail* to complete the delivery if the local station is acting as a relay, or *sendmail* passes the message to the mail transfer program. For TCP/IP SMTP, *sendmail* also performs the mail transfer.

When you send a mail message on a network that uses TCP/IP, several layers of network software are involved. Figure 20-1 shows the layers of TCP/IP mail network software.

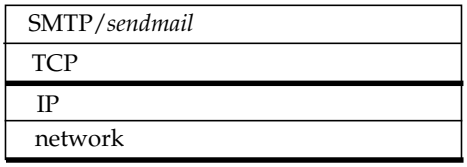


Figure 20-1 Layers of TCP/IP mail software

An Overview of sendmail

The Transmission Control Protocol (TCP) layer supports SMTP, which *sendmail* uses to transfer mail to other TCP/IP stations. *sendmail* is responsible for calling local delivery programs, mail routing, and TCP/IP mail transfer; it may also call other mail transfer programs. For example, *sendmail* uses the UUCP transmission program to handle messages sent to UUCP stations.

sendmail's implementation features aliasing, forwarding, automatic routing to network gateways, and flexible configuration.

In a simple network, each node has an address, and resources can be identified with a host-resource pair. For example, a mail system can refer to users with a host–user-name pair. Station names and numbers must be administered by a central authority, but user names can be assigned locally to each station.

In an internetwork, multiple networks with different characteristics and management must communicate. In particular, the syntax and semantics of resource identification change. You can handle certain simple cases by using improvised techniques, such as providing network names that appear local to stations on other networks. However, the general case is extremely complex. For example, some networks require point-to-point routing, which simplifies the database update problem, because only adjacent stations are entered into the system tables; others use end-to-end addressing. Some networks use a left-associative syntax; others use a right-associative syntax, causing ambiguity in mixed addresses.

Internetwork standards seek to eliminate these problems. Initially, these standards proposed expanding the address pairs to address triples, consisting of *network*, *station*, *resource*. Network numbers must be universally agreed upon; stations can be assigned locally on each network. The user-level presentation was quickly expanded to address domains, composed of a local resource identification and a hierarchical domain specification with a common static root, as defined in RFC 1034. The domain technique separates the issue of physical versus logical addressing. For example, an address of the form "jane@iris1.company.com" describes only the logical organization of the address space.

sendmail bridges the gap between the world of totally isolated networks that know nothing of each other and the clean, tightly coupled world of unique network numbers. *sendmail* can accept old arbitrary address syntaxes, resolving ambiguities by using heuristics specified by the network administrator, as well as domain-based addressing. *sendmail* helps guide the conversion of message formats between disparate networks. In short, *sendmail* is designed to assist a graceful transition to consistent internetwork addressing schemes.

System Organization

The design goals for *sendmail* included the following:

1. Message delivery should be reliable, guaranteeing that every message is correctly delivered or at least brought to the attention of a human for correct disposal; no message should ever be completely lost.
2. Existing software should be used to do actual message delivery whenever possible.
3. *sendmail* should be easy to expand to fairly complex environments.
4. Configuration should not be compiled into the code.
5. *sendmail* should let various groups maintain their own mailing lists, and let individuals specify their own forwarding, without modifying the station's alias file.
6. Each user should be able to specify which mailer to execute to process mail being delivered for him. This feature allows users with specialized mailers that use a different format to build their environments without changing the system, and facilitates specialized functions (such as returning an "I am on vacation" message).
7. To minimize network traffic, addresses should be batched to a single station where possible, without assistance from the user.

Figure 20-2 illustrates the *sendmail* system structure that is based on the original design goals for *sendmail.*

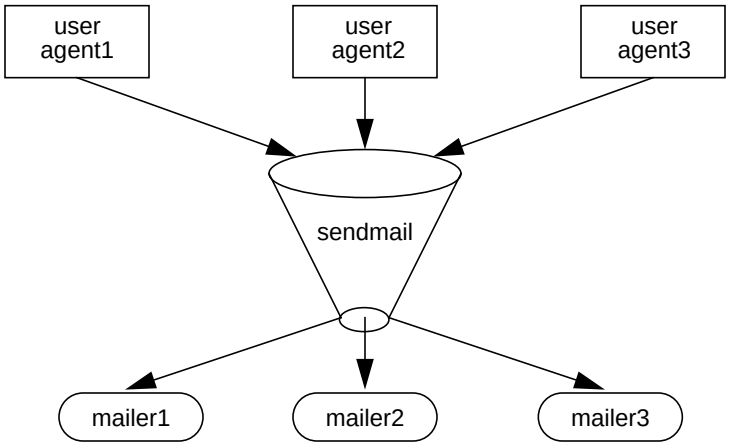


Figure 20-2 *sendmail* System Structure

sendmail neither interfaces with the user nor does actual mail delivery. Rather, it collects a message generated by a user agent program such as Berkeley *Mail*, edits the message as required by the destination network, and calls appropriate mailers to do mail delivery or queueing for network transmission. The exception is mail sent to a file; in this case, *sendmail* delivers the mail directly

This discipline allows the insertion of new mailers at minimum cost.

Because some of the senders may be network servers and some of the mailers may be network clients, *sendmail* can be used as an internetwork mail gateway.

How sendmail Works

Understanding the *sendmail* programs requires understanding a variety of components. Some of these components are daemons, scripts, files, and commands. This section describes the various *sendmail* components.

The sendmail Daemon

For *sendmail* to process incoming mail, a daemon must be running. The *sendmail* daemon is the *sendmail* program with specific flags. (Appendix E describes the *sendmail* command-line flags in detail.) The daemon is automatically started by the */etc/init.d/mail* script at station start-up. The default command for the *sendmail* daemon is:

```
/usr/lib/sendmail -bd -q15m
```

The **-bd** flag causes *sendmail* to run in daemon mode. The **-q15m** flag causes *sendmail* to fork a subdaemon for queue processing every fifteen minutes. The **-bd** and **-q** flags can be combined in one call.

sendmail Scripts

There are two scripts provided with your system that perform common functions in *sendmail*. Use these scripts whenever possible, as they have been tested and are known to perform the task correctly.

/etc/init.d/mail

Under rare circumstances, a user may need to stop or start the *sendmail* daemon manually. For example, to implement changes to the configuration file, you must stop all running *sendmail* processes, "refreeze" the configuration file, and restart the *sendmail* daemon before the new configuration will take effect. To simplify the task of starting and stopping *sendmail*, IRIX provides a shell script called */etc/init.d/mail*.

This script takes a single argument, either "**start**" or "**stop**," which starts or stops the *sendmail* daemon respectively. You must be superuser (root) to use this script. For example, to stop *sendmail*, use the following command:

```
/etc/init.d/mail stop
```

When */etc/init.d/mail* is called with the "start" argument, it verifies the existence and permissions of various *sendmail* related files and directories (see "sendmail Related Files and Directories" on page 642). If a required component such as the */var/spool/mqueue* directory is missing, the script creates it. For more complex components such as */etc/aliases*, the script exits with a message.

When the */etc/init.d/mail* script is called with the "stop" argument, it kills all running *sendmail* processes with a SIGTERM signal.

Note: Station start-up includes an automatic call to the */etc/init.d/mail* script with the `start` argument. If station start-up runs in "verbose" mode (that is, */etc/chkconfig* verbose on), the following message appears, verifying that *sendmail* has been started:

```
Mailer daemons: sendmail
```

For more information, examine the */etc/init.d/mail* script.

/usr/etc/configmail

The */usr/etc/configmail* script provides an interface between command line input and the *sendmail.cf* file. For more information, see "sendmail Related Files and Directories" on page 642 It pipes the macro and class definitions into the *sendmail.params* file . This script simplifies the *sendmail* configuration process.

The *configmail* script allows the user to interact with several *sendmail.cf* parameters. These parameters are equivalent to *sendmail.cf* macros and/or classes. You can verify the current parameter settings, set specific parameters, issue a quick setup command, and get some basic online help. *configmail* stores your changes in the *sendmail.params* file, which is read by *sendmail* at startup time.

sendmail Related Files and Directories

The *sendmail* configuration files and directories are:

- */etc/sendmail.cf*
- */etc/sendmail.fc*
- */etc/sendmail.hf*
- */etc/sendmail.st*
- */etc/aliases*
- */var/spool/mqueue*

- */var/mail*

/etc/sendmail.cf

At the heart of the *sendmail* program is the *sendmail* configuration file */etc/sendmail.cf*. The *sendmail.cf* file is an ASCII file that contains most of the configuration information and is read at run time. This file encodes options, header declarations, mailer declarations, trusted user declarations, message precedences, address-rewriting rules, macro definitions, and class definitions.

As the mail administrator and in order for you to successfully set up *sendmail*, you must know which *sendmail.cf* macros and variables to change. The *sendmail.cf* file takes advantage of *sendmail*'s ability to read macro and class definitions from pipes, thereby simplifying and automating the *sendmail* configuration process. This file takes command line input from the *sendmail.params* file and */usr/etc/configmail* script and incorporates the input into the appropriate macros and classes.

/etc/sendmail.fc

The *sendmail.fc* file is a frozen configuration file. A frozen configuration file is an image of the data space that belongs to *sendmail* when the configuration file is read. The *sendmail.fc* file is not present by default. You can create the *sendmail.fc* file using the *touch* (1) command. After the *sendmail.fc* file is created, it is used in place of */etc/sendmail.cf*. This process improves start-up speed.

Note: All modifications to *sendmail* macros and classes should be made to *sendmail.cf*.

However, if the */etc/sendmail.fc* file exists, changes to it are not honored until you rebuild */etc/sendmail.fc*. The mail script, */etc/init.d/mail* automatically rebuilds the frozen configuration file if the sendmail.cf file exists. *Always* use the mail script, */etc/init.d/mail*, as it will automatically rebuild the *sendmail.fc* file. If you need to manually rebuild the frozen configuration file, the command is:

```
/usr/lib/sendmail -bz
```

/etc/sendmail.hf

The *sendmail.hf* file is the SMTP help file. It contains some brief information about the various Simple Mail Transfer Protocol (SMTP) commands.

/etc/sendmail.st

The *sendmail.st* file is used to collect statistics related to *sendmail*. By default, the file is not present. You can create the file using the *touch*(1) command. If the file is present, *sendmail* automatically updates the file with relevant *sendmail* statistics.

/etc/aliases

The *aliases* file contains the text form of the alias database used by the *sendmail* program. The alias database contains aliases for local mail recipients. For example, the following alias delivers mail addressed to *jd* on the local station to *johndoe@company.com*:

jd:johndoe@company.com

When sendmail starts up, it automatically processes the aliases file into the files */etc/aliases.dir* and */etc/aliases.pag*. The *aliases.dir* and *aliases.pag* are DBM versions of the *aliases* database. The DBM format improves *sendmail* performance.

Note: The *newaliases* program must be run after modifying the alias database file. See "Building the Aliases Database" on page 647 for more information about building the alias database.

/var/spool/mqueue

The mail queue, */var/spool/mqueue*, is the directory in which the mail queue and temporary files reside. The messages are stored in various queue files that exist under the */var/spool/mqueue* directory. Queue files take the form of:

- qf* - control (queue) files for messages
- df* - data files
- tf* - temporary files

- nf* - a file used when a unique ID is created
- xf* - transcript file of the current session

Normally, a *sendmail* subdaemon processes the messages in this queue periodically, attempting to deliver each message. (The */etc/init.d/mail* script starts the *sendmail* daemon so that it will fork a subdaemon every 15 minutes to process the mail queue.) Each time *sendmail* processes the queue, it reads and sorts the queue, then attempts to run all jobs in order.

/var/mail

/var/mail is the directory that houses all incoming mail. Each user on a local station will receive his/her mail in a subdirectory of */var/mail*. For example, the user *guest* receives his/her mail in the directory */var/mail/guest*.

sendmail Commands

These section describes some of the related *sendmail* programs and commands. The programs and commands discussed in this section are:

- sendmail
- newaliases
- mailq

sendmail

sendmail is the program that implements *sendmail*'s routing and transfer service. As a program it has many flags that can be set on the command line to tailor the sendmail environment. Appendix D, "IRIX sendmail Reference," and "sendmail Command-Line Flags" on page 860 provide complete descriptions of the various command line flags and options.

/usr/bsd/newaliases

Program used to rebuild the DBM version of the *aliases* database. This program *must* be run anytime the text version of the *aliases* file is modified. If *newaliases* is not run after making changes to the *aliases* file, the changes are not incorporated into the DBM *alias* database and are not seen by the *sendmail* program. See "The Aliases Database" on page 646 for more details about the *alias* database.

/usr/bin/mailq

The command prints a current listing of the mail queue.

The Aliases Database

The aliases database is an *ndbm*(3B) database that contains mail aliases to be used by the *sendmail* program. The text form of the database is maintained in the file */etc/aliases.* The aliases are of this form:

name`:` *name*`1 [,` *name2*`, ...]`

For example, the following command delivers mail addressed to *jd* to *johndoe@company.com*:

jd:johndoe@company.com

Note: Only the local part of an address can be aliased. For example, the following command is wrong and will not have the desired effect:

```
jd@big.university.edu:jd@company.com
```

sendmail consults the alias database only after deciding that the message (as originally addressed) should be delivered locally, and after it has rewritten the address to contain only the local part.

An alias continuation line must start with a space or a tab. Blank lines and lines beginning with the number sign (#) are treated as comments.

If you are running NIS, *sendmail* can use the contents of the NIS alias database with the local *aliases* database by adding the following special alias to the */etc/aliases* file:

+:+

This special alias tells *sendmail* to consult the NIS alias database if the alias cannot be found in the local alias database. When the same alias is specified in both the local and NIS aliases file, the local alias supersedes the NIS alias.

Building the Aliases Database

At start-up *sendmail* automatically uses the *ndbm*(3B) library to process the */etc/aliases* file into the files */etc/aliases.dir* and */etc/aliases.pag*. Using these files to resolve alias is a technique that improves performance.

To rebuild the DBM version of the database without restarting *sendmail,* execute this command:

```
newaliases
```

Executing this command is equivalent to giving *sendmail* the **-bi** flag:

```
/usr/lib/sendmail -bi
```

When building the DBM version of the database, *sendmail* checks the left-hand side of each entry to make sure that it is a local address. *sendmail* issues a warning for each entry in */etc/aliases* with a non-local left-hand side. Such entries are not entered into the DBM version of the database.

If the NIS alias database is used with the local *usr/lib/aliases* database, the special "+:+" alias is entered into the DBM version of the database. If *sendmail* cannot find an alias in the DBM version of the database, it looks for the special "+:+" alias. If it finds the special alias, *sendmail* then queries the NIS alias database. This query permits you to change the global NIS alias database without having to rebuild the local alias database. However, the left-hand sides of the NIS alias are *not* checked by *sendmail* to ensure that they contain only local addresses.

If the configuration or the command line specifies the **D** option, *sendmail* will automatically try to rebuild the alias database when it is out of date.

sendmail rebuilds the alias database if either of the following conditions exists:

- The DBM version of the database is mode 666.
- *sendmail* is running *setuid* to root.

Auto-rebuild can be dangerous on heavily loaded stations with large alias files. If it takes more than five minutes to rebuild the database, there is a chance that several processes will start the rebuild process simultaneously.

Testing the Aliases Database

You can test the alias database with the -bv options of the sendmail program. See “sendmail Command-line Flags” on page 669 for more details.

Potential Problems

Problems can occur with the alias database, especially if a *sendmail* process accesses the DBM version before it is completely rebuilt. Two circumstances can cause this problem:

- One process accesses the database while another process is rebuilding it.
- The process rebuilding the database dies because it has been killed or a station crash has occurred before completing the rebuild.

sendmail has two techniques for trying to relieve these problems. First, to avoid the problem of a partially rebuilt database, *sendmail* ignores interrupts while rebuilding the database. Second, at the end of the rebuild it adds an alias of the following form (which is not normally legal):

```
@: @
```

Before *sendmail* accesses the database, it ensures that this entry exists. For this action to occur, the configuration file must contain the **-a** option.

If the @:@ entry does not exist, *sendmail* waits for it to appear. After the specified waiting period elapses, *sendmail* will force a rebuild itself. For this action to occur, the configuration file must include the **D** option. If the **D** option is not specified, a warning message is generated and *sendmail* continues.

Another alias problem can arise for stations incorporating the NIS alias database in */etc/aliases* through the use of the +:+ alias. If the NIS alias server goes down or is otherwise nonresponsive to NIS queries, *sendmail* will not see the aliases normally obtained from the NIS server. This situation may result in mail being returned, marked "`User unknown`."

List Owners

If an error occurs when mail is sent to a certain address (*x*, for example), *sendmail* looks for an alias of the following form to receive the errors:

```
owner-x
```

This scheme is typically useful for a mailing list where a user mailing to the list has no control over the maintenance of the list itself; in this case the list maintainer would be the owner of the list. For example, the following would cause *jd@1company.com* to get the error that occurs when someone sends mail to *unix-hackers* and *sendmail* finds the phony user *nosuchuser* on the list.

```
unix-hackers: jd@company1.com,
ed@big.university.edu,nosuchuser,

jane@company2.com
owner-unix-hackers: jd@company1.com
```

sendmail Network Configurations

This section explains the functions of domains, forwarders, and relays in a mail network. It also explains how each of these components is designated in the */usr/etc/configmail* script and in the *sendmail.cf* file. It is important to understand these designations, since you will be expected to enter this information in the working copy of these files for your station.

Mail Domains

Within the sendmail environment, a domain is an administratively-defined area of control with logical rather than physical boundaries.

You can configure three general types of domains:

- Root Domain: top level domain of the local domain space. For example, *horses.com* is the top level domain for the domain *pintos.horses.com*.
- Direct Domain: the domain(s) a station can send mail to directly (no forwarders or relays involved).
- Local Domain: the domain to which a station belongs.

The following parameters designate domains in the */usr/etc/configmail* script and the */etc/sendmail.cf* file:

configmail parameters: rootdomain, directdomain, and localdomain

sendmail.cf macros/classes:

- Root Domain: *T macro*
- Direct Domain: *D class*
- Local Domain: *D macro*

Mail Forwarders

A forwarder station is a station that acts as a mail gateway into another network. Typically, stations on either side of a gateway cannot connect directly to each other, making the forwarder station a physical (not just an administrative) necessity.

Forwarder stations are not necessarily "smarter" about mail routing than other stations in the network, but they are "better connected." Forwarder stations deliver mail to "all points beyond" some point in the domain name space.

The designation of the forwarder station is primarily determined by the physical topology of the network; The default *sendmail.cf* file can designate only a single forwarder; the name of that station must be hard-coded in the configuration file.

The following parameters designate a mail forwarder in the */usr/etc/configmail* script and the */etc/sendmail.cf* file:

configmail parameter: *forwarder*

sendmail.cf macro and class: *F*

Mail Relays

A relay station is a station that acts as a collection point for mail destined for a specified domain or group of domains. In the absence of *MX* records and mail exchangers, relay stations provide a mechanism whereby mail can be concentrated onto centralized locations prior to actual delivery. For more information about MX records and mail exchangers, see Appendix D, "IRIX sendmail Reference."

Relay stations are not necessarily "better connected" than other stations in the network, but they are "smarter" about mail routing. They deliver mail to "all points within" some point in the domain name space.

For example, a company with a domain *company.com*, has configured *sendmail* to treat *alpha.company.com* as the forwarder station and *omega.company.com* as the relay station. *sendmail* assumes that *alpha.company.com* is ultimately responsible for all mail to domains other than *company.com* and that station *omega.company.com* is ultimately responsible for all mail to the *company.com* domain itself. Note that there is nothing to prevent the relay and forwarder functions from residing on the same station.The designation of a relay station is primarily determined by administrative decision. *sendmail* can recognize a number of relay stations.

The relay station name is a special name used to identify relay stations in the network. This special name is defined by means of the *R* macro and is typically the name "relay." A relay station is so designated by being aliased to the name "relay." The default *sendmail.cf* file probes for a station named or aliased to the special relay station name and delivers mail to any such station in preference to the actual destination station. Mail is also sent to relay stations whenever the local station cannot determine the proper routing.

The following parameters designate a mail relays in the */usr/etc/configmail* script and the */etc/sendmail.cf* file:

configmail parameter: *relayname*

sendmail.cf macro: *R*

User Configurable Macros and Classes

The *sendmail.cf* file defines your mail network by assigning each element in the network a *macro* and/or *class* value. The default values in your distribution *sendmail.cf* file will not work without some modification.

Instead of modifying your *sendmail.cf* file directly, you can use the */usr/etc/configmail* script. It takes your input, saves it in *sendmail.params* and configures the appropriate macros and classes according to your *sendmail* environment.

(D)omain Name Macro and Class

The **D** macro defines the local domain name. Be sure that the macro contains the name of the domain in which this station resides. If domains are not used, you can leave this macro empty or comment it out.

- The **D** class explicitly lists all domains for which this station should send mail directly by means of the local area network mailer.
- Mail for domains listed in the **D** class is sent directly to the destination station if possible. No attempt is made to send the mail by means of a relay station.
- Mail for domains not listed in the **D** class is sent by means of a relay station associated with the destination station, if possible, rather than directly to the recipient station.
- If this station is a relay for a particular domain, you must enter that domain name in the **D** class. It is recommended that you always enter the domain name, regardless of whether the station is a relay or not.

(F)orwarder Station Name Macro and Class

The **F** macro defines the station name or alias of the station to which this station will forward mail for unknown stations or domains.

- The **F** class contains all known names for the station defined in the **F** macro.

- Mail is sent to the forwarder as a last resort only if one of these conditions exists:
  - This station cannot make the destination station name canonical or determine an appropriate relay or mail exchanger for the mail.
  - The appropriate station, relay, or exchanger for the mail exists outside the top-level domain. (See the description of the **T** macro later in this section.)
- If either condition for sending mail to this forwarder station exists, this station puts messages to unknown stations or domains "on the wire" and hopes for the best.
- If no such station exists in your environment, leave the **F** macro and class empty. If this station is the forwarder station, put this station's name in the **F** macro. Put all known names for the station in the **F** class.

(R)elay Station Name Macro

The **R** macro defines the station name (or an alias) used by all stations that act as relay stations. Relay stations are forwarders to known internal domains and are defined by the use of this relay station name as their station name or alias. This macro comes preconfigured as "*relay*," a name that is strongly suggested.

- Do not leave this macro blank, even if your network has no relay stations configured.
- Mail relay stations provide an alternative to an *MX* scheme, and can also be useful as an emergency backup to the use of *MX* records for internal mail routing.

(T)op-Level Domain Macro

The **T** macro defines the name of the top level of the local domain space. For example, if this station resides in a subdomain named *bar.foo.com* under the *foo.com* domain, and if all stations under the *foo.com* domain or any subdomain under the *foo.com* domain are considered to be internal stations, the **T** macro contains *foo.com.*

- The top-level domain is used with the forwarder station (**F**) macro and class described earlier in this section. All mail sent to stations outside the top level domain is sent by means of the forwarder station.

(K)illed Stations Class

The **K** class is a list of all known "killed" or "dead" stations in the local domain. This is only defined on mail forwarders to detect mail to stations that no longer exist. Any mail directed to a "dead" station is automatically sent to the mail forwarder.

(P)athalias Database Macro

The **P** macro defines the location of the pathalias database that is used by *sendmail* for UUCP mail routing.

A sendmail Planning Checklist

Here is a list of items to consider *prior* to configuring your *sendmail* environment.

- What is the layout of your sendmail network? (domains, forwarders, relays)
- If you are using the */usr/etc/configmail* script, do you have the values for the parameters you need to modify?
- If you are manually modifying the */etc/sendmail.cf* file, do you have the values for the macros and classes you need to modify?
- Are you setting up any custom sendmail aliases? If you are, have aliases ready for the aliases database file.

- Are the */etc/hosts* and */etc/passwd* files up to date? The files should include the special *relay* station name for the mail forwarder, any station aliases, and user accounts for mail users, etc.) If you are using domain names, the /etc/hosts file should use the following format:

  ip_address fully_qualified_domain_name alias1, alias2, ...

  A common problem with *sendmail* is that administrators place the aliases before the fully qualified domain name in the */etc/hosts* file.

- If you are using the Network Information Service (NIS) and/or BIND, are the sendmail-related files configured correctly (*aliases, hosts, passwd*)?

Configuring sendmail

Configuring *sendmail* involves these tasks:

1. Customizing the *sendmail.cf* file
2. Modifying the *aliases* file
3. Starting the *sendmail* daemon

This section provides an example for configuring a fictitious *sendmail* environment. The fictitious environment includes:

- a stand-alone station where users can send mail locally *(solitaire)*
- a simple isolated sendmail network with one domain *(lab.fictitious.com)*
- a hierarchical sendmail network with relays and one domain *(eng.fictitious.com)*
- a hierarchical sendmail network with relays and multiple domains *(corp.fictitious.com* and *fin.fictitious.com)*
- a complex hierarchical sendmail network with forwarders, relays, and multiple domains (*corp.fictitious.com* and *eng.fictitious.com*)
- a UUCP sendmail connection *(uk.com*)

Note: In the following examples, an *empty* macro or class has no values assigned to it. The macro or class is left blank.

Figure 20-3 illustrates the fictitious sendmail environment to be used for configuring *sendmail*.

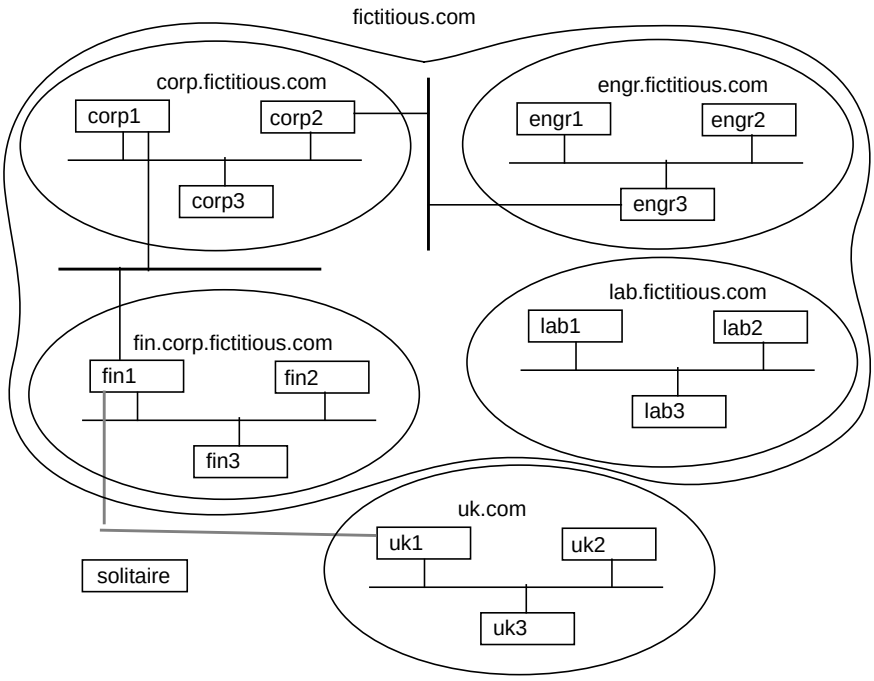


Figure 20-3 *sendmail* Configuration Environment (fictitious)

Customizing the sendmail.cf File

The configuration file describes mailers, tells *sendmail* how to parse addresses and rewrite message headers, and sets various *sendmail* options. The standard configuration file shipped with IRIX supports a wide variety of mail configurations and *does not* work "out of the box."

All examples are based on the default *sendmail.cf* configuration file as it is shipped with IRIX. Note that *sendmail* macros and classes are both case sensitive. See "User Configurable Macros and Classes" on page 653.

Stand-alone Station

In this case, there is a single, isolated station named *solitaire*. Mail is only sent from one user on the station to another user on the same station. No mail is sent to any other station, and no mail is received from any other station.

Using the *configmail* script, set up mail like this:

```
/usr/etc/configmail set directdomains NULL
/usr/etc/configmail set localdomain NULL
/usr/etc/configmail set forwarder NULL
/usr/etc/configmail set rootdomain NULL
```

If configuring the *sendmail.cf* file by hand, make the required adjustments to the following macros and classes:

The D macro and class:
Make sure that both the D macro and class are empty.

The F macro and class:
Make sure that both the F macro and class are empty.

The T macro: Make sure that the T macro is empty.

Note: If this is the only station you are configuring for sendmail, proceed to "Modifying the Aliases Database" on page 667.

Simple Isolated Network

This is the simplest network mail environment. A number of stations reside on a private network and send mail to each other on a peer-to-peer basis. All stations exist in the same domain; no subdomains exist. There is no connection or gateway between this private network and the outside world. No station in the network has greater responsibility for mail delivery than any other station. No relay or forwarder stations exist. The stations in the network are named *lab1, lab2,* and *lab3.* All stations exist under the *lab.fictitious.com* domain.

Each station in the network uses the same configuration. Using *configmail* on each station, make the changes shown below. For example, use *configmail* to

set up mail on station *lab1*:

```
/usr/etc/configmail set directdomains lab.fictitious.com
/usr/etc/configmail set localdomain lab.fictitious.com
/usr/etc/configmail set forwarder NULL
/usr/etc/configmail set rootdomain NULL
```

If configuring the *sendmail.cf* file by hand, make the required adjustments to the following macros and classes:

The D macro and class:
: Change the D macro and class to contain the lab.fictitious.com domain name.

The F macro and class:
: Make sure that both the F macro and class are empty.

The T macro: Make sure that the T macro is empty.

If you modified *sendmail.cf* by hand, you can copy the modified *sendmail.cf* file to all other stations in the *lab.fictitious.com* domain. When complete, proceed to "Modifying the Aliases Database" on page 667.

Hierarchical (Relay) Network with a Single Domain

In this example, all stations do not bear the same responsibility for mail delivery. One or more stations are designated as mail relay stations, where mail is concentrated for further processing or queueing before delivery.

This scheme has particular advantages if some stations frequently are powered off or are otherwise unable to communicate. In such a situation, one or more relay stations are "more reliable"; they are never or infrequently out of communication with the network and are designated as mail concentration points. When mail is sent to a station that is down, the mail travels to the relay station, where it is queued for later delivery, rather than being queued on the originating station. When the destination station returns to operation, it is more likely that the relay station will be up than the originating station. Therefore, the mail will be delivered to the destination station in a timely manner.

This hierarchical scheme also offers administrative advantages. For example, if a single station goes down for an extended period of time, or is simply failing to accept mail, the situation is easier to detect when there is a

central mail queue. An administrator can check the mail queue on the relay station to see which stations are not accepting mail. If there were no relay station, mail to the down station would be queued on stations throughout the network, and the problem could be harder to spot.

All stations exist under the *engr.fictitious.com* domain. The stations in the network are named *engr1, engr2,* and *engr3.* The mail relay station is *engr1.* The other stations in the network are expected to send mail through *engr1* rather than delivering it directly.

Each of the non-relay stations runs the same *sendmail.cf* configuration file. The *sendmail.cf* file on relay station *engr1* has a slightly different **D** class definition.

For example, using the *configmail* script, configure mail on the relay station *engr1*:

```
/usr/etc/configmail set directdomains engr.fictitious.com
/usr/etc/configmail set localdomain engr.fictitious.com
/usr/etc/configmail set forwarder NULL
/usr/etc/configmail set rootdomain engr.fictitious.com
```

Using the *configmail* script, set up mail on the remaining stations in the *engr.fictitious.com* domain. Note that the *directdomains* parameter is set to NULL on all stations except the relay station *engr1*.

```
/usr/etc/configmail set directdomains NULL
/usr/etc/configmail set localdomain engr.fictitious.com
/usr/etc/configmail set forwarder NULL
/usr/etc/configmail set rootdomain engr.fictitious.com
```

If configuring the *sendmail.cf* file by hand, make the required adjustments to the following macros and classes:

The **D** macro and class

On all stations, change the **D** macro to contain the *engr.fictitious.com* domain name.

On the relay station *engr1,* make sure that the **D** *class* contains the *engr.fictitious.com* domain name so that *engr1* sends mail directly to all stations in the *engr.fictitious.com* domain.

On the remaining stations, make sure that the **D** *class* is empty so that they *do not* send mail directly to other stations. (They are to send the mail to *engr1.*)

The **F** macro and class
: Make sure that the **F** macro and class are empty.

The **T** macro
: On all stations, change the **T** macro to contain the *engr.fictitious.com* domain name.

For *engr1* to be recognized as the mail relay station, the special relay station name "relay" (as defined by the **R** macro) must be one of the station alias that belongs to *engr1.* Include the station name "relay" in the entry for *engr1* in */etc/hosts* and/or the DNS and NIS equivalent.

When you have completed this exercise, proceed to "Modifying the Aliases Database" on page 667.

Hierarchical (Relay) Network with Multiple Domains

In this example, the hierarchical model is extended to multiple subdomains. This type of environment is a logical extension of the preceding one and is probably the easiest model to expand as the number of stations on the network increases. The environment requires that domain names be used for proper mail addressing.

The entire local domain is named *corp.fictitious.com.* There is one subdomain under the *corp.fictitious.com* domain: *fin.corp.fictitious.com.* The stations in the *corp.fictitious.com* domain are *corp1, corp2,* and *corp3.* The stations in the *fin.corp.fictitious.com* domain are *fin1, fin2,* and *fin3. corp3* is the relay for the *corp.fictitious.com* domain; *fin3* is the relay for the *fin.corp.fictitious.com* domain.

The stations in each of the two domains *(corp.fictitious.com* and *fin.corp.fictitious.com)* are configured much like those described in the preceding subsections.

Using the *configmail* script, set up mail on the relay station *corp3*:

```
/usr/etc/configmail set directdomains corp.fictitious.com
/usr/etc/configmail set localdomain corp.fictitious.com
/usr/etc/configmail set forwarder NULL
```

```
/usr/etc/configmail set rootdomain corp.fictitious.com
```

Using the *configmail* script, set up mail on the relay station *fin3*:

```
/usr/etc/configmail set directdomains \
fin.corp.fictitious.com
/usr/etc/configmail set localdomain fin.corp.fictitious.com
/usr/etc/configmail set forwarder NULL
/usr/etc/configmail set rootdomain corp.fictitious.com
```

Using the *configmail* script, set up mail on the remaining non-relay stations in the *corp.fictitious.com* domain. Note that the *directdomains* parameter is set to NULL on non-relay stations.

```
/usr/etc/configmail set directdomains NULL
/usr/etc/configmail set localdomain corp.fictitious.com
/usr/etc/configmail set forwarder NULL
/usr/etc/configmail set rootdomain corp.fictitious.com
```

Using the *configmail* script, set up mail on the remaining non-relay stations in the *fin.corp.fictitious.com* domain. Note that the *directdomains* parameter is set to NULL on non-relay stations.

```
/usr/etc/configmail set directdomains NULL
/usr/etc/configmail set localdomain fin.corp.fictitious.com
/usr/etc/configmail set forwarder NULL
/usr/etc/configmail set rootdomain corp.fictitious.com
```

If configuring the *sendmail.cf* file by hand, make the required adjustments to the following macros and classes:

The **D** macro — For all stations in the *corp.fictitious.com* domain, change the **D** macro to contain the *corp.fictitious.com* domain name.

For all stations in the *fin.corp.fictitious.com* domain, change the **D** macro to contain the *fin.corp.fictitious.com d*omain name.

The **D** class: — On the relay station *corp3,* make sure that the **D** class contains the *corp.fictitious.com* domain name so that *corp3* will send mail directly to all stations in the *corp.fictitious.com* domain.

On the relay station *fin3,* make sure that the **D** class contains the *fin.corp.fictitious.com d*omain name so that *fin3* will send mail directly to all stations in the *fin.corp.fictitious.com d*omain.

On the remaining stations in the network, make sure that the **D** class is empty so that they *do not* send mail directly to other stations.

The **F** macro and class:
Make sure that the **F** macro and class are empty.

The **T** macro: On each of the stations, change the **T** macro to contain the *corp.fictitious.com* domain name.

For the relay stations *corp3* and *fin3 t*o be recognized as such, the special relay station name "relay" (as defined by the **R** macro) must be an alias for each of them. There can only be one *relay* alias in the */etc/hosts* file. Here is how to set up each alias:

- For *corp3,* the alias *relay.corp.fictitious.com a*nd optionally "relay" should be included in its entry in */etc/hosts* and/or the DNS and NIS equivalent.
- For *fin3,* the alias *relay.fin.corp.fictitious.com s*hould be included in its entry in */etc/hosts* and/or the DNS and NIS equivalent.

When you have completed this procedure, proceed to "Modifying the Aliases Database" on page 667.

A Complex (Forwarder) Hierarchical (Relay) Network with Domains

This section explains how to configure a station to act as the forwarder station; a forwarder station can be added to any of the scenarios described in the preceding subsections. Please see "sendmail Network Configurations" on page 650 for an explanation of the forwarder station concept as used by IRIX *sendmail.*

This discussion applies to mail environments of all types. Whatever the form of your internal mail environment, whenever you want to use a mail gateway station between your internal mail network and the external world, a forwarder station is required. The *sendmail* configuration for using this

forwarder station is exactly the same for all stations on the internal side of the gateway.

The internal mail environment consists of the domain *corp.fictitious.com* and any or all domains under *corp.fictitious.com (fin.corp.fictitious.com).* All stations within the *corp.fictitious.com* domain are capable of communicating with each other. For example, there is no physical restriction to prevent station *fin1.fin.corp.fictitious.com* from sending mail to *corp1.corp.fictitious.com.*

Station *corp2.corp.fictitious.com* can connect to all stations within the *corp.fictitious.com* domain and can also connect to stations in other domains, such as *engr.fictitious.com.* Station *corp2.corp.fictitious.com is* therefore the forwarder station to all domains beyond the *corp.fictitious.com* domain. In this example, station *corp2.corp.fictitious.com* is aliased to *corp2* for ease of addressing in the internal mail environment.

In addition to the changes to *sendmail.cf* required for the internal mail environment, make the following changes to the *sendmail.cf* file on all stations within the *corp.fictitious.com* domain. Using the *configmail* script, change the appropriate parameter:

```
/usr/etc/configmail set forwarder \
             corp2.corp.fictitious.com corp2
```

If configuring the *sendmail.cf* file by hand, make the required adjustments to the following macros and classes:

The **F** macro: Make sure that the **F** macro contains the station name *corp2.corp.fictitious.com*.

The **F** class: Make sure that the **F** class contains the two names *corp2* and *corp2.corp.fictitious.com*, by which the forwarder station is known.

When you have completed the procedure, proceed to “Modifying the Aliases Database” on page 667.

UUCP Mail

The default *sendmail.cf* file shipped with IRIX includes support for sending mail through UUCP. This section discusses these capabilities and explains

how to integrate UUCP mail into the local mail environment. UUCP support can be added to any of the scenarios described in the preceding subsections.

The *sendmail.cf* file directs *sendmail* to read the */etc/uucp/Systems* file on start-up. All UUCP station names are read from this file, and stations marked "domain-machine" are noted. If a UUCP pathalias database is maintained on the station, the location of the database is set with the **P** macro.

If the */etc/uucp/Systems* file indicates that there are stations connected to the local station through UUCP, *sendmail* sends mail received on the local station, and addressed to one of the stations described in the */etc/uucp/Systems* file, on to the proper place. If the **P** macro is set and points to a valid UUCP pathalias database, *sendmail* will attempt to find a UUCP path to a station for which it cannot find an address or *MX* record. If the database returns a good UUCP path to the destination station, *sendmail* attempts to send the mail to the left-most station on the path.

Depending upon the network environment, UUCP mail may range from the only form of network mail to one part of a much larger network mail environment. The following example shows a common technique for adding UUCP to an existing local area mail network.

Sample Environment

The local domain is named *uk.com.* Station *uk1.uk.com is the* forwarder station, as described in "A Complex (Forwarder) Hierarchical (Relay) Network with Domains" on page 663 in "Customizing the sendmail.cf File" on page 657.

To avoid forwarder loops, the default *sendmail.cf* file permits only one forwarder station to be configured. Therefore, station *uk1.uk.com* is also the UUCP forwarder station.

Changes to sendmail.cf

No changes to *sendmail.cf* are necessary beyond those required to configure station *uk1.uk.com* as the forwarder station. (See "A Complex (Forwarder) Hierarchical (Relay) Network with Domains" on page 663 in "Customizing the sendmail.cf File" on page 657) If station *uk1.uk.com* maintains a pathalias database, the **P** macro should be set to the pathname of the pathalias database.

Other Changes

The */etc/uucp/Systems* file must also be configured before *sendmail* will see any of the UUCP-connected stations. For more information see Chapter 21, "UUCP."

When you have completed this procedure, look ahead to "Modifying the Aliases Database" on page 667.

Non-Domain Addressing

This section discusses issues related to a mail network that does not use domain addressing. Note that all of the previously discussed mail environments, with the exception of the "hierarchical multi-domain" environment, are possible in a network that does not implement domains.

If a network does not use domain addressing, specific changes are required in the *sendmail.cf* file on all stations in the network. First, make changes to the *sendmail.cf* file as described in the appropriate examples in this section. Next, make the following changes, even if they replace changes you have just made.

Using the *configmail* script, set up mail on each station on the network.

```
/usr/etc/configmail set directdomains NULL
/usr/etc/configmail set localdomain NULL
/usr/etc/configmail set rootdomain NULL
```

If configuring the *sendmail.cf* file by hand, make the required adjustments to the following macros and classes:

The **D** macro and class
: Make sure that the **D** macro and class are both empty.

The **T** macro
: Make sure that the **T** macro is empty.

In an isolated network, you must create the file */etc/resolv.conf* and add the line:

```
hostresorder local bind yp
```

For more information about the *resolv.conf* file, see the *resolv.conf*(4) reference page.

When you have completed this procedure, proceed to "Modifying the Aliases Database" on page 667.

Modifying the Aliases Database

After modifying a station's *sendmail.cf* file (see "Customizing the sendmail.cf File" on page 657) the alias database file should also be modified to reflect your *sendmail* environment. If you don't have any "private" company aliases, you still need to modify the *aliases* file to provide it with a valid *postmaster* alias.

Creating the Aliases File

Continuing with the fictitious example used in "Customizing the sendmail.cf File" on page 657 we will assume that the administrator for the domain *corp.fictitious.com* has derived a list of aliases for the *corp.fictitious.com doma*in. The list of aliases is shown in Table 18-1.

Table 20-1 Sample *aliases* File Entries

Alias Name	Member	Station Name
finance	john	fin1
finance	paul	fin2
finance	mary	fin3
corp	sharon	corp1
corp	pam	corp2
corp	peter	corp3
all	finance and corp	N/A
postmaster	mailmgr	corp1

The */etc/aliases* file entries based on the list of aliases generated by the administrator would look like this:

```
##################################################################
 # Aliases in this file will NOT be expanded in the header
# from Mail, but WILL be visible over networks or from
# /bin/mail.
# >>>>>>>>> The program "newaliases" must be run after
# >> NOTE >> this file is updated for any changes to
# >>>>>>>>>> show through to sendmail.
##################################################################
start of common aliases--do not remove this line
# Add the following alias to enable Yellow Page aliases. If
# enabled, the YP database defines anything not defined in
# this file.
#+:+
# Alias for mailer daemon
MAILER-DAEMON:postmaster
# send mail likely to be lost to the mail server
rootcsh:postmaster
rootsh:postmaster
.
.
.
games:postmaster
# Following alias is required by RFC 822
# You should change 'root' in the first line below to
# the administrator of this machine, and un-comment the
# following line.
postmaster:root
root:mailmgr@corp1
# aliases to handle mail to msgs and news
nobody: /dev/null
# end of common aliases--do not remove this line
# corp.fictitious.com aliases
finance:john@fin1,paul@fin2,mary@fin3
corp:sharon@corp1,pam@corp2,peter@corp3
all:finance,corp
```

Updating the aliases Database File

Anytime you modify the */etc/aliases* text database file, you *must* run the *newaliases* program. This program incorporates the text changes into the DBM files, */etc/aliases.dir* and */etc/aliases.pag*.

Update the aliases database file:

```
/usr/bsd/newaliases
```

If there is nothing wrong with your aliases database, *newaliases* should list the number of aliases and then return your prompt. If you see any other message, most likely there is a problem with your aliases file. See "Debugging Flags" on page 671 for hints on troubleshooting the alias file.

Starting the sendmail Daemon

After customizing the *sendmail.cf* files and modifying the *aliases* database, you are ready to start *sendmail*.

By default, IRIX automatically starts *sendmail* at station start-up by using the shell script */etc/init.d/mail.* However, if you are configuring and testing *sendmail* and don't want to reboot the station, you can run the */etc/init.d/mail* script manually. You should always use the *mail* script to stop and start *sendmail*. It processes and checks *sendmai*l related files and programs and in the correct order.

Start the *sendmail* daemon:

/etc/init.d/mail start

If you need to stop *sendmail*, enter the following command:

/etc/init.d/mail stop

Managing sendmail

This section describes some of the tasks related to managing the *sendmail* environment.

sendmail Command-line Flags

You can include one or more flags on the command line to tailor a *sendmail* session. This section describes some of the more frequently used flags. For a

complete description of command-line flags, Appendix D, "IRIX sendmail Reference."

Changing the Values of Configuration Options

The **-o** flag overrides an option in the configuration file. The override is for the current session only. In the following example, the **T** (timeout) option becomes two minutes for this session only:

```
/usr/lib/sendmail -oT2m
```

For a complete discussion of configuration options, see Appendix D, "IRIX sendmail Reference."

Delivery Mode

One configuration option frequently overridden on the command line is the **d** option, which specifies the *sendmail* delivery mode. The delivery mode determines how quickly mail is delivered:

i	deliver interactively (synchronously)
b	deliver in background (asynchronously)
q	queue only (don't deliver)

There are trade-offs. Mode **i** passes the maximum amount of information to the sender, but is rarely necessary.

Mode **q** puts the minimum load on your station, but if you use it, delivery may be delayed for up to the queue interval.

Mode **b** is probably a good compromise. However, in this mode, *sendmail* may initiate a large number of processes if you have a mailer that takes a long time to deliver a message.

Queue Mode

The **-q** flag causes *sendmail* to process the mail queue at regular intervals. The syntax is as follows, where *time* defines the interval between instances of queue processing:

-q [*time*]

Time is expressed in number of minutes: 15m sets the interval to 15 minutes. If *time* is omitted, *sendmail* processes the queue once and returns. The **-q** flag is often used in conjunction with daemon mode, described in the next subsection.

Daemon Mode

To process incoming mail over sockets, a daemon must be running. The **-bd** flag causes *sendmail* to run in daemon mode. The **-bd** and **-q** flags can be combined in one call, as in the following example:

```
/usr/lib/sendmail -bd -q30m
```

This command causes *sendmail* to run in daemon mode and to fork a subdaemon for queue processing every half hour.

The script for starting *sendmail* that is provided with IRIX includes the following command line:

```
/usr/lib/sendmail -bd -q15m
```

Verify Mode

Using the **-bv** flag directs *sendmail* to validate addresses, aliases, and mailing lists. In this mode *sendmail* performs verification only. It does not try to collect or deliver a message. *sendmail* expands all aliases, suppresses duplicates, and displays the expanded list of names. For each name, *sendmail* indicates if it knows how to deliver a message to that destination.

Test Mode

The **-bt** flag places *sendmail* in test mode so that it describes how the current configuration rewrites addresses. Test mode is extremely useful for debugging modifications to the */etc/sendmail.cf* configuration file. For more information, see Appendix D, “IRIX sendmail Reference.”

Debugging Flags

Several debugging flags are built into *sendmail.* Each flag includes a number and a level. The number identifies the debugging flag. The level, which

defaults to 1, dictates how much information is printed. A low level causes minimal information to print; a high level causes more comprehensive information to print. By convention, levels greater than 9 are not recommended, since so much information prints that it is of limited value. Debugging flags use the following syntax:

-d *debug-list*

- Set flag 13 to level 1.

  -d13

- Set flag 13 to level 3.

  -d13 3

- Set flags 5 though 18 to level 1.

  -d5-18

- Set flags 5 through 18 to level 4.

  -d5-18 4

Many debugging flags are of little use to the average *sendmail* user. Some are occasionally useful for helping to track down obscure problems. Appendix D, "IRIX sendmail Reference," includes a complete list of debugging flags.

Using a Different Configuration File

The **-C** flag directs *sendmail* to use an alternate configuration file. For example, the following line directs *sendmail* to use the *test.cf* file instead of the default */etc/sendmail.cf* file:

```
/usr/lib/sendmail -Ctest.cf
```

If the **-C** flag appears without a file name, *sendmail* uses the file *sendmail.cf* in the current directory. Thus, the **-C** flag directs *sendmail* to ignore any */etc/sendmail.fc* ("frozen") file that may be present.

The Mail Queue

This section discusses how to print and force the mail queue.

Listing the Queue

You can list the contents of the queue by using the *mailq* command or by specifying the **-bp** flag to *sendmail*. The list includes a listing of the queue IDs, the size of each message, the date the message entered the queue, and the sender and recipients.

Forcing the Queue

The **-q** flag (with no value) forces *sendmail* to process the queue. It is sometimes useful to use the **-v** flag (verbose) also when running the queue manually, as follows:

/usr/lib/sendmail -q -v

In verbose mode, *sendmail* displays the SMTP chatter with other stations as well as messages indicating any delivery errors and final message disposition.

Because of the locking algorithm, it is impossible for one job to freeze the queue. However, an uncooperative recipient station or a program recipient that never returns can consume many station resources. Unfortunately, there is no way to resolve this situation without violating the SMTP protocol used by *sendmail*.

In some cases, if a major station goes down for a couple of days, a prohibitively large queue may be created. As a result, *sendmail* will spend an inordinate amount of time sorting the queue. You can remedy this situation by moving the queue to a temporary location and creating a new queue. The old queue can be run later when the offending station returns to service.

Use the following commands to move the entire queue directory. The mail queue should be owned by *root* and belong to the *mail* group.

```
cd /var/spool
```

```
mv mqueue omqueue

mkdir mqueue

chmod 755 mqueue
```

Then kill the existing *sendmail* daemon (because it will still be processing in the old queue directory) and create a new daemon:

```
/etc/init.d/mail stop

/etc/init.d/mail start
```

To run the old mail queue, use the following command:

```
/usr/lib/sendmail -oQ/var/spool/omqueue -q
```

The **-oQ** flag specifies an alternate queue directory and the **-q** flag causes *sendmail* to run every job in the queue once and then return. Use the **-v** (verbose) flag to watch what is going on. It may be necessary to run the old mail queue a number of times before all of the messages can be delivered.

When the queue is finally emptied, the directory can be removed:

```
rmdir /var/spool/omqueue
```

The .forward File

As an alternative to the alias database, a user can put a file with the name *.forward* in his home directory. If the *.forward* file exists, *sendmail* redirects mail for that user to the list of recipients in the file. The recipients are separated by commas or new lines. For example, if the home directory for user *jane* has a *.forward* file with the following contents, any mail arriving for *jane* is redirected to the specified accounts:

```
zippy@state.edu
bongo@widgets.com
```

The *.forward* file also allows the user to redirect mail to files or programs. A *.forward* file with the following contents will redirect any incoming messages

to *jd@company.com*, append a copy of the message to the file */var/tmp/mail.log*, and pipe a copy of the message to *stdin* of the */usr/bin/mymailer* program:

```
jd@company.com
/var/tmp/mail.log
| /usr/bin/mymailer
```

In general, file-type recipients must be writable by everyone. However, if *sendmail* is running as *root* and the file has *setuid* or *setgid* bits set, then the message will be written to the file.

Users can redirect mail to themselves in addition to sending it to other recipients. This feature is particularly useful if the users want to continue to receive mail in their own mailboxes while passing copies of each incoming message to some alternative destination. For example, say that the home directory for user *john* contains a *.forward* file with the following contents:

```
\john, |/usr/sbin/vacation
```

sendmail will behave as follows:

- It will send each incoming message to john's regular mailbox; the backslash (\) preceding the name indicates that no further aliasing is to occur.
- It will pipe a copy of each message to *stdin* of the */usr/sbin/vacation* program. (The vertical bar [|] is the standard UNIX pipe symbol.)

Questions, Problems, and Troubleshooting

1. What are MX records?

   MX records are resource records in the BIND database. Each record contains the name of a target station, a preference level, and the name of an exchanger station that handles mail for the target station. (The exchanger station may be the target station itself.)

   The BIND database can contain several MX records for each target station; the record with the lowest preference level is tried first.

MX records provide a way to direct mail to alternative stations. Using MX records lets you eliminate static routes from your *sendmail* configuration file. For more details about setting up MX records, see Appendix D, "IRIX sendmail Reference."

2. *sendmail* doesn't seem to see my changes to *sendmail.cf.*

   Don't forget to save your changes to the configuration file by issuing the following commands:

   ```
   /etc/init.d/mail stop
   ```

   ```
   /etc/init.d/mail start
   ```

   The stopping and starting of the daemon forces *sendmail* to reconfigure the file. Otherwise, *sendmail* will run using the old, "frozen" version of the configuration file, thus ignoring your changes. For more information on "frozen" configuration files, see Appendix D, "IRIX sendmail Reference."

3. Can I put comments in macro definitions?

   Do *not* include comments on macro or class definition lines in the *sendmail.cf* file. For example,

   ```
   DDfoo.com # my domain name
   ```

   would define the **D** macro as:

   ```
   "foo.com # my domain name"
   ```

   Likewise,

   ```
   CD foo.com bar.com # my local domains
   ```

   would define the **D** class as containing "foo.com," "bar.com," "#," "my," "local," and "domains."

4. Where can I find information to help me troubleshoot my *sendmail* installation?

   See Appendix D, "IRIX sendmail Reference."

Notes to Current sendmail Users

In general, the current version of *sendmail* should be backward-compatible with previous versions. Most users should be able to replace their existing *sendmail* with the new IRIX 4.0 *sendmail* and run as before without any

changes. However, there are certain differences between this and previous versions of *sendmail* that may cause compatibility problems. These differences are described in the subsections that follow.

MX Record Support

For each destination station contacted by means of an IPC-type mailer, (P=[IPC] in the mailer definition line), *sendmail* will query the DNS database for *MX* records associated with the destination station. If the *MX* query succeeds, mail will be routed through the appropriate exchanger station found among the returned *MX* records as described in RFC 974 and required by RFC 1123.

The result is that mail to stations for which *MX* records are available may be routed differently by this version of *sendmail* than was done by previous versions. See Appendix D, "IRIX sendmail Reference," for information regarding mailer definitions.

With the advent of *MX* records, you may want to edit your *sendmail.cf* file to remove previously required static routes.

Multi-Token Class Match

Some *sendmail.cf* implementations inadvertently rely on the inability of *sendmail* to do multi-token class-matching. One such implementation is the standard *sendmail.cf* file distributed with IRIX Releases 3.2 and 3.3. If your *sendmail.cf* file is based upon one of those standard *sendmail.cf* files, you should read this section. If your *sendmail.cf* file is *not* based on one of those standard files, this section may serve as an example if you encounter odd behavior with class-matching while running the new *sendmail.*

The standard IRIX Release 3.2 and 3.3 *sendmail.cf* files define an **S** class that is scanned in from the */etc/hosts* file. This class is used to detect single-token station names that appear in the local */etc/hosts* file. With the advent of multi-token class-matching, the **S** class no longer operates as intended.

The problem is that station names appearing in the */etc/hosts* file are scanned into the **S** class whether they are single- or multi-token station names (that

is, whether or not they contain dots.) The **S** class still worked as intended with previous versions of *sendmail,* because if an attempt was made to match a multi-token station name in the class, the match would always fail. With the new *sendmail,* that same match will (incorrectly) succeed. This problem is observed when rules such as this one began matching qualified station names such as *foo.bar.sgi.com*:

```
# Assume that unqualified names are local.
R$*<@$=S>$* $1<@$2.$D>$3
```

The result was that stations such as *foo.bar.sgi.com* that appeared on the LHS of the rewrite rule shown here were being rewritten to *foo.bar.sgi.com.bar.sgi.com*, which is obviously wrong.

The problem was not the use of the **S** class, but rather the practice of scanning multi-token station names from the */etc/hosts* file into the class in the first place.

An examination of the *sendmail.cf* file shows that the **S** class is being scanned in by the following scan sets:

```
# Directly-connected SMTP hosts
FS/etc/hosts %*[.0-99] %[-_.a-zzA-ZZ0-99]
FS/etc/hosts %*[.0-99] %*[-._a-zzA-ZZ0-99] %[-_.a-zzA-Z0-99]
```

These scan sets read in the two left-most station names from */etc/hosts* regardless of whether they contain dots. To correct the situation, modify the scan sets to read in only the station names from */etc/hosts* in their single-token, unqualified form, as follows:

```
# Directly-connected SMTP hosts
FS/etc/hosts %*[.0-99] %[-_a-zzA-ZZ0-99]
FS/etc/hosts %*[.0-99] %*[-._a-zzA-ZZ0-99] %[-_a-zzA-ZZ0-99]
```

Note the removal of the dots from the right-most patterns.

Depending on your use of class-matching, this incompatibility may not affect you. If you suspect there might be a problem, you should examine your use of classes and your class definitions. If you are currently using a *sendmail.cf* file supplied by Silicon Graphics, you should examine the **S** class scan sets and make the corrections indicated here. If you use the default *sendmail.cf* as supplied in this release, you should be free from any such problems.

Chapter 21

UUCP

Chapter 21 discusses UUCP, the UNIX to UNIX Copy Program. This is probably the most widely used UNIX networking software. Other sites may support various networking protocols, but almost all sites (some not even UNIX-based) support UUCP. Topics covered in this chapter include:

- *Choosing TCP/IP or UUCP.*
- *Hardware and UUCP.*
- *UUCP commands.*
- *Directions for setting up UUCP.*

Chapter 21

UUCP

UUCP stands for "UNIX to UNIX copy program,'' and is a set of utilities that lets computers using versions of the UNIX operating system (such as IRIX) 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 simple encoding and decoding (*uuencode* and *uudecode*), to those used for remote login and command execution (*cu* and *uux*). The following topics are covered in this chapter:

- How to decide which networking software to use. See "Choosing TCP/IP or UUCP" on page 682.
- What hardware is necessary to use UUCP. See "Networking Hardware" on page 683.
- A list of UUCP commands. See "UUCP Commands" on page 684.
- A list of UUCP daemons. See "UUCP Daemons" on page 686.
- The UUCP supporting database. See "Supporting Databases" on page 687.
- A list of UUCP administrative files. See "UUCP Administrative Files" on page 710.
- Directions for setting up UUCP. See "Setting Up UUCP" on page 712.
- Directions for setting up UUCP over a TCP/IP network connection. See "Setting up UUCP on a TCP/IP Connection" on page 722.
- A list of UUCP Error Messages. See "UUCP Error Messages" on page 726.

The UUCP system is contained in the *eoe2* subsystem of your IRIX distribution, in the package called *eoe2.sw.uucp*. Use the *versions*(1M) command to determine if you have this subsystem installed.

UUCP connections using telephone lines and modems are used to distribute electronic mail and "net news" among thousands of computers in the USENET network.

As an administrator, you need to be familiar with the administrative tools, logs, and database files used by UUCP. This chapter provides details about the UUCP files, directories, daemons, and commands.

Choosing TCP/IP or UUCP

This section compares UUCP and the TCP/IP protocol suite for various purposes. You can use them both together; each for the tasks for which it is best suited. Both UUCP and TCP/IP software are standard features of the IRIX operating system. To use the TCP/IP software, you must have one of these communications mechanisms:

- a connection to an Ethernet network
- the optional FDDI hardware and software
- the Serial Line Internet Protocol (SLIP) software

To use UUCP, you must be connected to a serial network or to any TCP/IP network.

TCP/IP provides reliable interactive and batch services. UUCP is a batch-mode service; when you issue a *uucp* command, it is placed in a queue with other commands. The system checks the queue at regular intervals and executes the commands that it finds. After your command is carried out, UUCP reports the results of the command. The time it takes to carry out a command on a remote station varies on different stations.

Table 21-1 shows a comparison of features of TCP/IP and UUCP.

Table 21-1 Comparison of TCP/IP and UUCP

TCP/IP Features	UUCP Features
runs on Ethernet and FDDI, and over serial lines	runs over serial lines or over TCP/IP links
transfers files interactively	transfers files in batch mode

Table 21-1 (continued) Comparison of TCP/IP and UUCP

TCP/IP Features	UUCP Features
executes commands on remote stations interactively	executes commands on remote stations in batch mode
sends mail interactively or in batch mode	sends mail in batch mode
starts a shell on a remote station	starts a shell on a remote station
provides remote login facilities with *rlogin/telnet*	provides remote login facilities with *cu*
transfers data to any station running TCP/IP	transfers data to any station running UUCP

Networking Hardware

Before your computer can communicate with other computers through UUCP, 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.

Note: Refer to Chapter 10, "Terminals and Modems," and the *Personal System Administration Guide* for information on setting up modems and other hardware.

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 if 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 38,400 bits per second (bps) are possible when computers are directly linked. Such direct links are now rarely used because local area networks provide faster, easier-to-use connections.

Telephone lines
: Using a modem capable of dialing telephone numbers, your computer can communicate with other computers over standard phone lines. The modem dials the telephone

number requested by the networking utilities. The computer it is trying to contact must have a modem capable of answering incoming calls.

UUCP Commands

UUCP programs can be divided into two categories: user programs and administrative programs. The subsections that follow describe the programs in each category.

UUCP User Programs

The UUCP user programs are in */usr/bin*. No special permission is needed to use these programs; they are all described in the online reference pages.

cu — Connects your computer to a remote computer so you can log in to that computer, allowing you to transfer some files or execute commands on either computer without dropping the initial link.

uucp — Lets you copy a file from one computer to another. This program 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 (*/var/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 sends mail to the remote user who requested that it be picked up with *uupick*.

uupick — Retrieves the files placed under */var/spool/uucppublic/receive* when files are transferred to a computer that is 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's execution.

uustat Displays the status of requested transfers (*uucp*, *uuto*, or *uux*). This program also provides a way to control queued transfers.

UUCP Administrative Programs

Most of the administrative programs are in */usr/lib/uucp*, though the UUCP database files reside in */etc/uucp.* The only exception is *uulog,* which is in */usr/bin.* These commands are described in their respective reference pages.

You should use the *uucp* login ID when you administer UUCP because it owns the basic networking and spooled data files. The home directory of the *uucp* login ID is */usr/lib/uucp.* The other UUCP login ID is *nuucp*, used by remote computers that do not have their own login IDs to access your computer. A computer that logs in with *nuucp* gets *uucico* as its shell.

The following programs are the administrative utilities of UUCP:

uulog Displays the contents of a specified computer's log files. A log file is created for each remote computer with which your computer communicates. The log files contain records of each use of *uucp*, *uuto*, and *uux*.

uucleanup Cleans up the spool directory. This command 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. This command invokes the *uucico* daemon, in debug mode, to establish a communications link between your computer and the remote computer that you specify.

uucheck Checks for the presence of UUCP directories, programs, and support files. This program can also check certain parts of the Permissions file for obvious syntactic errors.

genperm Generates the Permissions file for stations that assign each remote station its own login ID.

UUCP Daemons

There are several daemons in UUCP. 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. It also starts *uuxqt* to execute any requested commands. 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 the *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. This daemon 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 are called. *uusched* is executed by a shell script called *uudemon.hour*, which is started by *cron*.

The following three programs are also used:

uugetty — This program is very similar to the *getty*(1M) program except that it permits a line (port) to be used in both directions. *uugetty*(1M) will be assigned to a port in the */etc/inittab* file if you want a port to be bi-directional. *uugetty* is executed as a function of the *init* program and is described in the IRIX Reference Pages.

fix-dsi	This program initializes dsi® modems
fix-intel	This program initializes Intel® modems.
fix-telebit	This program initializes Telebit® modems.
fix-hayes	This program initializes the Hayes® Smartmodem series.

Supporting Databases

The UUCP support files are in the */etc/uucp* directory.

Devices	Contains information concerning the location and line speed of the automatic call units (modems) and direct links.
Dialers	Contains character strings required to negotiate with automatic call units (ACUs) or modems in establishing connections to remote computers.
Systems	Contains information needed by the *uucico* daemon and the *cu* program to establish a link to a remote computer. This file 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, the telephone number, the login ID, and the password.
Dialcodes	Contains dial-code abbreviations that can be used in the phone number field of Systems file entries.
Permissions	Defines the level of access that is granted to remote users using *uucp* or *uux* when they attempt to transfer files or remotely execute commands on your computer.
Poll	Defines computers that are to be polled by your station and when they are polled.
Sysfiles	Assigns different or multiple files to be used by *uucico* and *cu*, such as *Systems*, *Devices*, and *Dialers* files.

The subsections that follow provide details on the structure of these files so you can edit them.

There are several other files that can be considered part of the supporting database; these files are not directly related to the process of establishing a link and transferring files. The files, *Maxuuxqts*, *Maxuuscheds*, and *remote.unknown*, are described briefly in "Other UUCP Files" on page 709.

The Devices File

The *Devices* file (*/etc/uucp/Devices*) contains information for all the devices that can be used to establish a link to a remote computer, such as automatic call units, direct links, and network connections.

Note: This file works interdependently 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
```

Entries for use with modems should always have the form:

```
Name device null speed 212 x dialer
```

Entries for use over TCP/IP network connections have the form:

```
TCP - - Any TCP uucp
```

Devices file fields are defined in the following sections

The Type Field

The keyword used in the Type field is matched against the third field of *Systems* file entries. The Type field can contain one of these keywords: Direct, ACU, or a station 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).

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.

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.

The Line Field

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 is attached to the */dev/ttyf5* line, the name entered in this field is *ttyf5*.

You should always use the *ttyf* devices when working with modems. These devices support hardware flow control, which is used by all modems that support V.32 or V.32*bis*.

The Line2 Field

If the keyword ACU is used in the Type field and the ACU is an 801-type dialer, the Line2 field contains the device name of the 801 dialer. (801-type ACUs do not contain a modem. Therefore, a separate modem is required and must be connected to a different line, defined in the Line field.) The need for a separate modem line 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 (-) or the word "null" as a place holder. A place holder is necessary for most modems.

The Class Field

The keyword used in the Class field of the Devices file is matched against the fourth field of Systems file entries:

```
Devices: ACU ttyf5 null D9600 212 x telebit

Systems: eagle Any ACU D9600 14155551212 login:nuucp
password:Oakgrass
```

Some devices can be used at any speed, so the keyword "Any" can 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 9600 bps. If the keyword *ACU* or *Direct* is used in the Type field, the Class field might contain only the speed of the device. However, the speed can also be preceded by a letter (for example, *C9600*, *D9600*) to differentiate between classes of dialers (Centrex or Dimension PBX). Including the dialer class is necessary in larger offices that have more than one type of telephone network: One network may be dedicated to serving only internal communications while another handles external. In such a case, it becomes necessary to distinguish which line(s) should be used for internal and which for external.

The Dialer-Token-Pairs Field

This field contains pairs of dialers and tokens. The Dialer portion may be the name of an automatic-dial modem, or "Direct" for a direct-link device. You can have any number of Dialer-Token-Pair (DTP) fields. The Token portion may be supplied immediately following the Dialer portion; if not present, the Token portion will be taken from a related entry in the Systems file.

This field has the format:

```
dialer   token   dialer   token
```

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.

The *DTP* field can be structured in different ways, depending on the device associated with the entry.

If an automatic-dial modem is connected directly to a port on your computer, the *DTP* field of the associated *Devices* file entry will only have one pair. This pair would normally be the name of the modem. This name is used to match the particular *Devices* 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 ttyf2 null 9600 212 x telebit
Dialers: telebit =&-% "" \r\p\r\c $ <K\T%%\r>\c ONLINE!
```

Notice that only the *Dialer* portion (*telebit*) 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)

If a direct link is established to a particular computer, the *DTP* field of the associated entry contains the keyword *Direct*. This is true for both types of direct-link entries, *Direct* and *System-Name*.

If an automatic-dial modem is connected to a switch, your computer must first access the switch; the switch then makes the connection to the modem. This type of entry requires two *Dialer-Token-Pairs*. The *Dialer* portion of each pair (fifth and seventh fields of entry) is used to match entries in the *Dialers* file:

```
Devices: ACU ttyf2 null 9600 212 x t2500 telebit T25
Dialers: telebit "" "" \pr\ps\c est:\007 \E\D\e \007
Dialers: T25 =&-% "" \r\p\r\c $ <K\T%%\r>\c ONLINE!
```

In the first pair, *t2500* is the *Dialer* and *telebit* is the token that is passed to the Develcon switch to tell it which device (telebit modem) to connect to your computer. This token is unique for each modem switch since each switch may be set up differently. Once the telebit modem has been connected, the second pair is accessed, where *T25* is the dialer and the token, the telephone number, is retrieved from the *Systems* file. (See the discussion of the Systems file's Phone field in "The Systems File" on page 694.)

Two escape characters can appear in a *DTP* field:

`\T`	Indicates that the Phone (Token) field should be translated by means of the *Dialcodes* file.
`\D`	Indicates that the Phone (Token) field should not be translated by means of the *Dialcodes* file. If no escape character is specified at the end of a *Devices* entry, the `\D` is assumed (default).

Device 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 station you are calling. (See the discussion of the *Systems* file,

specifically the Type field.) If you do specify the protocol, you must do it 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. This is the default protocol.
e	This protocol is faster than *g*, but it assumes error-free transmission, such as over a TCP/IP network connection.
t	This protocol, like *e*, is for use in an error-free environment, such as a TCP/IP network connection. The *t* protocol is used by systems running BSD UNIX operating systems.

The Dialers File

The *Dialers* file (*/etc/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 with an ASCII dialer (such as an automatic-dial modem).

As shown in the preceding section, the fifth field in a *Devices* file entry is an index into the *Dialers* file or a special dialer type. 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 technique is usually used to translate the equal sign (=) and the hyphen (-) characters into whatever the dialer requires for “wait for dial tone” and “pause.”

The *expect-send* fields are character strings.

The following list describes some of the escape characters used in the *Dialers* file. A sequence beginning with a backslash (\) is an escape sequence.

\p	pause (approximately 1/4 to 1/2 second)
\d	delay (approximately two 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 can be used are listed in “The Systems File” on page 694.

The *penril* entry in the *Dialers* file is executed as follows. First, the phone number argument is translated, replacing any equal sign with a **W** (wait for dial tone) and replacing any hyphen with a **P** (pause).

The handshake given by the remainder of the line works as follows:

" "	Wait for nothing; in other words, proceed to the next step.
\d	Delay for two seconds.
>	Wait for a “greater than” sign (>).
s\p9\c	Send an s, pause for 1/2 second, send a 9, send no terminating new line.
)-W\p\r\ds\p9\c-)	Wait for a closing parenthesis [)]; if it is not received, process the string between the hyphens 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 closing parenthesis.

y\c	Send a y without a new line.
:	Wait for a colon (:)
\E\TP	Enable echo checking. (From this point on, whenever a character is transmitted, handshake processing will wait for the character to be received before proceeding.) Then send the phone number. The \T instructs the program to take the phone number passed as an argument, apply the *Dialcodes* translation and the modem function translation specified by field two of this entry, then send a **P.**
9\c	Send a 9 without a new line.
OK	Wait for the string **OK.**

The Systems File

The *Systems* file (*/etc/uucp/Systems*) contains the information needed by the *uucico* daemon to establish a communications link to a remote computer. Each entry in the file represents a computer that can call or be called by your computer. In addition, UUCP software by default is configured to prevent any computer that does not appear in this file from logging in to your computer. (Refer to "Other UUCP Files" on page 709 for a description of the *remote.unknown* file.) More than one entry may be present for a particular computer. The additional entries represent alternative communications paths that will be tried in sequential order.

Using the *Sysfiles* file, you can define several files to be used as *Systems* files. See "The Sysfiles File" on page 708 for details.

Each entry in the *Systems* file has the following format:

```
System-name Time Type Class Phone Login
```

These fields are defined in the following sections:

The System-name Field

This field contains the node name of the remote computer.

The Time Field

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 options, possibly compounded with comma delimiters:

```
Su Mo Tu We Th Fr Sa
```

for individual days;

```
Wk
```

for any weekday (Mo Tu We Th Fr);

```
Any
```

for any day; and

```
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 with respect to the remote computer. (For more information on permissions, see "The Permissions File" on page 699.)

Here is an example:

```
Wk1700-0800,Sa,Su
```

This example allows calls from 5:00 p.m. to 8:00 a.m., Monday through Friday, and 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 such as 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 at 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 5 minutes after the first failure, 10 after the second, and so on until a delay of about 24 hours. If the retry subfield is present, that wait is used after every failure. 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."

The Type Field

This field contains the device type that should be used to establish the communications 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 login:nuucp password:
Oakgrass

Devices: ACU ttym2 - D1200 penril
```

You can define the protocol used to contact the station by adding it on to the Type field. The example just given shows how to attach the protocol **g** to the device type **ACU**. For direct connects, use the name of the station to which you are connecting. See "Device Protocols" on page 691.

The Class Field

This field is used to indicate the transfer speed of the device used to establish the communications link. It may contain a letter and speed (for example, *C1200*, *D1200*) to differentiate between classes of dialers. (See the discussion of the Class field in "The Devices File" on page 688.) 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 as shown here

```
Systems: eagle Any ACU D1200 NY3251 login:nuucp
password:Oakgrass

Devices:  ACU ttym2 - D1200 penril
```

If information is not required for this field, use a hyphen as a place holder for the field.

The Phone Field

This field is used to provide the phone number (token) of the remote computer for automatic dialers. 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 login:nuucp password:
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 hyphen (-) in the string instructs the ACU to pause four seconds before dialing the next digit.

If your computer is connected to a modem switch, you may access other computers that are connected to that switch. The *Systems* file entries for these computers does not have a phone number in the Phone field. Instead, this field contains the token that must be passed on to the switch so it will know which computer your computer wishes to communicate with. This token is usually just the station name. The associated *Devices* file entry should have a **\D** at the end of the entry to ensure that this field is not translated by means of the *Dialcodes* file.

The Login Field

This field contains login information given as a series of fields and subfields of the format:

```
expect send
```

The *expect* string is received and the *send* string is sent when the *expect* string is received.

The expect field may be made up of subfields of the form:

```
expect[-send-expect]...
```

The send field 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 expects *login*. If UUCP gets *login*, it goes on to the next field. If it does not get *login*, it sends nothing, followed by a new line, then

looks 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 are sent followed by a newline unless the send string is terminated with a **\c**.

Here is an example of a *Systems* file entry that uses an expect-send string:

```
owl Any ACU 1200 NY6013 "" \r login:-BREAK-login: uucpx
word: xyzzy
```

This example means "send a carriage return and wait for `ogin:` (for `Login:`). If you do not get `ogin:`, send a **<BREAK>**. When you do get `ogin:`, send the login name **uucpx**; then, when you see `word:`(the last part of `Password:`), send the password **xyzzy**."

Several escape sequences cause specific actions when they are a part of a string sent during the login sequence. The following escape sequences 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. Ignored otherwise.
\d	Delay two seconds before sending or reading more characters.
\p	Pause for approximately 1/4 to 1/2 second.
\E	Start echo checking. (From this point on, whenever a character is transmitted, login processing will wait for the character to be received before proceeding.)
\e	Turn off echo checking.
\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 backslash (\) 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.

The Dialcodes File

The *Dialcodes* file (*/etc/uucp/Dialcodes*) contains the dial-code abbreviations that can be used in the Phone field of the *Systems* file. Each entry has the following format:

```
abb dial-seq
```

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 *Systems* file entry is accessed.

For example, the following entry would work with a Phone field in the *Systems* file such as `jt7867`:

```
jt 9=555-
```

When the entry containing `jt7867` was encountered, the sequence `9=555-7867` would be sent to the dialer if the token in the dialer-token-pair was **\T**.

The Permissions File

The *Permissions* file (*/etc/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 specifies the commands that a remote site can execute on the local computer.

The program */etc/uucp/genperm* is recommended for creating a sample or default *Permissions* file from the *Systems* file.

How Permissions File Entries Are Structured

Each entry is a logical line with physical lines terminated by a backslash (\) to indicate continuation. (Note that such continuations are not possible in most other UUCP files.) 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 number sign (#) and 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 to (calls) your computer.
MACHINE	Specifies permissions that take effect when your computer logs in to (calls) a remote computer.

LOGNAME entries begin with a LOGNAME option and MACHINE entries begin with a MACHINE option.

Permissions File Considerations

Keep these rules in mind when using the *Permissions* file to restrict the level of access granted to remote computers:

- Any login ID used by a remote computer to log in 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:
  - Local send and receive requests will be executed.
  - The remote computer will be able to send files to your computer's */var/spool/uucppublic* directory.
  - The command sent by the remote computer for execution on your computer must be one of the default commands, usually *rmail*.

Permissions File Options

This section describes each option, specifies how it is used, and lists its default value.

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 string that follows 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 will be used if the REQUEST option is not specified. The REQUEST option can appear in either a LOGNAME (remote calls you) entry or a MACHINE (you call remote) entry.

A note on security: When a remote computer calls you, unless you have a unique login and password for that computer, you won't know if the computer 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 string shown here 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 string that follows specifies that files queued in your computer will be sent only when your computer calls the remote computer:

```
SENDFILES=call
```

The call value is the default for the SENDFILE option. This option is significant only in LOGNAME entries since MACHINE entries apply when calls are made to remote computers. If the option is used with a MACHINE entry, it will be ignored.

READ and WRITE

These options specify the various parts of the file system that *uucico* can read from or write to. The READ and WRITE options can be used with either MACHINE or LOGNAME entries.

The default for both the READ and WRITE options is the *uucppublic* directory as shown in the following strings:

```
READ=/var/spool/uucppublic
```

```
WRITE=/var/spool/uucppublic
```

These strings specify permission to access any file that can be accessed by a local user with “other” permissions:

```
READ=/ WRITE=/
```

Because this suggestion may compromise security, use it only if required.

The value of these entries is a colon-separated list of pathnames. The READ option is for requesting files, and the WRITE option for depositing files. One of the values must be the prefix of any full pathname of a file coming in or going out. To grant permission to deposit files in */usr/news* as well as in the public directory, the following values would be used with the WRITE option:

```
WRITE=/var/spool/uucppublic:/usr/news
```

Note that if you use the READ and WRITE options, you must specify all pathnames because the pathnames are not added to the default list. For instance, if the */usr/news*

pathname were the only one specified in a WRITE option, permission to deposit files in the public directory would be denied.

You should be careful which directories you make accessible for reading and writing by remote stations. For example, you probably wouldn't want remote computers to be able to write over your */etc/passwd* file, so */etc* shouldn't be open to writes.

NOREAD and NOWRITE

The NOREAD and NOWRITE options specify exceptions to the READ and WRITE options or defaults. The strings shown here would permit one remote computer to read any file except those in the */etc* directory (and its subdirectories—remember, these are prefixes) and to write only to the default */var/spool/uucppublic* directory:

```
READ=/ NOR EAD=/etc WRITE=/var/spool/uucppublic
```

NOWRITE works in the same manner as the NOREAD option. NOREAD and NOWRITE can be used in both LOGNAME and MACHINE entries.

CALLBACK

The CALLBACK option is used in LOGNAME entries to specify that no transaction will take place until the calling station is called back. You would use CALLBACK for two reasons: From a security standpoint, if you call back a station you can be sure it is the station it says it is. If you are doing long data transmissions, you can choose the station that will be billed for the longer call.

The string that follows specifies that your computer must call the remote computer back before any file transfers will take place:

```
CALLBACK=yes
```

The default for the CALLBACK option is

```
CALLBACK=no
```

The CALLBACK option is very rarely used. Note that if two sites have this option set for each other, a conversation cannot be started.

COMMANDS The COMMANDS option can be hazardous to the security of your station. Use it with extreme care.

The *uux* program generates remote execution requests and queues 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 defines command permissions, whether you call the remote station or it calls you.

This 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, in the following example, the entry overrides the COMMANDS default so that the computers *eagle*, *owl*, and *hawk* can now execute *rmail* and *rnews* on your computer:

```
MACHINE=eagle:owl:hawk REQUEST=yes
COMMANDS=rmail:/usr/bin/rnews
READ=/  WRITE=/
```

In addition to the names as specified above, there can be full pathnames of commands. For example, this line specifies that command *rmail* use the default path:

```
COMMANDS=rmail:/usr/bin/rnews:/usr/local/lp
```

The default paths for your computer are */bin /usr/sbin*, */usr/bsd*, and */usr/bin*. When the remote computer specifies *rnews* or */usr/bin/rnews* for the command to be executed, */usr/bin/rnews* will be executed regardless of the default path. Likewise, */usr/local/lp* is the *lp* command that will be executed.

Note: Including the ALL value in the list means that any command from the remote computer(s) specified in the entry will be executed. If you use this value, you give the remote computer full access to your computer. *Be careful*. This value allows far more access than normal users have.

This string illustrates the greater access:

```
COMMANDS=/usr/bin/rnews:ALL:/usr/local/lp
```

Two points about this string should be noted. The ALL value can appear anywhere in the string, and the pathnames specified for *rnews* and *lp* will be used (instead of the default) if the requested command does not contain the full pathnames for *rnews* or *lp*.

The VALIDATE option should be used with the COMMANDS option whenever potentially dangerous commands like *cat* and *uucp* are specified with the COMMANDS option. Any command that reads or writes files is potentially dangerous to local security when executed by the UUCP remote execution daemon (*uuxqt*).

VALIDATE — The VALIDATE option is used 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 and password for UUCP transactions. An important aspect of this validation is that the login and 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 system with a privileged login and password for UUCP transactions. Giving another system these privileges is like giving anyone on that computer a normal login and

password on your computer. Therefore, if you cannot trust everyone at the remote site, do not provide that system with a privileged login and password.

LOGNAME

The LOGNAME option ensures that remote stations attempting to log in to your computer have login privileges. The following LOGNAME entry specifies that if one of the remote computers that claims to be *eagle*, *owl*, or *hawk* logs in to 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 and password, marauding is trivial.

But what does this have to do with the COMMANDS option, which appears only 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/bin/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/bin/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*.

MACHINE Entry for "Other" Systems

You may want to specify different option values for computers your computer calls that are not mentioned in specific MACHINE entries. This situation 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/bin/Photo:/usr/bin/
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 two entries that follow 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=/
```

MYNAME — The MYNAME option is used to override the name of the local computer, when the local computer identifies itself to the remote computer. This facility is useful when a computer is replaced or renamed, and its neighbors need to process old traffic to the old name.

The Poll File

The *Poll* file (*/etc/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 at which the computer should be called. The format of entries in the *Poll* file is:

```
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 merely 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*.

The Sysfiles File

The */etc/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 to use different *Systems* files so requests for login services can be made to different phone numbers than requests for *uucp* services.
- You may want to use *Dialers* files that have 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
```

The *w* parameter 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 */etc/uucp* directory, unless a full path is given. A backslash-carriage return (**\<Return>**) can be used to continue an entry to the next line.

Here is an example using a local *Systems* file in addition to the usual *Systems* file:

```
service=uucico:cu systems=Systems:Local_Systems
```

If this line is in */etc/uucp/Sysfiles*, then both *uucico* and *cu* will first look in */etc/uucp/Systems*. If the station they're trying to call doesn't have an entry in that file, or if the entries in the file fail, then they'll look in */etc/uucp/Local_Systems*.

When different *Systems* files are defined for *uucico* and *cu* services, your station will store two different lists of stations. You can print the *uucico* list by using the *uuname* command or the *cu* list by using the *uuname* ***-c*** command.

Other UUCP Files

Three files in addition to those described in the preceding subsections have an impact on the use of basic networking facilities. In most cases, the default values are fine and no changes are needed. If you want to change the default values, however, use any standard IRIX text editor (**ed**, **vi**, or **jot**).

Maxuuxqts — This file defines the maximum number of *uuxqt* programs that can run at once. The default number is two.

Maxuuscheds — This file defines the maximum number of *uusched* programs that can run at once. The default number is two.

unknown — This file is a program that executes when a station that is not in any of the *Systems* files starts a conversation. The program logs the conversation attempt and refuses the

connection. If you change the permissions of this file so it cannot execute (*chmod 000 unknown*), your station will accept any conversation requests.

UUCP Administrative Files

The UUCP administrative 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 UUCP processes under the spool directory (for example, */var/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 following format:

  `TM.pid.ddd`

  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 pathname specified in the *C.sysnxxxx* file (discussed later in this section) 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 */var/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 this format:

  `LCK..str`

  str is either a device or computer name. These files may remain in the spool directory if the communications link is unexpectedly dropped (usually because of a computer crash). Lock files will be ignored (removed) after the parent process is no longer active. Each 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 following format:

```
C.sysnxxxx
```

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 or 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.
- Login name of the remote user to be notified upon completion of the transfer.

D. (data file) Data files are created when the command line specifies that the source file should be copied to the spool directory. The names of data files have the following format:

```
D.systmxxxxyyy
```

systm is the first five characters in the name of the remote computer and *xxxx* is a four-digit job sequence number assigned by UUCP. The four-digit job sequence number may be followed by a subsequence number, *yyy,* used when several **D.** files are 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
```

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.

Shortcut: Setting Up UUCP

Setting up UUCP involves five steps:

1. Determining the *remote* and *local* stations.
2. Making the physical connection
3. Configuring the *local* (calling) station
4. Configuring the *remote* (called) station
5. Testing the UUCP connection

Determining the Remote and Local Stations

Typically, the *local* station is the station that initiates the UUCP connection. The *remote* station is the station that responds to UUCP connection requests. However, with the arrival of *uugetty* (a program that allows bi-directional line usage) the distinction between the local and remote station is usually only the station name.

For our example, *japan* is the local station and *us* is the remote station.

Making the Physical Connection

UUCP supports physical connections for TCP/IP local area network connections, direct links, or telephone lines. This example assumes a direct link. The procedure for running UUCP over telephone line or local area networks is similar, requiring minor adjustments to the various configuration files.

A direct link constitutes a connection between two Data Terminal Equipment (DTE) devices. The devices must be fooled into thinking they are communicating with a Data Communication Equipment (DCE) device. The way to get around this is with a null modem.

The minimum pinning configuration shown in Table 21-2.

Table 21-2 Three Wire Null Modem Pinning Configuration

IRIS A	IRIS B
2 Transmit Data	3 Receive Data
3 Receive Data	2 Transmit Data
7 Signal Ground	7 Signal Ground

Attach the null modem cable to serial port two (*ttyf*2) on the local and remote workstations.

Note: The preferred cable for use with the serial ports on your system connects more pins than the minimum configuration. This cable is available from Silicon Graphics. Contact your sales representative or SGI Express. This cable can be used with a null modem adapter. The commercially available MAC SE to modem cable (“off the shelf ”') will not work properly with SGI software.

The preferred serial cable pinning configuration for both Mini-DIN8 and DB25 serial ports is described in Table 21-3:

Table 21-3 Preferred Serial Cable

Function	Mini-DIN8-Male	DB25-Male
DTR	1	9
CTS	2	5
TXD	3	2
GND	4	7
RXD	5	3
RTS	6	4
DCD	7	8
GND	8	7

Note: All pins not shown are no connects (nc). For additional information see the *serial*(7) reference page.

Configuring the Local Station

The remote station name in our example is *us*, and the local station name is *japan*. There are two steps in configuring the local station:

1. Updating standard system files
2. Modifying the UUCP configuration files

Updating Standard System Files

The three system files that you need to be concerned with are:

- */etc/passwd*
- */etc/group*
- */etc/inittab*

/etc/passwd

To ensure proper security and access, you need to ensure that the user entries for *uucp* and *nuucp* are both present and correct. The *uucp* entry in the *passwd* file is for ownership purposes and the *nuucp* entry is for remote UUCP access. Ensure that your password file has both entries and that they are the same as the following example. If the *uucp* and *nuucp* entries don't match the following, edit those accounts so they do match.

```
uucp:*:3:5:UUCP Owner:/usr/lib/uucp:/bin/csh
```

```
nuucp::10:10:Remote UUCP User:/var/spool/uucppublic:/usr/lib/
uucp/uucico
```

In the above example, the *passwd* entry for *nuucp* is split across two lines due to formatting constraints. In the actual file, the entry appears on a single line.

On a newly installed station, neither *uucp* nor *nuucp* will have a password. It is a good idea to put a "*" in the password field for uucp, since no one should log in as *uucp*. You need to assign *nuucp* a valid password that matches the password you assign for *nuucp* in the Systems file. (See "The Systems File" on page 694. For example, assign *nuucp* the password "secret".)

```
New password: secret
Re-enter new password: secret
```

/etc/group

Check this file to ensure that there are valid groups for both *uucp* and *nuucp*. Compare your uucp and nuucp group entries with the following. If there is a discrepancy, correct it now.

```
uucp::5:uucp
nuucp::10:nuucp
```

/etc/inittab

This sample entry is for the local station. It allows calls to be initiated on port two, but does not allow incoming calls on the port. Edit your */etc/inittab* entry for "t2" as follows:

```
t2:23:off:/usr/lib/uucp/uugetty -Nt 60 ttyf2 co_9600 # port 2
```

For complete information on the *uugetty* command, see the *uugetty*(1M) reference page. As usual, anytime you make a change to the */etc/inittab*, you must tell init to read the file again with the *telinit q* command. Issue the following command:

```
/etc/telinit q
```

Modifying the UUCP Configuration Files

The UUCP configuration files to be modified are:

- */etc/uucp/Systems*
- /etc/uucp/Devices
- /etc/uucp/Dialers
- /etc/uucp/Permissions

/etc/uucp/Systems

The *Systems* file contains information describing the station(s) that the local station knows about. Add the following line to the bottom of the file.

```
us Any systemx 9600 unused ogin:--ogin: nuucp ssword: \
secret
```

Note: As the *Systems* file is read-only, if using *vi* you will have to force this change to be written out by exiting *vi* with the **:wq!** option.

The first field specifies the name of the station that can call (the remote station). The second field indicates that the specified station can call at any time. The third field tells *uucp* the name of the device to use (**systemx**). The third field must match one of the first field entries found in */etc/uucp/Devices*. The forth field specifies the transfer speed (9600). The fifth field is normally used for a phone number, unused for direct links. The rest of the line handles the login sequence. It is the chat script negotiated between the local station and the remote station. This chat script is very important for a successful *uucp* connection.

/etc/uucp/Devices

The *Devices* file contains information about the physical connection between the two stations. Remove the pound sign from the *systemx* device entry so it looks like the following:

```
# ---A direct connection to a system
systemx ttyf2 - Any direct
```

Note: If you have another direct connection to a station on another port, copy the *systemx* device entry and modify the port number accordingly.

The first field in the *Devices* file links the device to the *Systems* file (third field entry). The second field tells *uucp* which port to access. The third field is used with an Automatic Call Unit (ACU). Direct links use a dash in the third field. The fourth field specifies the line speed. The "Any" entry allows the speed to be determined by the */etc/inittab* file for that particular device. The fifth field contains the dialer name. It must be a valid entry in the */etc/uucp/Dialers* file.

/etc/uucp/Dialers

This file contains the chat script for the *uucp* device. Since this is a direct connection, the chat script is picked up from the *Systems* file. However, there still has to be a valid dialers entry for the direct connection. Verify that the *Dialers* file has an entry for the "direct" dialer. Type the following command:

grep direct /etc/uucp/Dialers

The system responds with:

```
direct
#   The following entry is for use with direct connections
uudirect ""   "" \r\d in:--in:
```

/etc/uucp/Permissions

The *Permissions* file controls remote *uucp* access with regard to remote users and stations. See "The Permissions File" on page 699 for descriptions on all options. For this example, edit the Permissions file to look like the following:

```
#dent"@(#)uucp:Permissions2.2"
# This entry for public login.
# It provides the default permissions.
# See the Basic Networking Utilities Guide for more
information.
LOGNAME=nuucp MACHINE=us READ=/var/spool/uucp/uucppublic \
WRITE=/var/spool/uucppublic REQUEST=yes SENDFILES=yes \
COMMANDS=rmail
```

Note: This entry must be interpreted as a single line, even if it expands more than one physical line.

This entry specifies that the user, *nuucp*, is allowed to login from the remote station (*us*). The *nuucp* user on *us* may read any files that reside in */var/spool/uucp/uucppublic* directory and write to the general public directory */var/spool/uucppublic*. The users on *us* may request. Users on *japan* can send files.

Configuring the Remote Station

The remote station name in our example is *japan*. The local station name in our example is *us*. There are two steps in configuring the remote station:

1. Updating standard system files
2. Modifying the UUCP configuration files

Updating Standard System Files

The three system files that you need to be concerned with are:

- /etc/passwd
- /etc/group
- /etc/inittab

/etc/passwd

To ensure proper security and access, you need to ensure that the user entries for *uucp* and *nuucp* are both present and correct. The *uucp* entry in the passwd file is for ownership purposes and the *nuucp* entry is for remote UUCP access. Ensure that your password file has both entries and that they are the same as the following example. If the *uucp* and *nuucp* entries don't match the following, edit those accounts so they do match.

```
uucp:*:3:5:UUCP Owner:/usr/lib/uucp:/bin/csh
```

```
nuucp::10:10:Remote UUCP User:/var/spool/uucppublic:/usr/lib/
uucp/uucico
```

In the above example, the *passwd* entry for *nuucp* is split across two lines due to formatting constraints. In the actual file, the entry appears on a single line.

On a newly installed station, neither *uucp* nor *nuucp* has a password. It is a good idea to put a "* "in the password field for *uucp*, since no one should log in as *uucp*. You need to assign *nuucp* a valid password that matches the password you assign for *nuucp* in the Systems file. (See "The Systems File" on page 694. Assign *nuucp* the password "secret" with the command:

```
passwd nuucp
New password: secret
Re-enter new password: secret
```

/etc/group

Check this file to ensure that there are valid groups for both *uucp* and *nuucp*. Compare your *uucp* and *nuucp* group entries with the following example. If there is a discrepancy, correct it now.

```
uucp::5:uucp
nuucp::10:nuucp
```

/etc/inittab

This sample entry is for the remote station. It allows calls to be received on serial port 2, but does not allow outgoing calls on the port. Edit your */etc/inittab* entry for "t2" as follows:

```
t2:23:respawn:/usr/lib/uucp/uugetty -Nt 60 ttyf2 co_9600#pt 2
```

For complete information on the *uugetty* command, see the *uugetty*(1M) reference page. As usual, anytime you make a change to the */etc/inittab*, you must tell *init* to read the file again with the *telinit q* command. Issue the following command:

/etc/telinit q

Modifying the UUCP Configuration Files

The UUCP configuration files to be modified are:

- /etc/uucp/Systems
- /etc/uucp/Permissions

/etc/uucp/Systems

The *Systems* file contains information describing the station(s) the remote station knows about. Give the following commands and add the given line to the bottom of the file.

cd /etc/uucp

vi Systems

Add the line:

```
japan Never
```

to the bottom of the *Systems* file.

Note: The permission of the *Systems* file is read-only. If you edit this file, you may have to use a forced write in order to save your changes. Consult the *jot* or *vi* reference page for more information.

The first field specifies the name of a station that can call the local station. The second field indicates the times this station make the call. Since *japan* is a remote station which always receives calls and never calls out, *Never* indicates that the other station always calls this station.

/etc/uucp/Permissions

The *Permissions* file controls remote *uucp* access with regard to remote users and stations. See "The Permissions File" on page 699 for descriptions on all options. For this example, edit the Permissions file to look like the following:

```
#ident"@(#)uucp:Permissions2.2"
# This entry for public login.
# It provides the default permissions.
# See the Basic Networking Utilities Guide for more
                                                    information.
LOGNAME=nuucp MACHINE=japan READ=/var/spool/uucp/uucppublic\
WRITE=/var/spool/uucppublic REQUEST=yes SENDFILES=yes \
COMMANDS=rmail
```

Note: This entry must be interpreted as a single line, even if it expands to more than one physical line.

This entry specifies that the user, *nuucp*, is allowed to login from the local station (*japan*). The *nuucp* user on *japan* may read any files that reside in */var/spool/uucp/uucppublic* directory and write to the general public directory */var/spool/uucppublic*. The users on *japan* may request files. The users on *us* can send files.

Setting up UUCP on a TCP/IP Connection

In many cases, you may decide to use the UUCP tools over conventional TCP/IP network connections. There are entries in the Devices file provided for this, but you must make some changes in the */usr/etc/inetd.conf* and */etc/uucp/Systems* files. Follow these steps:

1. Edit the */usr/etc/inetd.conf* file on the remote host and find this line:

   ```
   #uucp stream tcp nowait root /usr/lib/uucp/uucpd uucpd
   ```

   Remove the leading hashmark (#) to uncomment the line, and use the commands:

   ```
   /etc/init.d/network stop
   /etc/init.d/network start
   ```

   to make the change take effect.

   This change tells the remote system to run the *uucpd* daemon when a request comes in for UUCP transfer.

2. Add a line similar to the following to your local */etc/uucp/Systems* file:

   ***remotehost* Any TCP Any**

   The name *remotehost* should be replaced with the name of the remote host you will be calling.

3. Run the */etc/uucp/genperm* command as **root** on the local host to generate the uucp permissions file.

4. Then, give the command:

   ```
   /usr/lib/uucp/uucheck -v
   ```

   To check that everything is set up correctly. There is a great deal of output, You should see output similar to the following for the specific entry you made:

   ```
   When we call system(s): (remotehost)
    We DO allow them to request files.
    They can send files to
    /var/spool/uucppublic (DEFAULT)
    They can request files from
    /var/spool/uucppublic
    /usr/lib/mail
    /usr/people/ftp
    Myname for the conversation will be MyName.
   ```

```
 PUBDIR for the conversation will be /var/spool/
uucppublic.

Machine(s): (remotehost)
CAN execute the following commands:
command (rmail), fullname (/bin/rmail)
command (rnews), fullname (/usr/bin/rnews)
command (cunbatch), fullname (/usr/lib/news/cunbatch)
```

The *cu*(1C) command does not work for UUCP over a TCP connection. Use the */usr/lib/uucp/Uutry* command instead. *Uutry* is documented completely in the *Uutry*(1M) reference page and in "Testing with Uutry" on page 725.

Testing the UUCP Connection

There are two basic tools for testing a UUCP connection:

- The *cu* program
- The *Uutry* program

Testing with cu

The *cu* program is used to test basic functionality of the UUCP connection. When you use *cu* directly, you are performing the login process as if you were the *uucp* programs. The *cu* command is also used for direct modem connections for terminal emulation. The **-d** option to *cu* is used for diagnostics and causes traces of information to be printed out to the standard output (your shell window). You should always use this mode when testing a UUCP connection.

The following command tests the physical connection to the remote station, UUCP configuration files, operating system files, and the *uucico* daemon on the remote station.

Note: The default permissions on devices (*/dev/ttyf2*) are set to 622. For *cu* to access your device, you need to change the permissions to 666.

The *cu* command must be executed from the local (calling) station. Execute the *cu* command from the local station (*japan*) as follows:

/usr/bin/cu -d us

You get output similar to the following:

```
conn(us)
Device Type us wanted
mlock ttyf2 succeeded
filelock: ok
fixline(5, 9600)
processdev: calling setdevcfg(cu, us)
gdial(direct) called
getto ret 5
device status for fd=5
F_GETFL=2,iflag='12045',oflag='0',cflag='6275',lflag='0',line
='1'
cc[0]='177',[1]='34',[2]='10',[3]='25',[4]='1',[5]='0',[6]='0
',[7]='0',
call _mode(1)
Connected
_receive started
transmit started
```

There is a pause at this point. Follow these steps now:

1. Press <**Return**> now.
2. You should see a login prompt.
3. Log in as *nuucp* and supply the password for *nuucp*. (In this case, the password is **secret**).

Now, *uucico* starts up on the remote station and displays the following prompts. (Your expected input is in bold:

```
Break your connection with a tilde(~) dot(.) and a carriage
return (<CR>).
us login: nuucp
Password: secret
IRIX System V Release 3.3.2 us
Copyright (c) 1988,1989,1990 Silicon Graphics, Inc.
All Rights Reserved.
here=japan~[us].
call tilda(.)
call _quit(0)
call _bye(0)
Disconnected
```

```
call cleanup(0)
call _mode(0)
```

Testing with Uutry

Uutry is the program that tests the copy-in/copy-out program (*uucico*). *uucico* must be functioning properly before you can actually transfer data. Issue the *Uutry* command from the local station (*japan*) to the remote station (*us*):

```
/usr/lib/uucp/Uutry us
```

You should see output similar to the following:

```
/usr/lib/uucp/uucico -r1 -sus  -x5 >/tmp/us 2>&1&
tmp=/tmp/us
mchFind called (us)
conn(us)
Device Type us wanted
mlock ttyf2 succeeded
processdev: calling setdevcfg(uucico, us)
gdial(direct) called
getto ret 5
expect: (ogin:)
```

Note: The system may pause here for several minutes.

```
sendthem (^M)
expect: (ogin:)
^M^M^J^M^J^Jus login:got it
sendthem (nuucp^M)
expect: (ssword:)
nuucp^M^JPassword:got it
sendthem (secret^M)
Login Successful: System=us
msg-ROK
Rmtname us, Role MASTER, Ifn - 5, Loginuser - root
rmesg - 'P' got Pg
wmesg 'U'g
Proto started g
*** TOP ***  -  role=1, setline - X
wmesg 'H'
rmesg - 'H' got HY
PROCESS: msg - HY
HUP:
```

```
wmesg 'H'Y
cntrl - 0
send OO 0,exit code 0
Conversation Complete: Status SUCCEEDED
```

When you see "Status SUCCEEDED," *Uutry* has successfully tested *uucico*. Press <**Ctrl-C**> to break out of *Uutry*.

UUCP Error Messages

This section describes common error messages associated with the UUCP environment. UUCP error messages can be divided into two categories: ASSERT Error Messages and STATUS Error Messages.

ASSERT Error Messages

When a process is aborted, the station records ASSERT error messages in */var/spool/uucp/.Admin/errors*. These messages include the file name, *sccsid*, *line number*, and the text listed in Table 21-4. In most cases, these errors are the result of file system problems.

Table 21-4 Assert Error Messages

Error Message	Description/Action
CAN'T OPEN	An *open*() or *fopen*() failed.
CAN'T WRITE	A *write*(), *fwrite*(), *fprint*(), or other call failed.
CAN'T READ	A *read*(), *fgets*(), or other call failed.
CAN'T CREATE	A *create*() call failed.
CAN'T ALLOCATE	A dynamic allocation failed.
CAN'T LOCK	An attempt to make an 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.

Table 21-4 (continued) Assert Error Messages

Error Message	Description/Action
CAN'T LINK	A *link*() call failed.
CAN'T CHDIR	A *chdir*() call failed.
CAN'T UNLINK	An *unlink*() call failed.
WRONG ROLE	This is an internal logic problem.
CAN'T MOVE TO CORRUPT DIR	An attempt to move some bad C. or X. files to the */var/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 was attempted, but the file already exists. This situation occurs when there is a problem with the sequence-file access. This usually indicates a software error.
NO UUCP SERVER	A TCP/IP call was 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	The *uid* cannot be found in the */etc/passwd* file. The file system is in trouble, or the */etc/passwd* file is inconsistent.
ULIMIT TOO SMALL	The **ulimit** for the current user process is too small. File transfers may fail, so transfer will not be 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.

Table 21-4 (continued) Assert Error Messages

Error Message	Description/Action
SYSLST OVERFLOW	An internal table in *gename.c* overflowed. A big or strange request was attempted.
TOO MANY SAVED C FILES	An internal table in *gename.c* overflowed. A big or strange request was attempted.
RETURN FROM FIXLINE IOCTL	An *ioctl*, which should never fail, failed. There is likely a system driver problem.
PERMISSIONS file: BAD OPTION	There is a bad line or option in the *Permissions* file. Fix it immediately.
BAD SPEED	A bad line-speed appears in the *Devices*/*Systems* files (Class field).
PKCGET READ	The remote station probably hung up. No action is required.
PKXSTART	The remote station aborted in a nonrecoverable way. This message can generally be ignored.
SYSTAT OPENFAIL	There is a problem with the modes of */var/spool/uucp/.Status*, or there is a file with bad modes in the directory.
TOO MANY LOCKS	There is an internal problem.
XMV ERROR	There is a problem with some file or directory. The problem is likely caused by the spool directory, since the modes of the destinations should have been checked before this process was attempted.
CAN'T FORK	An attempt to fork and execute failed. The current job should not be lost, but will be attempted later (*uuxqt*). No action need be taken.

STATUS Error Messages

Status error messages are stored in the */var/spool/uucp/.Status* directory. This directory contains a separate file for each remote station that your station

attempts to communicate with. These files contain status information on the attempted communication, indicating whether it was successful or not. Table 21-5 lists the most common error messages that can appear in these files.

Table 21-5 STATUS Error Messages

Error Message	Description/Action
OK	System status is normal
NO DEVICES AVAILABLE	There is currently no device available for the call. Make sure that there is a valid device in the *Devices* file for the particular station. Check the *Systems* file for the device to be used to call the station.
WRONG TIME TO CALL	A call was placed to the station at a time other than that specified in the *Systems* file.
TALKING	Self-explanatory.
LOGIN FAILED	The login for the given station failed. The problem could be a wrong login and password, wrong number, a very slow station, or failure in getting through the Dialer-Token-Pairs script.
CONVERSATION FAILED	The conversation failed after successful startup. This situation usually means that one side went down, the program aborted, or the line (link) was dropped.
DIAL FAILED	The remote station never answered. The problem could be a bad dialer or the wrong phone number.
BAD LOGIN/MACHINE COMBINATION	The station called you with a login or station name that does not agree with the Permissions file. This could be an attempt to breach system security.
DEVICE LOCKED	The calling device to be used is currently locked and in use by another process.

Table 21-5 (continued) STATUS Error Messages

Error Message	Description/Action
ASSERT ERROR	An ASSERT error occurred. Check the */var/spool/uucp/.Admin/errors* file for the error message and refer to "Other UUCP Files" on page 709
SYSTEM NOT IN Systems	The station is not in the *Systems* file.
CAN'T ACCESS DEVICE	Typically, this message means that the permissions on the device file (*/dev/tty**) are not set correctly. Some programs set these permissions, and if terminated abnormally, do not reset them to correct states. Also, check the appropriate entries in the *Systems* and *Devices* files.
DEVICE FAILED	The attempt to open the device failed.
WRONG MACHINE NAME	The called station is reporting a different name than expected.
CALLBACK REQUIRED	The called station requires that it call your computer back to start a connection.
REMOTE HAS A LCK FILE FOR ME	The remote site has a LCK file for your computer. The remote station could be trying to call your computer. If they have an older version of UUCP, the process that was talking to your station may have failed earlier, leaving the LCK file. If the remote site has the new version of UUCP and they are not communicating with your computer, then the process that has a LCK file is hung.
REMOTE DOES NOT KNOW ME	The remote computer does not have the node name of your computer in its *Systems* file.
REMOTE REJECT AFTER LOGIN	The ID used by your computer to log in does not agree with what the remote computer was expecting.

Table 21-5 (continued) STATUS Error Messages

Error Message	Description/Action
REMOTE REJECT, UNKNOWN MESSAGE	The remote computer rejected the communication with your computer for an unknown reason. The remote computer may not be running a standard version of UUCP.
STARTUP FAILED	Login succeeded, but initial handshake failed.
CALLER SCRIPT FAILED	The problem indicated by this message is usually the same as that indicated by `DIAL FAILED`. However, if it occurs often, suspect the caller script in the *Dialers* file. Use *uutry* to check the caller script.

Chapter 22

Serial Line Internet Protocol

Chapter 22 describes Serial Line Internet Protocol, commonly known as SLIP. SLIP is very useful for creating temporary or variable Wide-Area-Network (WAN) connections. Since it requires no sophisticated hardware, SLIP is in common use by many administrators. Topics covered in this chapter are:

- *Configuring SLIP.*
- *Using SLIP to connect individual systems.*
- *Using SLIP to connect networks.*
- *Debugging SLIP.*
- *The Network File System with SLIP.*
- *Using SLIP for file transfer.*

Chapter 22

SLIP and PPP

This chapter introduces the Silicon Graphics implementation of the Serial Line Internet Protocol (SLIP) and Point to Point Protocol. SLIP/PPP offers:

- simultaneous operation of multiple processes on a serial line
- network connectivity without specialized hardware
- a high level of header compression

SLIP allows the simultaneous operation of multiple processes on a serial cable or telephone line. Processes such as *ftp* and *rlogin* can share a link that an IRIS station is using for electronic mail or UUCP. Networks can be connected with the Internet Protocol features that network users expect.

The Silicon Graphics implementation provides both RFC 1144 data compression and its own proprietary data compression, which compresses header framing, checksum, and TCP/IP information to three bytes. SLIP is part of the *eoe1* software subsystem, and is installed via the *eoe1.sw.slip* package. Use the *versions*(1M) command to check to see if you have SLIP installed on your system.

PPP offers substantially the same features as SLIP, but is generally considered to be faster. PPP is part of the *eoe1* software subsystem, and is installed via the *eoe1.sw.ppp* package. Use the *versions*(1M) command to check to see if you have PPP installed on your system. Information for configuring SLIP is also generally true for configuring PPP. The types of changes and specifics added to the files in */etc/uucp* are the same. The differences in usage and configuration are detailed in "Configuring PPP" on page 737. Once installed and working, a PPP link works in the same way as a SLIP link. Thus, the sections on file transfer and NFS are true for both protocols.

The following sections are included in this chapter:

- How to configure SLIP. See "SLIP Configuration Information" on page 736.

- How to configure PPP. See “Configuring PPP” on page 737.
- Using SLIP to connect individual systems. See “Connecting Two Systems with SLIP” on page 739.
- Using SLIP to connect networks. See “Connecting Networks with SLIP” on page 745.
- How to create an “as-needed” SLIP link. See “Demand Dialing SLIP” on page 746.
- How to debug SLIP. See “Debugging a SLIP Link” on page 747.
- Using NFS with SLIP. See “NFS Under SLIP” on page 748.
- Using SLIP for file transfer. See “File Transfer Under SLIP” on page 748.

SLIP Configuration Information

SLIP can be used to connect stations or networks. SLIP configuration allows you to choose the modem and port you wish to use. If you have installed *uucp* or used a modem on your system, you are probably already familiar with most of the configuration files used by SLIP.

SLIP can be used over a telephone line with a modem at each end of the connection, or over a serial cable. This section assumes that you have read the information in “Turning On Dial-In/Dial-Out Modem Software” on page 395 or in the *Personal System Administration Guide*, or are already familiar with modem installation.

If you have never used IRIX serial devices or if you need information on serial cable pin requirements for modem use, you may want to consult “Cabling the Serial Ports” on page 404 or the *Personal System Administration Guide* for information on using the ports. You may also wish to review the reference pages for *slip*(1M), *uugetty*(1M), *getty*(1M), and *inittab*(1M).

Modem Specifications

A modem capable of at least 9600 bits per second (bps, sometimes referred to as “baud”) is required for use with SLIP. However, at this speed, your SLIP link will be quite slow. A good modem choice is any modem that

supports the v.32bis standard and a speed of 14400 bps or greater. A half-duplex modem can be used but is less desirable for interactive tasks.

Silicon Graphics does not manufacture these modems and cannot be responsible for changes to the modems. Silicon Graphics cannot guarantee modem compatibility or the quality of the telephone line used with SLIP.

If your modem supports both v.32 and PEP™ modes (or another batch-oriented mode), you can take advantage of these capabilities while using SLIP. The v.32 mode is preferred for most uses, but switching to PEP before using a file-transfer utility such as *ftp* or *rcp* will yield higher performance.

If you are using a high-speed modem, be sure it supports hardware flow control using RTS/CTS. With such a modem, use the *ttyf** device name.

Note: Silicon Graphics supports DSI, Intel, Telebit, Zyxel, US Robotics, Hayes, and most Hayes-compatible modems. While many others work, configuring them may be much more complicated and their operation is not guaranteed. There are "fix" scripts in your software distribution that are placed in */etc/uucp* when you install your system. Most commercially available modems that support the AT command set will work under IRIX.

Cable Specifications

The pin definitions for a cable connecting a suitable modem are described in "Attaching a Modem" on page 389 and in the *Personal System Administration Guide*.

Configuring PPP

The main configuration file for PPP is */etc/ppp.conf*. This file is described in detail in the *ppp*(1M) reference page. Here is an example of a working *ppp.conf* file:

```
ppp-dialup out remotehost=pppserv.dialup.com
          lochost=mymachine.dialul.com
          uucp_name=pppserv netmask=255.255.252.0
          active_timeout=300 inactive_timeout=300 add_route
```

The entry *ppp-dialup* should match an entry in the */etc/uucp/Systems* file, such as the following:

```
ppp-dialup Any ACUSLIP 38400 555-1212 "" @\r\c \
           ogin: mymachine ssword: TbZ4QlMA PPP
```

There should also be a corresponding entry in the */etc/uucp/Dialers* file:

```
ppp-dialup  =W-, "" \d\r\pATs2=128s0=1&C0\r\c \
           OK\r-ATs2=128s0=1&C0\r\c-OK\r \
           ATdtw\T\r\c CONNECT
```

And an entry in the */etc/uucp/Devices* file:

```
ACUslip ttyf1 null 38400 212 x ppp-dialup
```

You should also have your */etc/inittab* file set up to expect the modem port speed you are using (in this case, 38400 bps) and to have *getty* or *uugetty* turned off. Specific details on editing the */etc/inittab* file and restarting *telinit* are found in the *Personal System Administration Guide* or in "Adding a Terminal or Modem" on page 384. The following entry works with the above listed file entries for PPP:

```
t1:23:off:/etc/uucp/uugetty ttyd1 dx_38400  # ppp modem
```

You may use dial-out PPP with the above listed entry, but for dial-in PPP, you will need to configure *uugetty* to answer the line, as described in "Adding a Terminal or Modem" on page 384.

It is important to note that the above listed entries are simply examples of one configuration that works for one example site. The same entries may not yield satisfactory results in every case, due to other differences in site configuration and modem manufacturer and model. For example, the PPP site you are dialing into may require different settings in the */etc/ppp.conf* file and the entry in the *Dialers* file assumes a specific brand and model of modem. Also, the example assumes you have issued the correct *fix* script command (found in the */etc/uucp* directories and described in "Modem ''fix'' Scripts" on page 744.)

When you have configured the files, you must issue the *ppp* command as **root**. The following command works for the above listed example file entries:

```
ppp -r ppp-dialup
```

For complete information on the *ppp* command and its options, see the *ppp*(1M) reference page.

Connecting Two Systems with SLIP

A SLIP link between two workstations or servers offers a TCP/IP work environment similar to that of Ethernet. SLIP provides the ability to use familiar TCP/IP utilities such as *ftp* and *rlogin* from any IRIS station accessible by telephone.

A SLIP connection can be initiated by either or both of the stations you wish to link. The simplest, most common SLIP connection is initiated by one of the two stations. The first part of this chapter describes the configuration of a simple SLIP link between two stations with distinct roles; one station always calls in and the other station always receives the call. The changes necessary to convert a simple SLIP link to a SLIP link that can be initiated by either station are also described in this chapter. See "Demand Dialing SLIP" on page 746. Demand-Dialing SLIP is the most economical method of using SLIP for an "as-needed" network connection.

In this chapter, the term *local station* refers to the station that initiates the SLIP connection. The *remote* station receives a call from the local station. When the local station successfully connects, a *slip* process is started on the remote station. Once the connection is established, the *slip* processes sleep on both stations until one or both are terminated.

Overview of Configuration

Before beginning configuration of SLIP, verify that the following conditions exist:

- The stations to be linked by SLIP know each other's unique SLIP IP address. The SLIP interface name and address should be different from the station's other network interface names and addresses. For example, *slip-$HOSTNAME*.
- UUCP is installed on the local station. If the remote station is ever to initiate the connection or otherwise dial out on its modem, UUCP software must be installed on it also.

Note: The UNIX-to-UNIX Copy software and SLIP subsystems are included as a standard part of IRIX (**eoe2.sw.uucp** and **eoe1.sw.slip**) but they are not installed by default.

To link two systems with SLIP, make these changes:

1. Edit the file */etc/uucp/Devices* on the local station to reflect the modem and port used by SLIP.
2. Change the attributes for the port selected on the local station in its */etc/inittab,* and the attributes for the port selected on the remote station in its */etc/inittab.*
3. Create a line in */etc/uucp/Systems* on the local station that contains dialing and login information.
4. Create an entry in the remote station's */etc/passwd* file for the SLIP link.
5. Add a line to */usr/etc/remoteslip* on the remote station that contains the *slip* command you want to execute on the remote station.
6. Install and configure a modem or connect a cable at each end of the SLIP connection.

The Local Station

This section explains how to configure the local station. The examples that follow use the local station name of *wenders*.

/etc/uucp/Devices

The file */etc/uucp/Devices* is used to configure the desired device, modem speed, and dialer program for SLIP on your IRIS station. The correct format for a line appropriate for SLIP in */etc/uucp/Devices* is:

```
ACUSLIP device  null speed 212 x dialer
```

device can be any flow control device associated with a port not currently in use. *speed* can be any speed supported by your modem. *dialer* can be any dial program listed in */etc/uucp/Dialers*.

If you want to configure more than one modem speed, use a different port, or use a modem that supports a different command set, you must create a new line in */etc/uucp/Devices* that reflects the change.

For example, the following command configures SLIP to use a Telebit™ T2500 modem at 38,400 bps on the serial port 2 hardware flow control device in */etc/uucp/Devices*:

```
ACUSLIP ttyf2 null 38400 212 x t25slip
```

/etc/inittab

To use the port you specified in */etc/uucp/Devices* for dialing out, change its attributes in */etc/inittab*. Ensure that either *getty* is turned off or *uugetty* is in use instead.

Edit the file */etc/inittab* and find the line that corresponds to the name of the port selected. For example, on the local station, SLIP uses *ttyf2*, so the line is changed to read:

```
t2:23:off:/etc/getty ttyf2 co_38400         # port 2
```

This line turns off the *getty* program on port number 2.

If the SLIP link can be initiated by either station, you must turn on *uugetty.* For example, to configure a symmetric link using Telebit T2500 modems, change the line to:

```
t2:23:respawn:/usr/lib/uucp/uugetty -Nt 60 -it25in,conn
ttyf2 \ dx_38400
```

Changes made to */etc/inittab* are acted upon when *init* reexamines the */etc/inittab* file. To make *init* reexamine */etc/inittab* immediately, use this command:

```
/etc/telinit q
```

For more information, see *inittab*(4).

/etc/uucp/Systems

SLIP uses the information in */etc/uucp/Systems* to call the remote station. The remote station's node name and telephone number, as well as the local

modem's speed and a password, are all kept here and are used to log in to the remote station. In this example, the local station *wenders* uses this line in its */etc/uucp/Systems* file to call station *lynch*. The connection is made at 38,400 bps with SLIP options recorded on *lynch* in */usr/etc/remoteslip*. The password is "hopper". SLIP logs in to the remote station by responding with "`slip-wenders`" to the login prompt, and "`hopper`" to the password prompt.

```
lynch Any ACUSLIP 38400 5551212 "" @\r\c ogin:--ogin: \
@slip-wenders asswd: hopper SLIP
```

The information must be in one continuous line. The last string, "`SLIP`", forces the local station to wait for the remote station to announce that it is starting the SLIP protocol.

For more information on the */etc/uucp/Devices*, */etc/uucp/Systems*, and */etc/uucp/Dialers* files, see Chapter 21, "UUCP." The file */usr/etc/remoteslip* is discussed in the next section.

To call *lynch* from *wenders*, invoke *slip*:

```
/usr/etc/slip -o -p comp -r lynch
```

The Remote Station

SLIP requires an entry in */etc/passwd* in order to log in. The user ID and group ID must both be zero (0). Instead of the shell specified at the end of a normal entry in */etc/passwd*, SLIP uses the file */usr/etc/remoteslip*. To log in as "*slip-wenders*," the station *lynch* should have this line in */etc/passwd*:

```
slip-wenders:3RsB768WRAN2.:0:0:slip from wenders: \ /usr/
people/slip-wenders:/usr/etc/remoteslip
```

An entry like this one is necessary for each station that calls in by using SLIP. For maximum system security, it is strongly suggested that you create a home directory for each SLIP user. Using open directories such as */tmp* opens your system up to a variety of threats.

Note: The encrypted password in the example does not represent a real password. You must use *passwd*(1M) to set the password for the SLIP login.

/usr/etc/remoteslip

In the */etc/passwd* entry for *slip-wenders*, the login shell is specified as the file */usr/etc/remoteslip*. This file is used to invoke SLIP on a remote station. In */usr/etc/remoteslip*, the *slip* command can specify the remote station's name and any other options appropriate to that connection.

/usr/etc/remoteslip is a Bourne shell script. You can add to the case statement as you would to any Bourne shell script case statement. The *sh*(1) online reference page contains detailed information about shell script programming. Each SLIP connection should have an entry in the following format:

```
slip-nodename )
    exec /usr/etc/slip options
    ;;
```

nodename is the name of the remote station. Options for *slip* are detailed in the *slip*(1M) reference page.

The */usr/etc/remoteslip* file might contain this entry on the station *lynch*:

```
# Edit the case statement as required.
case $USER in
       slip-wenders)
              exec /usr/etc/slip -p comp -i -r wenders
              ;;
       *)
              exec /usr/etc/slip -i -r $USER
              ;;
esac
```

The connection between *lynch* and *wenders* uses the Silicon Graphics proprietary header prediction/compression for faster data transfer because the **-p comp** option is specified. To use RFC 1144 compression, use **-p cslip** instead. The option tells SLIP that the session is input from another station. The *slip* option, **-r wenders**, specifies the remote station's name.

/etc/inittab

There must be a *getty* or *uugetty* running on the incoming port on the remote station. In other words, the incoming port must be enabled. For example, if

the remote station uses serial port 2 with Telebit T2500 modems, change the line in its */etc/inittab* to:

```
t2:23:respawn:/usr/lib/uucp/uugetty -Nt 60 -it25in,conn
ttyf2 \ dx_38400
```

Configuring the Modem

Modem configuration utilities are provided to facilitate the configuration of two popular brands of modems, Hayes and Telebit, and true work-alikes.

To send information across a telephone line, SLIP requires a modem at each end. To configure a modem at each end of the connection between the stations *wenders* and *lynch*, you must connect the modems and configure the software switches in the modems. Each modem must be physically connected to the desired port on one of the stations and a telephone line.

Modem ”fix” Scripts

To configure your modem you must use the Superuser account (*root*). Go to the directory */etc/uucp*. There you will find a number of modem ”fix” scripts. They are named with the form

`fix-`*modemtype*

These scripts contain useful information and sample lines for */etc/inittab*.

The scripts *fix-hayes* and *fix-telebit* can be used to configure modems that fully conform to two of the most popular modem configuration standards. The options to *fix-hayes* and *fix-telebit* are:

-i	Configures modem to accept incoming calls
-o	Configures modem to initiate outgoing calls
-io	Configures modem for both incoming and outgoing calls

The last argument must be followed by a serial port number, in the form **d#**. That is, to configure the modem to use the device *ttyf2*, use **d2** in your command line.

Once you have connected a modem to the station, you can use the appropriate utility to configure your modem. For example, if you want to install a Telebit T2500 modem on *ttyf2* of the station *wenders* so that station *lynch* can dial in, type:

```
/etc/uucp/fix-telebit -io d2
```

In addition to Hayes and Telebit fix scripts, there are also scripts provided for DSI, Intel, and Zyxel modems. Most modems that support the AT command set can be correctly configured by one of these scripts.

Configuring a Bidirectional SLIP Link

A simple SLIP link, in which one station must always initiate the connection, can be changed to allow either station to initiate the connection.

Follow the procedure described in "Connecting Two Systems with SLIP" on page 739 for configuring the files *Devices, Systems, inittab passwd,* and *remoteslip* on the local and remote stations. Each station can then connect to the other. In addition, the modem *fix* script should be used to configure the modem on each station for incoming and outgoing calls.

The *getty* utility normally used to facilitate the login process interferes with attempts to call out. To call in and out using the same port, configure the port to use *uugetty,* which is part of the UUCP software subsystem.

Edit the */etc/inittab* file on both stations to use *uugetty* as described above. Edit */usr/etc/remoteslip* on the local station and */etc/uucp/Systems* on the remote station. In other words, each station will now be both "local" and "remote." Therefore, both stations need both sets of changes described in "The Local Station" on page 740.

Connecting Networks with SLIP

SLIP can be used to connect Ethernet or FDDI networks that use the Internet Protocol. Network SLIP connections are set up like other SLIP connections.

To have a SLIP connection between networks start automatically, create a file in */etc/init.d* for that purpose. For example, the file */etc/init.d/slipstart* can be

used to automate the connection between the stations *wenders* and *lynch*. The *slipstream* file resides on the station where you want to initiate the SLIP link and should be linked to an appropriate startup file in the */etc/rc2.d* directory. For example, to have *lynch* automatically generate a SLIP connection between networks at station startup, on *lynch* you would:

1. Create the */etc/init.d/slipstart* file and add the following lines:

   ```
   #!/bin/sh
   # Bourne shell script for starting SLIP connection.
   /usr/etc/slip -c -p comp -r wenders &
   ```

   The **-c** option automates maintenance by redialing the remote station (in this case *wenders*) should the telephone line fail.

2. Create the necessary */etc/rc2.d* link on *lynch*:

   ```
   ln -s /etc/init.d/slipstart /etc/rc2.d/S32slipstart
   ```

Demand Dialing SLIP

If you have a SLIP link that is used irregularly but frequently, it is likely to be economical for you to make that link a demand-dialing link. With demand-dialing, the SLIP program will make the telephone connection on an as-needed basis, when there is network traffic to be transmitted, and it will drop the connection when there is no traffic. By default, much of the non-essential network traffic will not cause a link to be established. For example, the traffic generated by the time daemon (*timed*) would not cause a call to be placed, but a request for file transfer would cause the connection to be made.

To use demand-dialing, add the **-q** option to your slip command on system that initiates the connection. Demand-dialing mode is also known as ''quiet'' mode. For complete information on this and other options, see the *slip*(1M) reference page.

Debugging a SLIP Link

If you are having trouble bringing up your SLIP link, test the line with another utility. If you are familiar with *uucp*, you may want to establish a *uucp* link between the stations as a means of debugging the connection. Most SLIP users will find *cu* an easier way to debug a SLIP link.

Note: *cu* requires a *direct* entry in the */etc/uucp/Devices* file. Refer to "The Devices File" on page 688 for more details.

When debugging a SLIP connection, check each station separately. First check the port and modem on the local station by using a *cu*(1C) command like this:

```
cu -d -s speed -l port
```

For example, to test the port and modem installed on the station *wenders*, you would use this *cu* command:

```
cu -d -s 38400 -l ttyf2
```

(It may be necessary to turn off the *(uu)getty* first by changing `respawn` to `off` on the line for the port in the file */etc/inittab*.)

The modem should respond. Many modems respond by printing `AT`. If the modem does not respond as expected, review the SLIP configuration procedure for the local station and review the modem configuration and documentation. If the modem does respond as expected, disconnect from *cu* by typing a tilde followed by a dot:

```
~.
```

Connect the stations you want to link as you would for the SLIP link. On the local station, use *cu* to call the remote station through the port and connection you have already verified with *cu*.

Check the connection to the remote station with a *cu* command like this:

```
cu -d -sspeed telno
```

For example, to test the connection between *wenders* and *lynch*, you would call *lynch* from *wenders* with this *cu* command:

```
cu -d -s38400 5552002
```

You should see the local station tell the modem to call the remote station. Eventually, you should see the login prompt. Type the *send* strings from the */etc/uucp/Systems* file in response to the *expect* strings.

NFS Under SLIP

You can run NFS over a SLIP link. NFS will be very slow because of the amount of information transferred in NFS transactions. You may be able to improve performance with these measures:

- Use NFS with modem faster than 9600 bps.
- If your modem offers a choice between v.32 mode and "batch-oriented" or "half-duplex" mode (for example, PEP), use the batch-oriented mode.

  Some dialers require v.32 or PEP mode. When choosing a new modem mode, verify that the dialer you specify uses the mode you want. Dialer modes are noted in the file */etc/uucp/Dialers*.
- Use one of the SLIP header prediction/compression options.

  For more information regarding the two compression options, **comp** and **cslip**, see *slip*(1M).
- Adjust the NFS options **rsize**, **wsize, timeo**, and **retrans** when mounting NFS file systems.

  To improve performance, read and write smaller blocks and specify longer timeouts. See *fstab*(4) for more information on NFS file system options.

File Transfer Under SLIP

File transfer may be slow if other demanding utilities share the SLIP link. For faster file transfer, try these solutions:

- If your modem offers a choice between v.32 mode and batch-oriented mode (for example, PEP), use the batch-oriented mode.
- Use *uucp* if you do not want to share the line with other utilities.

Appendices

The following are the appendices to the Advanced Site and Server Administration Guide. They are, in order:

- *Appendix A, IRIX Kernel Tunable Parameters*
- *Appendix B, IRIX Device Files*
- *Appendix C, IRIX Kernel Error Messages*
- *Appendix D, IRIX sendmail Reference*
- *Appendix E, BIND Standard Resource Record Format*

Appendix A

IRIX Kernel Tunable Parameters

This appendix describes the tunable parameters that define kernel structures. These structures keep track of processes, files, and system activity. Many of the parameter values are specified in the files found in */var/sysgen/mtune*.

If the system does not respond favorably to your tuning changes, you may want to return to your original configuration or continue making changes to related parameters. An unfavorable response to your tuning changes can be as minor as the system not gaining the hoped-for speed or capacity, or as major as the system becoming unable to boot or run. This generally occurs when parameters are changed to a great degree. Simply maximizing a particular parameter without regard for related parameters can upset the balance of the system to the point of inoperability. For complete information on the proper procedure for tuning your operating system, read Chapter 5, "Tuning System Performance."

Format of This Appendix

The rest of this appendix describes the more important tunable parameters according to function. Related parameters are grouped into sections. These sections include:

- General tunable parameters. See "General Tunable Parameters" on page 753.
- Spinlock tunable parameters. See "Spinlocks Tunable Parameters" on page 761.
- System limits tunable parameters. See "System Limits Tunable Parameters" on page 763.
- Resource limits tunable parameters. See "Resource Limits Tunable Parameters" on page 766.

- Paging tunable parameters. See “Paging Tunable Parameters” on page 777.
- IPC tunable parameters–including interprocess communication messages, semaphores, and shared memory. See “IPC Tunable Parameters” on page 786, “IPC Messages Tunable Parameters” on page 788, “IPC Semaphores Tunable Parameters” on page 792, and “IPC Shared Memory Tunable Parameters” on page 797.
- Streams tunable parameters. See “Streams Tunable Parameters” on page 800.
- Signal parameters. See “Signal Parameters” on page 802.
- Dispatch parameters. See “Dispatch Parameters” on page 803.
- Extent File System (EFS) parameters. See “EFS Parameters” on page 807.
- Loadable driver parameters. See “Loadable Drivers Parameters” on page 809.
- CPU actions parameters. See “CPU Actions Parameters” on page 812.
- Switch parameters. See “Switch Parameters” on page 813.
- Timer parameters. See “Timer parameters” on page 817.
- Network File System (NFS) parameters. See “NFS Parameters” on page 819.
- UNIX Domain Sockets (UDS) parameters. See “UDS Parameters” on page 822.

Each section begins with a short description of the activities controlled by the parameters in that section, and each listed parameter has a description that may include:

- Name – the name of the parameter.
- Description – a description of the parameter, including the file in which the parameter is specified, and the formula, if applicable.
- Value – the default setting and, if applicable, a range. Note that the value given for each parameter is usually appropriate for a single-user graphics workstation.

- When to Change – the conditions under which it is appropriate to change the parameter.
- Notes – other pertinent information, such as error messages.

Note that the tunable parameters are subject to change with each release of the system.

General Tunable Parameters

The following group of tunable parameters specifies the size of various system structures. These are the parameters you will most likely change when you tune a system.

- *nbuf* – specifies the number of buffer headers in the file system buffer cache.
- *callout_himark* – specifies the high water mark for callouts
- *ncallout* – specifies the initial number of callouts
- *reserve_ncallout* – specifies the number of reserved callouts
- *ncsize* – specifies the name cache size.
- *ndquot* – used by the disk quota system.
- *nhbuf* – specifies the number of buffer hash buckets in the disk buffer cache.
- *nproc* – specifies the number of user processes allowed at any given time.
- *maxpmem* – specifies the maximum physical memory address.
- *syssegsz* – specifies the maximum number of pages of dynamic system memory.
- *maxdmasz* – specifies the maximum DMA transfer in pages.

nbuf

Description of nbuf

The *nbuf* parameter specifies the number of buffer headers in the file system buffer cache. The actual memory associated with each buffer header is dynamically allocated as needed and can be of varying size, currently 1 to 128 blocks (512 to 64KB).

The system uses the file system buffer cache to optimize file system I/O requests. The buffer memory caches blocks from the disk, and the blocks that are used frequently stay in the cache. This helps avoid excess disk activity.

Buffers are used only as transaction headers. When the input or output operation has finished, the buffer is detached from the memory it mapped and the buffer header becomes available for other uses. Because of this, a small number of buffer headers is sufficient for most systems. If *nbuf* is set to 0, the system automatically configures *nbuf* for average systems. There is little overhead in making it larger for non-average systems.

The *nbuf* parameter is defined in */var/sysgen/mtune*.

Value of nbuf

Default:	0 (Automatically configured if set to 0)
Formula:	100 + (*total number of pages of memory*/40)
Range:	up to 6000

When to Change nbuf

The automatic configuration is adequate for average systems. If you see dropping "cache hit" rates in *sar*(1M) and *osview*(1M) output, increase this parameter. Also, if you have directories with a great number of files (over 1000), you may wish to raise this parameter.

callout_himark

Description of callout_himark

The *callout_himark* parameter specifies the maximum number of callout table entries system-wide. The callout table is used by device drivers to provide a timeout to make sure that the system does not hang when a device does not respond to commands.

This parameter is defined in */var/sysgen/mtune* and has the following formula:

nproc + 32

where:

nproc is the maximum number of processes, system-wide.

Value of callout_himark

Default: 0 (Automatically configured if set to 0)

Formula: *nproc* + 32

Range: 42 - 1100

When to Change callout_himark

Increase this parameter if you see console error messages indicating that no more callouts are available.

ncallout

Description of ncallout

The *ncallout* parameter specifies the number of callout table entries at boot time. The system will automatically allocate one new callout entry if it runs out of entries in the callout table. However, the maximum number of entries in the callout table is defined by the *callout_himark* parameter.

Value of ncallout

Default: 40

Range: 20–1000

When to Change ncallout

Change *ncallout* if you are running an unusually high number of device drivers on your system. Note that the system automatically allocates additional entries when needed, and releases them when they are no longer needed.

reserve_ncallout

Description of reserve_ncallout

The *reserve_ncallout* parameter specifies the number of reserved callout table entries. These reserved table entries exist for kernel interrupt routines when the system has run out of the normal callout table entries and cannot allocate additional entries.

Value of reserve_ncallout

Default: 5

Range: 0-30

ncsize

Description of ncsize

This parameter controls the size of the name cache. The name cache is used to allow the system to bypass reading directory names out of the main buffer cache. A name cache entry contains a directory name, a pointer to the directory's in-core inode and version numbers, and a similar pointer to the directory's parent directory in-core inode and version number.

Value of ncsize

Default: 0 (Automatically configured if set to 0)

Range: 268-6200

ndquot

Description of ndquot

The *ndquot* parameter controls disk quotas.

Value of ndquot

Default: 0 (Automatically configured if set to 0)

Range: 268-6200

nhbuf

Description of nhbuf

The *nhbuf* parameter specifies the number of entries in the buffer hash table. Each table's entry is the head of a queue of buffer headers. When a block is requested, a hashing algorithm searches the buffer hash table for the requested buffer. A small hash table reduces the algorithm's efficiency; a large table wastes space.

nhbuf is defined in */var/sysgen/mtune* and has the following formula:

nbuf/16 (with the result rounded down to the nearest power of 2)

Value of nhbuf

Default: 0 (Automatically configured if set to 0)

Formula: *nbuf*/4 (result rounded down to the nearest power of 2)

When to Change nhbuf

You should not have to change this parameter.

nproc

Description of nproc

The *nproc* parameter specifies the number of entries in the system process (*proc*) table. Each running process requires an in-core *proc* structure. Thus *nproc* is the maximum number of processes that can exist in the system at any given time.

The default value of *nproc* is based on the amount of memory on your system. To find the currently auto-configured value of *nproc*, use the *systune*(1M) command.

The *nproc* parameter is defined in */var/sysgen/mtune*.

Value of nproc

Default: 0 (Automatically configured if set to 0)

Range: 30–10000

When to Change nproc

Increase this parameter if you see an overflow in the *sar -v* output for the *proc -sz ov* column or you receive the operating system message:

```
no more processes
```

This means that the total number of processes in the system has reached the current setting. If processes are prevented from forking (being created), increase this parameter. A related parameter is *maxup*.

Notes on nproc

If a process can't fork, make sure that this is system-wide, and not just a user ID problem (see the *maxup* parameter).

If *nproc* is too small, processes that try to fork receive the operating system error:

```
EAGAIN: No more processes
```

The shell also returns a message:

```
fork failed: too many processes
```

If a system daemon such as *sched*, *vhand*, *init*, or *bdflush* can't allocate a process table entry, the system halts and displays:

```
No process slots
```

maxpmem

Description of maxpmem

The *maxpmem* parameter specifies the amount of physical memory (in pages) that the system can recognize. If set to zero (0), the system will use all available pages in memory. A value other than zero defines the physical memory size (in pages) that the system will recognize.

This parameter is defined in */var/sysgen/mtune*.

Value of maxpmem

Default: 0 (Automatically configured if set to 0)

Range: 1024 pages – total amount of memory

When to Change maxpmem

You don't need to change this parameter, except when benchmarks require a specific system memory size less than the physical memory size of the system. This is primarily useful to kernel developers. You can also boot the system with the command:

```
maxpmem = memory_size
```

added on the boot command line to achieve the same effect.

syssegsz

Description of syssegsz

This is the maximum number of pages of dynamic system memory.

Value of syssegsz

Default: 0 (Autoconfigured if set to 0)

Range: 0x2000 - 0x10000

When to Change syssegsz

Increase this parameter correspondingly when *maxdmasz* is increased, or when you install a kernel driver that performs a lot of dynamic memory allocation.

maxdmasz

Description of maxdmasz

The maximum DMA transfer expressed in pages of memory. This amount must be less than the value of *syssegsz* and *maxpmem*.

Value of maxdmasz

Default: 1024 pages

Range: 1 - *syssegsz*

When to Change maxdmasz

Change this parameter when you need to be able to use very large read or write system calls, or other system calls that perform large scale DMA. This situation typically arises when using optical scanners, film recorders or some printers.

Spinlocks Tunable Parameters

R3000-based multiprocessor machines have a limited number of MPBUS hardware locks. To reduce the possibility of spinlock depletion, groups of spinlocks use shared pools of locks. The following parameters are included:

- *sema_pool_size* – the number of spinlocks pooled for all semaphores.
- *vnode_pool_size* – the number of spinlocks pooled for vnodes.
- *file_pool_size* – the number of spinlocks pooled for file structures.

sema_pool_size

Description of sema_pool_size

This parameter specifies the number of spinlocks pooled for all semaphores.

Value of sema_pool_size

Default: 8192

Range: 1024-16384

When to Change sema_pool_size

This parameter is used exclusively for R3000-based multiprocessor machines. The only time this parameter might need to be changed is if IRIX panics and delivers the error message:

```
out of spinlocks
```

This can generally happen only if a driver requiring a large number of spinlocks is added to the system. User spinlock requirements do not affect this parameter. In the case of such a panic and error message, reduce this parameter to the next smaller power of two. (For example, if the value is 8192, reduce it to 4096.)

vnode_pool_size

Description of vnode_pool_size

This parameter specifies the number of spinlocks pooled for vnodes.

Value of vnode_pool_size

Default: 1024

Range: 512-2048

When to Change vnode_pool_size

This parameter is only used for R3000-based multiprocessor machines. The only time this parameter might need to be changed is if IRIX panics and delivers the error message "out of spinlocks." This can generally only happen if a driver requiring a large number of spinlocks is added to the system. User spinlock requirements do not affect this parameter. In the case of such a panic and error message, reduce this parameter to the next smaller power of two. (For example, if the value is 1024, reduce it to 512.)

file_pool_size

Description of file_pool_size

This parameter specifies the number of spinlocks pooled for file structures.

Value of file_pool_size

Default: 1024

Range: 512-2048

When to Change file_pool_size

This parameter is used exclusively for R3000-based multiprocessor machines. The only time this parameter might need to be changed is if IRIX panics and delivers the error message "out of spinlocks." This can generally happen only if a driver requiring a large number of spinlocks is added to the

system. User spinlock requirements do not affect this parameter. In the case of such a panic and error message, reduce this parameter to the next smaller power of two. (For example, if the value is 1024, reduce it to 512.)

System Limits Tunable Parameters

IRIX has configurable parameters for certain system limits. For example, you can set maximum values for each process (its core or file size), the number of groups per user, the number of resident pages, and so forth. These parameters are listed below. All parameters are set and defined in */var/sysgen/mtune*.

- *maxup* – the number of processes per user
- *ngroups_max* – the number of groups to which a user may belong
- *maxwatchpoints* – the maximum number of ''watchpoints'' per process.
- *nprofile* – amount of disjoint text space to be profiled
- *maxsymlinks* – specifies the maximum number of symlinks expanded in a pathname.

maxup

Description of maxup

The *maxup* parameter defines the number of processes allowed per user login. This number should always be at least 5 processes smaller than *nproc*.

Value of maxup

Default: 150 processes

Range: 15–10000 (but never larger than *nproc* - 5)

When to Change maxup

Increase this parameter to allow more processes per user. In a heavily loaded time-sharing environment, you may want to decrease the value to reduce the number of processes per user.

ngroups_max

Description of ngroups_max

The *ngroups_max* parameter specifies the maximum number of multiple groups to which a user may simultaneously belong.

The constants NGROUPS_UMIN <= *ngroups_max* <= NGROUPS_UMAX are defined in </*usr/include/sys/param.h*>. NGROUPS_UMIN is the minimum number of multiple groups that can be selected at *lboot* time. NGROUPS_UMAX is the maximum number of multiple groups that can be selected at *lboot* time and is the number of group-id slots for which space is set aside at compile time. NGROUPS, which is present for compatibility with networking code (defined in </*usr/include/sys/param.h*>), must not be larger than *ngroups_max*.

Value of ngroups_max

Default: 16

Range: 0-32

When to Change ngroups_max

The default value is adequate for most systems. Increase this parameter if your system has users who need simultaneous access to more than 16 groups.

maxwatchpoints

Description of maxwatchpoints

maxwatchpoints sets the maximum number of watchpoints per process. Watchpoints are set and used via the *proc*(4) file system. This parameter specifies the maximum number of virtual address segments to be watched in the traced process. This is typically used by debuggers.

Value of maxwatchpoints

Default: 100

Range: 1-1000

When to Change maxwatchpoints

Raise *maxwatchpoints* if your debugger is running out of watchpoints

nprofile

Description of nprofile

nprofile is the maximum number of disjoint text spaces that can be profiled using the *sprofil*(2) system call. This is useful if you need to profile programs using shared libraries or profile an address space using different granularities for different sections of text.

Value of nprofile

Default: 100

Range: 100-200

When to Change nprofile

Change *nprofile* if you need to profile more text spaces than are currently configured.

maxsymlinks

Description of maxsymlinks

This parameter defines the maximum number of symbolic links that will be followed during filename lookups (for example, during the *open*(2) or *stat*(2) system calls) before ceasing the lookup. This limit is required to prevent loops where a chain of symbolic links points back to the original file name.

Value of maxsymlinks

Default: 30

Range: 0-50

When to Change maxsymlinks

Change this parameter if you have pathnames with more than 30 symbolic links.

Resource Limits Tunable Parameters

You can set numerous limits on a per-process basis by using *getrlimit*(2), *setrlimit*(2), and *limit*, the shell built-in command. These limits are inherited, and the original values are set in */var/sysgen/mtune*. These limits are different from the system limits listed above in that they apply only to the current process and any child processes that may be spawned. To achieve similar effects, you can also use the *limit* command within the Bourne, C, and Korn shells (*/bin/sh*, */bin/csh*, and */bin/ksh*).

Each limit has a default and a maximum. Only the superuser can change the maximum. Each resource can have a value that turns off any checking RLIM_INFINITY. The default values are adequate for most systems.

The following parameters are associated with system resource limits:

- *ncargs* – the number of bytes of arguments that may be passed during an *exec*(2) call.
- *rlimit-core-cur* – the maximum size of a core file.
- *rlimit-core-max* – the maximum value *rlimit-core-cur* may hold.
- *rlimit-cpu-cur* – the limit for maximum cpu time available to a process.
- *rlimit-cpu-max* – the maximum value *rlimit-cpu-current* may hold.
- *rlimit-data-cur* – the maximum amount of data space available to a process.
- *rlimit-data-max* – the maximum value *rlimit-data-cur* may hold.
- *rlimit-fsize-cur* – the maximum file size available to a process.
- *rlimit-fsize-max* – the maximum value *rlimit-fsize-cur* may hold.

- *rlimit-nofile-cur* – the maximum number of file descriptors available to a process.
- *rlimit-nofile-max* – the maximum value *rlimit-nofile-cur* may hold.
- *rlimit-rss-cur* – the maximum resident set size available to a process.
- *rlimit-rss-max* – the maximum value *rlimit-rss-cur* may hold.
- *rlimit-stack-cur* – the maximum stack size for a process.
- *rlimit-stack-max* – the maximum value *rlimit-stack-cur* may hold.
- *rlimit-vmem-cur* – the maximum amount of virtual memory for a process.
- *rlimit-vmem-max* – the maximum value *rlimit-vmem-cur* may hold.
- *rsshogfrac* – the percentage of memory allotted for resident pages.
- *rsshogslop* – the number of pages above the resident set maximum that a process may use.
- *shlbmax* – the maximum number of shared libraries with which a process can link.

ncargs

Description of ncargs

The *ncargs* parameter specifies the maximum size of arguments in bytes that may be passed during an *exec*(2) system call.

This parameter is specified in */var/sysgen/mtune.*

Value of ncargs

Default: 20480

Range: 5120–262144

When to Change ncargs

The default value is adequate for most systems. Increase this parameter if you get the following message from *exec*(2), *shell*(1), or *make*(1):

```
E2BIG arg list too long
```

Note on ncargs

Setting this parameter too large wastes memory (although this memory is pageable) and may cause some programs to function incorrectly. Also note that some shells may have independent limits smaller than *ncargs*.

rlimit_core_cur

Description of rlimit_core_cur

The current limit to the size of core image files for the given process.

Value of rlimit_core_cur

Default: 0x7fffffff

Range: 0–0x7fffffff

When to change rlimit_core_cur

Change this parameter when you want to place a cap on core file size.

rlimit_core_max

Description of rlimit_core_max

The maximum limit to the size of core image files.

Value of rlimit_core_max

Default: 0x7fffffff

Range: 0–0x7fffffff

When to change rlimit_core_max

Change this parameter when you want to place a maximum restriction on core file size. *rlimit_core_cur* cannot be larger than this value.

rlimit_cpu_cur

Description of rlimit_cpu_cur

The current limit to the amount of cpu time in minutes that may be used in executing the process.

Value of rlimit_cpu_cur

Default: 0x7fffffff (2147483647 minutes)

Range: 0–0x7fffffff

When to change rlimit_cpu_cur

Change this parameter when you want to place a cap on cpu usage.

rlimit_cpu_max

Description of rlimit_cpu_max

The maximum limit to the amount of cpu time that may be used in executing a process.

Value of rlimit_cpu_max

Default: 0x7fffffff

Range: 0–0x7fffffff

When to change rlimit_cpu_max

Change this parameter when you want to place a maximum restriction on general cpu usage.

rlimit_data_cur

Description of rlimit_data_cur

The current limit to the data size of the process.

Value of rlimit_data_cur

Default: *rlimit_vmem_cur* * NBPP (0x20000000)

Range: 0–2 GB (0x7fffffff)

When to change rlimit_data_cur

Change this parameter when you want to place a cap on data segment size.

rlimit_data_max

Description of rlimit_data_max

The maximum limit to the size of data that may be used in executing a process.

Value of rlimit_data_max

Default: *rlimit_vmem_cur* * NBPP (0x20000000)

Range: 0–2 GB (0x7fffffff)

When to change rlimit_data_max

Change this parameter when you want to place a maximum restriction on the size of the data segment of any process.

rlimit_fsize_cur

Description of rlimit_fsize_cur

The current limit to file size on the system for the process.

Value of rlimit_fsize_cur

Default: 2 GB (0x7fffffff)

Range: 0–0x7fffffff bytes

When to change rlimit_fsize_cur

Change this parameter when you want to place a limit on file size.

rlimit_fsize_max

Description of rlimit_fsize_max

The maximum limit to file size on the system.

Value of rlimit_fsize_max

Default: 2 GB (0x7fffffff)

Range: 0–0x7fffffff bytes

When to change rlimit_fsize_max

Change this parameter when you want to place a maximum size on all files.

rlimit_nofile_cur

Description of rlimit_nofile_cur

The current limit to the number of file descriptors that may be used in executing the process.

Value of rlimit_nofile_cur

Default: 200

Range: 20–0x7fffffff (2 GB)

When to change rlimit_nofile_cur

Change this parameter when you want to place a cap on the number of file descriptors.

rlimit_nofile_max

Description of rlimit_nofile_max

The maximum limit to the number of file descriptors that may be used in executing a process.

Value of rlimit_nofile_max

Default: 2500

Range: 0–0x7fffffff

When to change rlimit_nofile_max

Change this parameter when you want to place a maximum restriction on the number of file descriptors.

rlimit_rss_cur

Description of rlimit_rss_cur

The current limit to the resident set size (the number of pages of memory in use at any given time) that may be used in executing the process. This limit is the larger of the results of the following two formulae:

physical_memory_size - 4 MB

or

physical_memory_size ∗ 9/10

Value of rlimit_rss_cur

Default: 0 (Automatically configured if set to 0)

Range: 0–(*rlimit_vmem_cur* * NBPP) (0x7fffffff)

When to change rlimit_rss_cur

Change this parameter when you want to place a cap on the resident set size of a process.

rlimit_rss_max

Description of rlimit_rss_max

The maximum limit to the resident set size that may be used in executing a process.

Value of rlimit_rss_max

Default: (*rlimit_vmem_cur* * NBPP) (0x2000000)

Range: 0–(*rlimit_vmem_cur* * NBPP) (0x7fffffff)

When to change rlimit_rss_max

Change this parameter when you want to place a maximum restriction on resident set size.

rlimit_stack_cur

Description of rlimit_stack_cur

The current limit to the amount of stack space that may be used in executing the process.

Value of rlimit_stack_cur

Default: 64 MB (0x04000000)

Range: 0–2 GB (0x7fffffff)

When to change rlimit_stack_cur

Change this parameter when you want to place a limit on stack space usage.

rlimit_stack_max

Description of rlimit_stack_max

The maximum limit to the amount of stack space that may be used in executing a process.

Value of rlimit_stack_max

Default: *rlimit_vmem_cur* * NBPP (0x20000000)

Range: 0–2 GB (0x7fffffff)

When to change rlimit_stack_max

Change this parameter when you want to place a maximum restriction on stack space usage.

rlimit_vmem_cur

Description of rlimit_vmem_cur

The current limit to the amount of virtual memory that may be used in executing the process.

Value of rlimit_vmem_cur

Default: *rlimit_vmem_cur* * NBPP (0x20000000)

Range: 0–2 GB (0x7fffffff)

When to change rlimit_vmem_cur

Change this parameter when you want to place a cap on virtual memory usage.

rlimit_vmem_max

Description of rlimit_vmem_max

The maximum limit to the amount of virtual memory that may be used in executing a process.

Value of rlimit_vmem_max

Default: *rlimit_vmem_cur* * NBPP (0x20000000)

Range: 0–2 GB (0x7fffffff)

When to change rlimit_vmem_max

Change this parameter when you want to place a maximum restriction on virtual memory usage.

rsshogfrac

Description of rsshogfrac

The number of physical memory pages occupied by a process at any given time is called its resident set size (RSS). The limit to the RSS of a process is determined by its allowable memory-use resource limit. *rsshogfrac* is designed to guarantee that even if one or more processes are exceeding their RSS limit, some percentage of memory is always kept free so that good interactive response is maintained.

Processes are permitted to exceed their RSS limit until either:

- one or more processes exceed their default RSS limit (thereby becoming an "RSS hog") and the amount of free memory drops below *rsshogfrac* of the total amount of physical memory; or

- the amount of free memory drops below GPGSHI

In either of these cases, the paging daemon runs and trims pages from all RSS processes exceeding the RSS limit.

The parameter RSSHOGFRAC is expressed as a fraction of the total physical memory of the system. The default value is 75 percent.

This parameter is specified in */var/sysgen/mtune*. For more information, see the *gpgshi*, *gpgslo*, and *rsshogslop* resource limits.

Value of rsshogfrac

Default: 75% of total memory

Range: 0–100% of total memory

When to Change rsshogfrac

The default value is adequate for most systems.

rsshogslop

Description of rsshogslop

To avoid thrashing (A condition where the computer devotes 100% of its CPU cycles to swapping and paging), a process can use up to *rsshogslop* more pages than its resident set maximum (see "Resource Limits Tunable Parameters" on page 766).

This parameter is specified in */var/sysgen/mtune*. For more information, see the *rsshogfrac* resource limit.

Value of rsshogslop

Default: 20

When to Change rsshogslop

The default value is adequate for most systems.

shlbmax

Description of shlbmax

The *shlbmax* parameter specifies the maximum number of shared libraries with which a process can link.

This parameter is specified in */var/sysgen/mtune.*

Value of shlbmax

Default: 8

Range: 3–32

When to Change shlbmax

The default value is adequate for most systems. Increase this parameter if you see the following message from *exec*(2):

```
ELIBMAX cannot link
```

Paging Tunable Parameters

The paging daemon, *vhand*, frees up memory as the need arises. This daemon uses a ''least recently used'' algorithm to approximate process working sets and writes those pages out to disks that have not been touched during a specified period of time. The page size is 4K. When memory gets exceptionally tight, *vhand* may swap out entire processes.

vhand reclaims memory by:

- stealing memory from processes that have exceeded their permissible resident set size maximum, forcing delayed write data buffers out to disk (with *bdflush*) so that the underlying pages can be reused
- calling down to system resource allocators to trim back dynamically sized data structures
- stealing pages from processes in order of lowest-priority process first, and the least-recently-used page first within that process

- swapping out the entire process

The following tunable parameters determine how often *vhand* runs and under what conditions. Note that the default values should be adequate for most applications.

The following parameters are included:

- *bdflushr* – specifies how often, in seconds, the *bdflush* daemon is executed; *bdflush* performs periodic flushes of dirty file system buffers.
- *gpgsmsk* – specifies the mask used to determine if a given page may be swapped.
- *gpgshi* – the number of free pages above which *vhand* stops stealing pages.
- *gpgslo* – the number of free pages below which *vhand* starts stealing pages
- *maxlkmem* – The *maxlkmem* parameter specifies the maximum number of physical pages that can be locked in memory (by *mpin*(2) or *plock*(2)) by a non-superuser process.
- *maxsc* – the maximum number of pages that may be swapped by the *vhand* daemon in a single operation.
- *maxfc* – the maximum number of pages that will be freed at once.
- *maxdc* – the maximum number of pages that will be written to disk at once.
- *minarmem* – the minimum available resident pages of memory.
- *minasmem*– the minimum available swappable pages of memory.
- *tlbdrop* – number of clock ticks before a process' wired entries are flushed.

bdflushr

Description of bdflushr

The *bdflushr* parameter specifies how often, in seconds, the *bdflush* daemon is executed; *bdflush* performs periodic flushes of dirty file system buffers.

This parameter is specified in */var/sysgen/mtune.* For more information, see the *autoup* kernel parameter.

Value of bdflushr

Default: 5

Range: 1–31536000

When to Change bdflushr

This value is adequate for most systems. Increasing this parameter increases the chance that more data could be lost if the system crashes. Decreasing this parameter increases system overhead.

gpgsmsk

Description of gpgsmsk

The *gpgsmsk* parameter specifies the mask used to determine if a given page may be swapped. Whenever the pager (*vhand*) is run, it decrements software reference bits for every active page. When a process subsequently references a page, the counter is reset to the limit (NDREF, as defined in */usr/include/sys/immu.h*). When the pager is looking for pages to steal back (if memory is in short supply), it takes only pages whose reference counter has fallen to *gpgsmsk* or below.

This parameter is specified in */var/sysgen/mtune.*

Also see */usr/include/sys/immu.h* and */usr/include/sys/tuneable.h* and the *gpgshi* and *gpgslo* kernel parameters for more information.

Value of gpgsmsk

Default: 2

Range: 0–7

When to Change gpgsmsk

This value is adequate for most systems.

Notes on gpgsmsk

If the value is greater than 4, pages are written to the swap area earlier than they would be with the default value of *gpgsmsk*. Thus swapping/paging may occur before it should, unnecessarily using system resources.

gpgshi

Description of gpgshi

When the *vhand* daemon (page handler) is stealing pages, it stops stealing when the amount of free pages is greater than *gpgshi*.

In other words, *vhand* starts stealing pages when there are fewer than *gpgslo* free pages in the system. Once *vhand* starts stealing pages, it continues until there are *gpgshi* pages.

If, at boot time, *gpgslo* and *gpgshi* are 0, the system sets *gpgshi* to 8% of the number of pages of memory in the system, and sets *gpgslo* to one half of *gpgshi*.

This parameter is specified in */var/sysgen/mtune*. For more information, see the kernel parameters *gpgsmsk* and *gpgslo*.

Value of gpgshi

Default: 0 (automatically configured to 8% of memory if set to 0)

Range: 30 pages – 1/2 of memory

When to Change gpgshi

This value is adequate for most systems.

Notes on gpgshi

If this parameter is too small, *vhand* cannot supply enough free memory for system-wide demand.

gpgslo

Description of gpgslo

When the *vhand* daemon (page handler) executes, it won't start stealing back pages unless there are fewer than *gpgslo* free pages in the system. Once *vhand* starts stealing pages, it continues until there are *gpgshi* pages.

This parameter is specified in */var/sysgen/mtune*. For more information, see the *gpgshi* and *gpgsmsk* kernel parameters.

Value of gpgslo

Default: 0 (automatically configured to half of *gpgshi* if set to 0)

Range: 10 pages – 1/2 of memory

When to Change gpgslo

This value is adequate for most systems.

Notes on gpgslo

If this parameter is too small, *vhand* does not start swapping pages; thus entire processes must be swapped. If this parameter is too large, *vhand* swaps pages unnecessarily.

maxlkmem

Description of maxlkmem

The *maxlkmem* parameter specifies the maximum number of physical pages that can be locked in memory (by *mpin*(2) or *plock*(2)) per non-superuser process.

This parameter is specified in */var/sysgen/mtune*.

Value of maxlkmem

Default: 2000

Range: 0 pages – 3/4 of physical memory

When to Change maxlkmem

Increase this parameter only if a particular application has a real need to lock more pages in memory.

On multi-user servers, you may want to decrease this parameter and also decrease *rlimit_vmem_cur*.

Notes on maxlkmem

When pages are locked in memory, the system can't reclaim those pages, and therefore can't maintain the most efficient paging.

maxfc

Description of maxfc

The *maxfc* parameter specifies the maximum number of pages that may be freed by the *vhand* daemon in a single operation. When the paging daemon (*vhand*) starts stealing pages, it collects pages that can be freed to the general page pool. It collects, at most, *maxfc* pages at a time before freeing them. Do not confuse this parameter with *gpgshi*, which sets the total number of pages that must be free before *vhand* stops stealing pages.

This parameter is specified in */var/sysgen/mtune.*

Value of maxfc

Default: 100

Range: 50–100

When to Change maxfc

This value is adequate for most systems.

maxsc

Description of maxsc

The *maxsc* parameter specifies the maximum number of pages that may be swapped by the *vhand* daemon in a single operation. When the paging daemon starts tossing pages, it collects pages that must be written out to swap space before they are actually swapped and then freed into the general page pool. It collects at most *maxsc* pages at a time before swapping them out.

This parameter is specified in */var/sysgen/mtune.*

Value of maxsc

Default: 100

Range: 8–100

When to Change maxsc

You may want to decrease this parameter on systems that are swapping over NFS (Network File System). This is always the case for diskless systems to increase performance.

maxdc

Description of maxdc

maxdc is the maximum number of pages which can be saved up and written to the disk at one time.

Value of maxdc

Default: 100 pages

Range: 1-100

When to Change maxdc

If the system is low on memory and consistently paging out user memory to remote swap space (for example, mounted via NFS), decrease this parameter by not more than 10 pages at a time. However, this parameter's setting does not usually make any measurable difference in system performance.

minarmem

Description of minarmem

This parameter represents the minimum available resident memory that must be maintained in order to avoid deadlock.

Value of minarmem

Default: 0 (Autoconfigured if set to 0)

When to Change minarmem

The automatically configured value of this parameter should always be correct for each system. You should not have to change this parameter.

minasmem

Description of minasmem

This parameter represents the minimum available swappable memory that must be maintained in order to avoid deadlock.

Value of minasmem

Default: 0 (Autoconfigured if set to 0)

When to Change minasmem

The automatically configured value of this parameter should always be correct for each system. You should not have to change this parameter.

tlbdrop

Description of tlbdrop

This parameter specifies the number of clock ticks before a process' wired entries are flushed.

Value of tlbdrop

Default: 100

When to Change tlbdrop

If *sar*(1) indicates a great deal of transaction lookaside buffer (*utlbmiss*) overhead in a very large application, you may need to increase this parameter. In general, the more the application changes the memory frame of reference in which it is executing, the more likely increasing *tlbdrop* will help performance. You may have to experiment somewhat to find the optimum value for your specific application.

IPC Tunable Parameters

The IPC tunable parameters set interprocess communication (IPC) structures. These structures include IPC messages, specified in */var/sysgen/mtune/msg*; IPC semaphores, specified in */var/sysgen/mtune/sem*; and IPC shared memory, specified in */var/sysgen/mtune/shm*.

If IPC (interprocess communication) structures are incorrectly set, certain system calls will fail and return errors.

Before increasing the size of an IPC parameter, investigate the problem by using *ipcs*(1) to see if the IPC resources are being removed when no longer needed. For example, *shmget* returns the error ENOSPC if applications do not remove semaphores, memory segments, or message queues.

Note that IPC objects differ from most IRIX objects in that they are not automatically freed when all active references to them are gone. In particular, they are not deallocated when the program that created them exits.

Table A-1 lists error messages, system calls that cause the error, and parameters to adjust. Subsequent paragraphs explain the details you need to know before you increase the parameters listed in this table.

Table A-1 System Call Errors and IPC Parameters to Adjust

Message	System Call	Parameter
EAGAIN	*msgsnd*()	see below
EINVAL	*msgsnd*() *shmget*()	msgmax shmmax
EMFILE	*shmat*()	sshmseg
ENOSPC	*semget*() *shmget*()	msgmni semmni, semmns, shmmni

EAGAIN

If IPC_NOWAIT is set, *msgsnd* can return EAGAIN for a number of reasons:

- The total number of bytes on the message queue exceeds *msgmnb*.
- The total number of bytes used by messages in the system exceeds *msgseg * msgsz*.
- The total number of system-wide message headers exceeds *msgmax*.

EINVAL

shmget (which gets a new shared memory segment identifier) will fail with EINVAL if the given size is not within *shmmin* and *shmmax*. Since *shmmin* is set to the lowest possible value (1), and *shmmax* is very large, it should not be necessary to change these values.

EMFILE

shmat will return EMFILE if it attaches more than *sshmseg* shared memory segments. *sshmseg* is the total number of system-shared memory segments per process.

ENOSPC

shmget will return ENOSPC if:

- *shmmni* (the system-wide number of shared memory segments) is too small. However, applications may be creating shared memory segments and forgetting to remove them. So, before making a parameter change, use *ipcs*(1) to get a listing of currently active shared memory segments.

semget will return ENOSPC if:

- *semmni* is too small, indicating that the total number of semaphore identifiers is exceeded.
- *semmns* (the system-wide number of semaphores) is exceeded. Use *ipcs* to see if semaphores are being removed as they should be.

msgget will return ENOSPC if:

- *msgmni* is too small. Use *ipcs* to see if message queues are being removed as they should be.

IPC Messages Tunable Parameters

If no one on the system uses or plans to use IPC messages, you may want to consider excluding this module. The following tunable parameters are associated with interprocess communication messages (see the *msgctl*(2) reference page):

- *msgmax* – specifies the maximum size of a message.
- *msgmnb* – specifies the maximum length of a message queue.
- *msgmni* – specifies the maximum number of message queues system-wide.
- *msgseg* – specifies the maximum number of message segments system-wide.
- *msgssz* – specifies the size, in bytes, of a message segment.
- *msgtql* – specifies the maximum number of message headers system-wide.

msgmax

Description of msgmax

The *msgmax* parameter specifies the maximum size of a message.

This parameter is specified in */var/sysgen/mtune/msg*.

Value of msgmax

Default: 16 * 1024 (0x4000)

Range: 512-0x8000

When to Change msgmax

Increase this parameter if the maximum size of a message needs to be larger. Decrease the value to limit the size of messages.

msgmnb

Description of msgmnb

The *msgmnb* parameter specifies the maximum length of a message queue.

This parameter is specified in */var/sysgen/mtune/msg*.

Value of msgmnb

Default: 32 * 1024 (0x8000)

Range: *msgmax*–1/2 of physical memory

When to Change msgmnb

Increase this parameter if the maximum number of bytes on a message queue needs to be longer. Decrease the value to limit the number of bytes per message queue.

msgmni

Description of msgmni

The *msgmni* parameter specifies the maximum number of message queues system-wide.

This parameter is specified in */var/sysgen/mtune/msg*.

Value of msgmni

Default: 50

Range: 10–1000

When to Change msgmni

Increase this parameter if you want more message queues on the system. Decrease the value to limit the message queues.

Notes on msgmni

If there are not enough message queues, a *msgget*(2) system call that attempts to create a new message queue returns the error:

ENOSPC: No space left on device

msgseg

Description of msgseg

The *msgseg* parameter specifies the maximum number of message segments system-wide. A message on a message queue consists of one or more of these segments. The size of each segment is set by the *msgssz* parameter.

This parameter is specified in */var/sysgen/mtune/msg*.

Value of msgseg

Default: 1536

When to Change msgseg

Modify this parameter to reserve the appropriate amount of memory for messages. Increase this parameter if you need more memory for message segments on the system. Decrease the value to limit the amount of memory used for message segments.

Notes on msgseg

If this parameter is too large, memory may be wasted (saved for messages but never used). If this parameter is too small, some messages that are sent will not fit into the reserved message buffer space. In this case, a *msgsnd*(2) system call waits until space becomes available.

msgssz

Description of msgssz

The *msgssz* parameter specifies the size, in bytes, of a message segment. Messages consist of a contiguous set of message segments large enough to accommodate the text of the message. Using segments helps to eliminate fragmentation and speed the allocation of the message buffers.

This parameter is specified in */var/sysgen/mtune/msg*.

Value of msgssz

Default: 8

When to Change msgssz

This parameter is set to minimize wasted message buffer space. Change this parameter only if most messages do not fit into one segment with a minimum of wasted space.

If you modify this parameter, you may also need to change the *msgseg* parameter.

Notes on msgssz

If this parameter is too large, message buffer space may be wasted by fragmentation, which in turn may cause processes that are sending messages to sleep while waiting for message buffer space.

msgtql

Description of msgtql

The *msgtql* parameter specifies the maximum number of message headers system-wide, and thus the number of outstanding (unread) messages. One header is required for each outstanding message.

This parameter is specified in */var/sysgen/mtune/msg*.

Value of msgtql

Default: 40

Range: 10–1000

When to Change msgtql

Increase this parameter if you require more outstanding messages. Decrease the value to limit the number of outstanding messages.

Notes on msgtql

If this parameter is too small, a *msgsnd*(2) system call attempting to send a message that would put *msgtql* over the limit waits until messages are received (read) from the queues.

IPC Semaphores Tunable Parameters

If no one on the system uses or plans to use IPC semaphores, you may want to consider excluding this module.

The following tunable parameters are associated with interprocess communication semaphores (see the *semctl*(2) reference page):

- *semmni* – specifies the maximum number of semaphore identifiers in the kernel.
- *semmns* – specifies the number of ipc semaphores system-wide.
- *semmnu* – specifies the number of ''undo'' structures system-wide.
- *semmsl* – specifies the maximum number of semaphores per semaphore identifier.
- *semopm* – specifies the maximum number of semaphore operations that can be executed per *semop*(2) system call.
- *semume* – specifies the maximum number of ''undo'' entries per undo structure.
- *semvmx* – specifies the maximum value that a semaphore can have.
- *semaem* – specifies the adjustment on exit for maximum value.

semmni

Description of semmni

The *semmni* parameter specifies the maximum number of semaphore identifiers in the kernel. This is the number of unique semaphore sets that can be active at any given time. Semaphores are created in sets; there may be more than one semaphore per set.

This parameter is specified in */var/sysgen/mtune/sem*.

Value of semmni

Default: 10

When to Change semmni

Increase this parameter if processes require more semaphore sets. Increasing this parameter to a large value requires more memory to keep track of semaphore sets. If you modify this parameter, you may need to modify other related parameters.

semmns

Description of semmns

The *semmns* parameter specifies the number of ipc semaphores system-wide. This parameter is specified in */var/sysgen/mtune/sem*.

Value of semmns

Default: 60

When to Change semmns

Increase this parameter if processes require more than the default number of semaphores.

Notes on semmns

If you set this parameter to a large value, more memory is required to keep track of semaphores.

semmnu

Description of semmnu

The *semmnu* parameter specifies the number of ''undo'' structures system-wide. An undo structure, which is set up on a per-process basis, keeps track of process operations on semaphores so that the operations may be ''undone'' if the structure terminates abnormally. This helps to ensure that an abnormally terminated process does not cause other processes to wait indefinitely for a change to a semaphore.

This parameter is specified in */var/sysgen/mtune/sem*.

Value of semmnu

Default: 30

When to Change semmnu

Change this parameter when you want to increase/decrease the number of undo structures permitted on the system. *semmnu* limits the number of processes that can specify the UNDO flag in the *semop*(2) system call to undo their operations on termination.

semmsl

Description of semmsl

The *semmsl* parameter specifies the maximum number of semaphores per semaphore identifier.

This parameter is specified in */var/sysgen/mtune/sem*.

Value of semmsl

Default: 25

When to Change semmsl

Increase this parameter if processes require more semaphores per semaphore identifier.

semopm

Description of semopm

The *semopm* parameter specifies the maximum number of semaphore operations that can be executed per *semop*() system call. This parameter permits the system to check or modify the value of more than one semaphore in a set with each *semop*() system call.

This parameter is specified in */var/sysgen/mtune/sem*.

Value of semopm

Default: 10

When to Change semopm

Change this parameter to increase/decrease the number of operations permitted per *semop*() system call. You may need to increase this parameter if you increase *semmsl* (the number of semaphore sets), so that a process can check/modify all the semaphores in a set with one system call.

semume

Description of semume

The *semume* parameter specifies the maximum number of ''undo'' entries per undo structure. An undo structure, which is set up on a per-process basis, keeps track of process operations on semaphores so that the operations may

be ''undone'' if it terminates abnormally. Each undo entry represents a semaphore that has been modified with the UNDO flag specified in the *semop*(2) system call.

This parameter is specified in */var/sysgen/mtune/sem*.

Value of semume

Default: 10

When to Change semume

Change this parameter to increase/decrease the number of undo entries per structure. This parameter is related to the *semopm* parameter (the number of operations per *semop*(2) system call).

semvmx

Description of semvmx

The *semvmx* parameter specifies the maximum value that a semaphore can have.

This parameter is specified in */var/sysgen/mtune/sem*.

Value of semvmx

Default: 32767 (maximum value)

When to Change semvmx

Decrease this parameter if you want to limit the maximum value for a semaphore.

semaem

Description of semaem

The *semaem* parameter 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.

This parameter is specified in */var/sysgen/mtune/sem*.

Value of semaem

Default: 16384 (maximum value)

When to Change semaem

Change this parameter to decrease the maximum value for the adjustment on exit value.

IPC Shared Memory Tunable Parameters

The following tunable parameters are associated with interprocess communication shared memory:

- *shmall* – specifies the maximum number of pages of shared memory that can be allocated at any given time to all processes combined.
- *shmmax* – specifies the maximum size of an individual shared memory segment.
- *shmmin* – specifies the minimum shared memory segment size.
- *shmmni* – specifies the maximum number of shared memory identifiers system-wide.
- *sshmseg* – specifies the maximum number of attached shared memory segments per process.

shmall

Description of shmall

The *shmall* parameter specifies the maximum number of pages of shared memory that can be allocated at any given time to all processes on the system combined.

This parameter is specified in */var/sysgen/mtune/shm*.

Value of shmall

Default: 512

Range: 256-2048

When to Change shmall

Keep this parameter as small as possible so that the use of shared memory does not cause unnecessary swapping.

Decrease this parameter to limit the amount of memory that can be used for shared memory segments. You may do this if swapping is occurring because large amounts of memory are being used for shared memory segments. But be aware that if an application requires a larger shared memory segment, that application will not run if this limit is set too low.

shmmax

Description of shmmax

The *shmmax* parameter specifies the maximum size of an individual shared memory segment.

This parameter is specified in */var/sysgen/mtune/shm*.

Value of shmmax

Default: (*rlimit_vmem_cur* * NBPP) (0x20000000)

When to Change shmmax

Keep this parameter small if it is necessary that a single shared memory segment not use too much memory.

shmmin

Description of shmmin

The *shmmin* parameter specifies the minimum shared memory segment size.

This parameter is specified in */var/sysgen/mtune/shm*.

Value of shmmin

Default: 1 byte

When to Change shmmin

Increase this parameter if you want an error message to be generated when a process requests a shared memory segment that is too small.

shmmni

Description of shmmni

The *shmmni* parameter specifies the maximum number of shared memory identifiers system-wide.

This parameter is specified in */var/sysgen/mtune/shm*.

Value of shmmni

Default: 100

Range: 10–1000

When to Change shmmni

Increase this parameter by one (1) for each additional shared memory segment that is required, and also if processes that use many shared memory segments reach the *shmmni* limit.

Decrease this parameter if you need to reduce the maximum number of shared memory segments of the system at any one time. Also decrease it to reduce the amount of kernel space taken for shared memory segments.

sshmseg

Description of sshmseg

The *sshmseg* parameter specifies the maximum number of attached shared memory segments per process. A process must attach a shared memory segment before the data can be accessed.

This parameter is specified in */var/sysgen/mtune/shm*.

Value of sshmseg

Default: 100

Range: 1–SHMMNI

When to Change sshmseg

Keep this parameter as small as possible to limit the amount of memory required to track the attached segments.

Increase this parameter if processes need to attach more than the default number of shared memory segments at one time.

Streams Tunable Parameters

The following parameters are associated with STREAMS processing.

- *nstrpush* – maximum number of modules that can be pushed on a stream.
- *strctlsz* – maximum size of the *ctl* part of message.
- *strmsgsz* – maximum stream message size.

nstrpush

Description of nstrpush

nstrpush defines the maximum number of STREAMS modules that can be pushed on a single stream.

Value of nstrpush

Default: 9

Range: 9-10

When to Change nstrpush

Change *nstrpush* from 9 to 10 modules when you need an extra module.

strctlsz

Description of strctlsz

strctlsz is the maximum size of the *ctl* buffer of a STREAMS message. See the *getmsg*(2) or *putmsg*(2) reference pages for a discussion of the *ctl* and *data* parts of a STREAMS message.

Value of strctlsz

Default: 1024

When to Change strctlsz

Change *strctlsz* when you need a larger buffer for the *ctl* portion of a STREAMS message.

strmsgsz

Description of strmsgsz

strmsgsz defines the maximum STREAMS message size. This is the maximum allowable size of the *ctl* part plus the *data* part of a message. Use this parameter in conjunction with the *strctlsz* parameter described above to set the size of the *data* buffer of a STREAMS message. See the *getmsg*(2) or *putmsg*(2) reference pages for a discussion of the *ctl* and *data* parts of a STREAMS message.

Value of strmsgsz

Default: 0x8000

When to Change strmsgsz

Change this parameter in conjunction with the *strctlsz* parameter to adjust the sizes of the STREAMS message as a whole and the *data* portion of the message.

Signal Parameters

The following signal parameters control the operation of interprocess signals within the kernel:

- *maxsigq* – specifies the maximum number of signals that can be queued.

maxsigq

Description of maxsigq

The maximum number of signals that can be queued. Normally, multiple instances of the same signal result in only one signal being delivered. With the use of the SA_SIGINFO flag, outstanding signals of the same type are queued instead of being dropped.

Value of maxsigq

Default	64
Range	32-32767

When to Change maxsigq

Raise *maxsigq* when a process expects to receive a great number of signals and 64 queue places may be insufficient to avoid losing signals before they can be processed. Change *maxsigq* to a value appropriate to your needs.

Dispatch Parameters

One of the most important functions of the kernel is ‘‘dispatching’’ processes. When a user issues a command and a process is created, the kernel endows the process with certain characteristics. For example, the kernel gives the process a priority for receiving CPU time. This priority can be changed by the user who requested the process or by the Superuser. Also, the length of time (*slice-size*) that a process receives in the CPU is adjustable by a dispatch parameter. The Periodic Deadline Scheduler (PDS) is also part of the dispatch group. The deadline scheduler is invoked via the *schedctl*(2) system call in a user program and requires the inclusion of <*sys/schedctl.h*>. The following parameters are included in the dispatch group:

- *ndpri_hilim* – sets the highest non-degrading priority a user process may have.
- *ndpri_lolim* – sets the lowest non-degrading priority a user process may have.

- *runq_dl_refframe* – sets a limit on the amount of the reference frame that can be allocated.
- *runq_dl_nonpriv* – controls the amount of the reference frame that can be allocated by non-privileged user processes.
- *runq_dl_use* – specifies the longest interval that a deadline process may request.
- *slice-size* – specifies the amount of time a process receives at the CPU.

ndpri_hilim

Description of ndpri_hilim

The *ndpri_hilim* parameter sets the highest non-degrading priority a user process may have.

Note that the higher the numeric value of *ndpri_hilim*, the lower the priority of the process.

Value of ndpri_hilim

Default	128
Range	30-255

When to Change ndpri_hilim

Change this parameter when you want to limit user process priority in a busy system with many users or when you want to enable a high priority user process, for example, a real-time graphics application.

ndpri_lolim

Description of ndpri_lolim

ndpri-lolim sets the lowest possible non-degrading priority for a user process. Note that lower priority values give a process higher scheduling priority.

Value of ndpri_lolim

Default: 39

Range: 30-255

When to Change ndpri_lolim

The default value is adequate for most systems.

runq_dl_maxuse

Description of runq_dl_maxuse

This parameter sets an absolute limit on the amount of the reference frame (set by the *runq_dl_refframe* parameter) that can be allocated under any circumstances.

Value of runq_dl_maxuse

Default: 700

Range: 0-100000

When to Change runq_dl_maxuse

If your deadline-scheduled processes require more scheduled CPU time, increase the value of *runq_dl_maxuse* and *runq_dl_nonpriv*.

runq_dl_nonpriv

Description of runq_dl_nonpriv

This parameter controls the amount of the reference frame (set by the *runq_dl_refframe* parameter) that can be allocated by non-privileged user processes.

Value of runq_dl_nonpriv

Default: 200

Range: 0-100000

When to change runq_dl_nonpriv

If your non-privileged deadline processes require more CPU time, increase this value.

runq_dl_reffrane

Description of runq_dl_reffrane

This parameter specifies the longest interval that a deadline process may request.

Value of runq_dl_reffrane

Default: 1000

Range: 0-100000

When to Change runq_dl_reffrane

If you change the values of *runq_dl_nonpriv* and *runq_dl_use*, you may need to change this parameter as well, to expand the reference frame in which *runq_dl_nonpriv* and *runq_dl_use* act.

slice-size

Description of slice-size

slice-size is the default process time slice, expressed as a number of ticks of the system clock. The frequency of the system clock is expressed by the constant Hz, which has a value of 100. Thus each unit of *slice-size* corresponds to 10 milliseconds. When a process is given control of the CPU, the kernel lets it run for *slice-size* ticks. When the time slice expires or when

the process voluntarily gives up the CPU (for example, by calling *pause*(2) or by doing some other system call that causes the process to wait), the kernel examines the run queue and selects the process with the highest priority that is eligible to run on that CPU.

The *slice-size* parameter is defined in */var/sysgen/mtune/disp* and has the following formula:

```
#define slice-size Hz / 30 int slice_size = slice-size
```

Since *slice_size* is an integer, the resulting value is 3. This means that the default process time slice is 30 milliseconds.

Value of slice-size

Default: 3

Range: 1–100

When to Change slice-size

If you use the system primarily for compute-intensive jobs and interactive response is not an important consideration, you can increase *slice-size*. For example, setting *slice-size* to 10 gives greater efficiency to the compute jobs, since each time they get control of the CPU, they are able to run for 100 milliseconds before getting switched out.

In situations where the system runs both compute jobs and interactive jobs, interactive response time will suffer as you increase the value of *slice-size*.

EFS Parameters

The IRIX Extent File System works closely with the operating system kernel, and the following parameters adjust the kernel's interface with the file system.

The following parameters are defined in the *efs* group:

- *efs_bmmax* – The number of efs bitmap buffers to keep cached.

The following parameters are set in the *kernel* parameter group, and are used for file system tuning. They determine how many pages of memory are clustered together into a single disk write operation. Adjusting these parameters can greatly increase file system performance. Available parameters include:

- *dwcluster* – number of delayed-write pages to cluster in each push.
- *autoup* – specifies the age, in seconds, that a buffer marked for delayed write must be before the *bdflush* daemon writes it to disk.

efs_bmmax

Description of efs_bmmax

This parameter represents the number of *efs* bitmap buffers to keep privately cached at any given time.

Value of efs_bmmax

Default: 10 buffers

When to Change efs_bmmax

It is not generally useful to change this parameter, but you may want to increase it over the default for systems with 10 gigabytes of disk space or more and a great deal of file system activity.

dwcluster

Description of dwcluster

This parameter sets the maximum number of delayed-write pages to cluster in each push.

Value of dwcluster

Default: 64

When to Change dwcluster

It should not be necessary to change this parameter. The automatically configured value is sufficient.

autoup

Description of autoup

The *autoup* parameter specifies the age, in seconds, that a buffer marked for delayed write must be before the *bdflush* daemon writes it to disk. This parameter is specified in */var/sysgen/mtune.* For more information, see the entry for the *bdflushr* kernel parameter.

Value of autoup

Default: 10

Range: 1–30

When to Change autoup

This value is adequate for most systems.

Loadable Drivers Parameters

IRIX 5.0 allows you to load and run device drivers while the system remains up and running. Occasionally, you may have to make adjustments to the running kernel to allow for the extra resources these loadable drivers require. The following parameters allow you to make the necessary adjustments:

- *bdevsw_extra* – specifies an extra number of entries in the block device switch.
- *cdevsw_extra* – specifies an extra number of entries in the character device switch.
- *fmodsw_extra* – specifies an extra number of entries in the streams module switch.

- *vfssw_extra* – specifies an extra number of entries in the virtual file system module switch.

bdevsw_extra

Description of bdevsw_extra

This parameter specifies an extra number of entries in the block device switch. This parameter is for use by loadable drivers only. If you configured a block device into the system at *lboot*(1M) time, you will not need to add extra entries to **bdevsw**.

Value of bdevsw_extra

Default: 21

Range: 1-254

When to Change bdevsw_extra

Change this parameter when you have more than 21 block devices to load into dynamically the system. IRIX provides 21 spaces in the **bdevsw** by default.

cdevsw_extra

Description of cdevsw_extra

This parameter specifies an extra number of entries in the character device switch. This parameter is for use by loadable drivers only. If you configured a character device into the system at *lboot*(1M) time, you will not need to add extra entries to **cdevsw**.

Value of cdevsw_extra

Default: 23

Range: 3-254

When to Change cdevsw_extra

Change this parameter when you have more than 23 character devices to load dynamically into the system. IRIX provides 23 spaces in the **cdevsw** by default.

fmodsw_extra

Description of fmodsw_extra

This parameter specifies an extra number of entries in the streams module switch. This parameter is for use by loadable drivers only. If you configured a streams module into the system at *lboot*(1M) time, you will not need to add extra entries to **fmodsw**.

Value of fmodsw_extra

Default: 20

Range: 0-254

When to Change fmodsw_extra

Change this parameter when you have more than 20 streams modules to load dynamically into the system. IRIX provides 20 spaces in the **fmodsw** by default.

vfssw_extra

Description of vfssw_extra

This parameter specifies an extra number of entries in the *vnode* file system module switch. This parameter is for use by loadable drivers only. If you configured a *vfs* module into the system at *lboot*(1M) time, you will not need to add extra entries to **vfssw**.

Value of vfssw_extra

Default: 5

Range: 0-254

When to Change vfssw_extra

Change this parameter when you have more than 5 virtual file system modules to load dynamically into the system. IRIX provides 5 spaces in the **vfssw** by default.

CPU Actions Parameters

CPU actions parameters are used in multi-processor systems to allow the user to select the processor or processors that will be used to perform a given task.

The following parameters are defined:

- *nactions* – specifies the number of action blocks.

nactions

Description of nactions

The *nactions* parameter controls the number of action blocks. An action block lets you queue a process to be run on a specific CPU. The value of the *nactions* parameter is found by the formula:

maxcpu + 60

Value of nactions

Default: 0 (Auto-configured if set to 0)

Range: 60-200

When to Change nactions

Increase the value of *nactions* if you see the kernel error message:

PANIC: Ran out of action blocks

Switch Parameters

The following parameters are simple on/off switches within the kernel that allow or disallow certain features, such as whether shells that set the user ID to the superuser are allowed:

- *svr3pipe* – controls whether SVR3.2 or SVR4 pipes are used.
- *nosuidshells* – when set to 0, allows applications to create superuser-privileged shells. When set to any value other than 0, such shells are not permitted.
- *posix_tty_default* – if the value of this switch is 0, the default Silicon Graphics line disciplines are used. If the value is set to 1, POSIX line disciplines and settings are used.
- *resettable_clocal* – allows you to use either the default behavior or POSIX compliant behavior.
- *restricted_chown* – allows you to decide whether you want to use BSD UNIX style *chown*(2) system call or the System V style.
- *force_old_dump* – When set to 1, forces the system to use old-style core dump formats, rather than the new IRIX 5 format.
- *use_old_serialnum* – When set to 1, forces the kernel to use the old method of calculating a 32-bit serial number for *sysinfo -s*. This variable affects only Onyx and Challenge L or XL systems.

Note that all the above listed parameters are enforced system-wide. It is not possible to select different values on a per-process basis.

svr3pipe

Description of svr3pipe

This parameter, when set to 1, specifies SVR3.2 style pipes, which are unidirectional. When set to 0, SVR4 style pipes are specified, which are bidirectional.

Value of svr3pipe

Default: 1 (SVR3.2 style pipes)

Range: 0 or 1

When to Change svr3pipe

Change this parameter if you wish to take advantage of SVR4 style pipes.

nosuidshells

Description of nosuidshells

Some programs are written so that they perform actions that require superuser privilege. In order to perform these actions, they create a shell in which the user has superuser privilege. Such shells pose a certain manageable risk to system security, but application developers are generally careful to limit the actions taken by the application in these shells. The *nosuidshells* switch, when set to 0, allows these applications to create superuser-privileged shells. When set to any value other than 0, such shells are not permitted.

Value of nosuidshells

Default: 1 (setuid shells not permitted)

When to Change nosuidshells

Change this switch to allow setuid shells.

posix_tty_default

Description of posix_tty_default

IRIX uses a default system of line disciplines and settings for serial lines. These default settings are different from those specified by POSIX. If the value of this switch is 0, the default Silicon Graphics line disciplines are used. If the value is set to 1, POSIX line disciplines and settings are used.

Value of posix_tty_ default

Default: 0

Range: 0 or 1

When to Change posix_tty_default

Change this switch if you need to use POSIX line disciplines.

resettable_clocal

Description of resettable_clocal

In the standard configuration, the CLOCAL environment variable on a tty is read-only, but under POSIX, CLOCAL can be reset. This switch allows you to use either the default behavior or POSIX compliant behavior.

Value of resettable_clocal

Default: 0

Range: 0 or 1

When to Change resettable_clocal

Change this switch if you need POSIX compliance in your tty handling.

restricted_chown

Description of restricted_chown

This switch allows you to decide whether you want to use a BSD UNIX style *chown*(2) system call or the System V style. Under the BSD version, only the Superuser can use the *chown* system call to ''give away'' a file – to change the ownership to another user. Under the System V version, any user can give away a file or directory. If the value of the switch is 0, System V *chown* is enabled. If the value is not zero, BSD *chown* is enabled.

Value of restricted_chown

Default: 0

Range: 0 or 1

When to Change restricted_chown

Change this switch to choose which behavior you prefer for the *chown*(2) system call.

force_old_dump

Description of force_old_dump

When set to 1, this parameter forces the system to use old-style core dump formats, rather than the new IRIX 5 format.

Value of force_old_dump

Default: 0

Range: 0 or 1

When to Change force_old_dump

This parameter is for use when the new form of compressed dumps are inadequate.

use_old_serialnum

Description of use_old_serialnum

When set to 1, this parameter forces the kernel to use the old method (before IRIX Version 5) of calculating a 32-bit serial number for *sysinfo -s*. This variable affects only Onyx and Challenge L or XL systems.

Value of use_old_serialnum

Default: 0

Range: 0 or 1

When to Change use_old_serialnum

Change this parameter on your Challenge or Onyx system if you need to use some older software that requires a 32-bit serial number.

Timer parameters

Timer parameters control the functioning of system clocks and timing facilities. The following parameters are defined:

- *fasthz* – sets the profiling/fast itimer clock speed.
- *itimer_on_clkcpu* – determines whether *itimer* requests are queued on the clock processor or on the running processor, respectively.
- *timetrim* – the system clock is adjusted every second by the signed number of nanoseconds specified by this parameter.

fasthz

Description of fasthz

This parameter is used to set the profiling/fast *itimer* clock speed.

Value of fasthz

Default: 1000

Range: 500-2500

When to Change fasthz

Change this parameter to give a finer or coarser grain for such system calls as *gettimeofday*(3B), *getitimer*(2) and *settimer*(2).

itimer_on_clkcpu

Description of itimer_on_clkcpu

This parameter is set to either 0 or 1, to determine whether *itimer* requests are queued on the clock processor or on the running processor, respectively.

Value of itimer_on_clkcpu

Default: 0

Range: 0 or 1

When to Change itimer_on_clkcpu

If a process uses the *gettimeofday*(2) call to compare the accuracy of the *itimer* delivery, then you should set this parameter to 1, to take advantage of the clock processor. If the *itimer* request is for the purpose of implementing a user frequency-based scheduler, then set this parameter to 0 to queue the requests on the current running processor.

timetrim

Description of timetrim

The system clock is adjusted every second by the signed number of nanoseconds specified by this parameter. This adjustment is limited to 3 milliseconds or 0.3%. *timed*(1M) and *timeslave*(1M) periodically place suggested values in */var/adm/SYSLOG*.

Value of timetrim

Default: 0

Range: 0-0.3% of a second (3 milliseconds)

When to Change timetrim

Change this parameter as suggested by *timed* and *timeslave*.

NFS Parameters

The following parameters control the kernel-level functions of the Network File System (NFS). Reducing these values is likely to cause significant performance decreases in your system:

- *nfs_portmon* – set to 0, clients may use any available port. If it is set to 1, clients must use only privileged ports.
- *first_timeout* – sets the *portmapper* query timeout.
- *normal_timeout* – sets the *lockd* ping timeout.
- *working_timeout* – sets the *lockd* requests timeout.
- *svc_maxdupregs* – sets the number of cached NFS requests.

nfs_portmon

Description of nfs_portmon

This parameter determines whether or not a client must use a "privileged" port for NFS requests. Only processes with superuser privilege may bind to a privileged port. The *nfs_portmon* parameter is binary. If it is set to 0, clients may use any available port. If it is set to 1, clients must use only privileged ports.

Value of nfs_portmon

Default: 0

Range: 0 or 1

When to Change nfs_portmon

You should change this parameter only if it is absolutely necessary to maintain root privilege on your NFS mounted file systems and you have checked each NFS client to be sure that it requests a privileged port. If there are any clients requesting non-privileged ports, they will be unable to mount the file systems.

Additionally, changing the value of *nfs_portmon* to 1 can give a false sense of security. A process must have *root* privilege in order to bind to a privileged port, but a single "insecure" machine compromises the security of this privilege check.

first_timeout

Description of first_timeout

This parameter determines the length of time before a portmapper request is retransmitted.

Value of first_timeout

Default: 1

Range: 1 or 2

When to Change first_timeout

Change this value to 2 if your portmapper requests are consistently being timed out. Decreasing this value can seriously impede system performance.

normal_timeout

Description of normal_timeout

This parameter determines the time before a *ping* request times out and is sent again.

Value of normal_timeout

Default: 5

Range: 1-5

When to Change normal_timeout

Increase this value if your portmapper requests are consistently being timed out. Decreasing this value can seriously impede system performance.

working_timeout

Description of working_timeout

This parameter set the time allowed before a *lockd* request times out and is sent again.

Value of working_timeout

Default: 5

Range: 1-5

When to Change working_timeout

Increase this value if your portmapper requests are consistently being timed out. Decreasing this value can seriously impede system performance.

svc_maxdupregs

Description of svc_maxdupregs

This parameter sets the number of cached NFS requests.

Value of svc_maxdupregs

Default: 1024

Range: 400-4096

When to Change svc_maxdupregs

This parameter should be adjusted to the service load so that there is likely to be a response entry when the first retransmission comes in.

UDS Parameters

Under UNIX domain sockets, there is a pair of buffers associated with each socket in use. There is a buffer on the receiving side of the socket, and on the sending side. The size of these buffers represent the maximum amount of data that can be queued. The behavior of these buffers differs depending on whether the socket is a streams socket or a data-gram socket.

On a streams socket, when the sender sends data, the data is queued in the receive buffer of the receiving process. If the receive buffer fills up, the data begins queueing in the sendspace buffer. If both buffers fill up, the socket blocks any further data from being sent.

Under data-gram sockets, when the receive buffer fills up, all further data-grams sent are discarded and the error EWOULDBLOCK is generated. Because of this behavior, the default receive buffer size for data-gram sockets is twice that of the send buffer.

The following parameters control UNIX domain sockets (UDS):

- *unpst_sendspace* – UNIX domain socket stream send buffer size.
- *unpst_recvspace* – UNIX domain socket stream receive buffer size.

- *unpdg_sendspace* – UNIX domain socket data-gram send buffer size.
- *unpdg_recvspace* – UNIX domain socket data-gram receive buffer size.

unpst_sendspace

Description of unpst_sendspace

This parameter controls the default size of the *send* buffer on streams sockets.

Value of unpst_sendspace

Default: 0x4000 (16KB)

When to Change unpst_sendspace

It is generally recommended that you change the size of socket buffers individually, since changing this parameter changes the send buffer size on all streams sockets, using a tremendous amount of kernel memory. Also, increasing this parameter increases the amount of time necessary to wait for socket response, since all sockets will have more buffer space to read.

See the *setsockopt*(2) reference page for more information on setting specific socket options.

unpst_recvspace

Description of unpst_recvspace

This parameter controls the default size of the receive buffer of streams sockets.

Value of unpst_recvspace

Default: 0x4000 (16 Kbytes)

When to Change unpst_recvspace

It is generally recommended that you change the size of socket buffers on an individual basis, since changing this parameter changes the receive buffer size on all streams sockets, using a tremendous amount of kernel memory. Also, increasing this parameter will increase the amount of time necessary to wait for socket response, since all sockets will have more buffer space to read.

See the *setsockopt*(2) reference page for more information on setting specific individual socket options.

unpdg_sendspace

Description of unpdg_sendspace

This parameter controls the size of a data-gram that can be sent over a socket.

Value of unpdg_sendspace

Default: 0x2000 (8 KB)

When to Change unpdg_sendspace

Data-gram sockets operate slightly differently from streams sockets. When a streams socket fills the receive buffer, all data is then sent to the send buffer, and when the send buffer fills up, an error message is issued. Data-gram sockets allow data-grams to fill the receive buffer, and when the receive buffer is full, all future data-grams are discarded, and the error EWOULDBLOCK is generated. Therefore, the *unpdg_sendspace* parameter serves only to limit the size of a data-gram to not more than can be received by the receive buffer.

Note that the default data-gram socket receive buffers are twice the size of the default data-gram send buffers, thus allowing the process to appear the same as the streams sockets.

It is generally recommended that you not change the size of this parameter without also changing the default receive buffer size for data-gram sockets.

If you raise the value of this parameter (*unpdg_sendspace*) without raising the receive buffer size (*unpdg_recvspace*), you will allow the sending half of the socket to send more data than can be received by the receiving half. Also it is generally recommended that socket buffer sizes be set individually via the *setsockopt*(2) system call. See the *setsockopt*(2) reference page for more information on setting specific individual socket options.

unpdg_recvspace

Description of unpdg_recvspace

This parameter controls the default size of data-gram socket receive buffers.

Value of unpdg_recvspace

Default: 0x4000 (16 Kbytes)

When to Change unpdg_recvspace

It is generally recommended that you change the size of socket buffers individually, since changing this parameter changes the receive buffer size on all data-gram sockets, using a tremendous amount of kernel memory. Also, increasing this parameter increases the amount of time necessary to wait for socket response, since all sockets will have more buffer space to read.

See the *setsockopt*(2) reference page for more information on setting specific individual socket options.

Appendix B

IRIX Device Files

This appendix contains a list of device files and directories that reside in the */dev* directory.

dsk/	Directory containing block device files for disks; see *ips*(7), *dks*(7), and *xyl*(7) for disk partition device names.
rdsk/	Directory containing raw (character) device files for disks; see *ips*(7), *dks*(7), and *xyl*(7) for disk partition device names.
root	Generic *root* partition (block device).
rroot	Generic *root* partition (raw device).
usr	Generic *usr* partition (block device).
rusr	Generic *usr* partition (raw device).
swap	Generic *swap* partition (block device).
rswap	Generic *swap* partition (raw device).
vh	Generic *root* volume header (block device).
rvh	Generic *root* volume header (raw device).
mt/	directory containing block device files for tapes; see *ts*(7) for ISI quarter-inch tape drive device names; see *tps*(7) for SCSI quarter-inch tape drive device names; see *xmt*(7) for Xylogics half-inch tape drive names.
rmt/	directory containing raw device files for tapes; see *ts*(7) for ISI quarter-inch tape drive device names; see *tps*(7) for SCSI quarter-inch tape drive device names; see *xmt*(7) for Xylogics half-inch tape drive names.
tape	Generic tape device; bytes are swapped in order to be backward-compatible with the IRIS Series 2000 and 3000 workstations; see *mtio*(7).

nrtape	Generic no-rewind tape device; bytes are swapped in order to be backward-compatible with the IRIS Series 2000 and 3000 workstations; see *mtio*(7).
tapens	Generic tape device; bytes are not swapped; see *mtio*(7).
nrtapens	Generic no-rewind tape device; bytes are not swapped; see *mtio*(7).
mem	Memory; see *mem*(7).
mmem	Mappable memory; see *mmem*(7).
kmem	Kernel memory; see *kmem*(7).
null	Null device (zero length on input, data sink on output); see *null*(7).
SA/	Block devices used by system administration tools; see *sysadm*(1M) and *sa*(7).
rSA/	Raw devices used by system administration tools; see *sysadm*(1M) and *sa*(7).
audio	Audio interface.
dn_ll	File used to create 4DDN logical links; see *dn_ll*(7).
dn_netman	File used by 4DDN network management software; see *dn_netman*(7).
cent	Centronics® color graphics printer device.
tek	Tektronics color graphics printer device.
vers	Versatek color graphics printer device.
vp0	Hard link to *vers*.
*gpib**	GPIB (IEEE-488) device; see *gpib*(7).
gse	Spectragraphics coax device; see *gse*(7).
plp	Parallel line printer interface; see *plp*(7).
prf	File used by operating system profiler; see *prf*(7).
t3270	Raw device file for IBM 3270™ Cluster Controller; see *t3270*(7).

hl/	Directory containing files used by IRIS GTX series machines hardware spinlock driver; see *usnewlock*(3P).
log	Named pipe that is read by the system logging daemon; see *syslogd*(1M).
ptc	Clonable pseudo-tty controller; see *clone*(7), *ptc*(7).
grconc	Master pseudo-teletype for the graphics console; see *pty*(7).
grcons	Slave pseudo-teletype for the graphics console; see *pty*(7).
gm	Logical console device for the Graphics Manager on the IRIS GT and GTX model machines. Messages from the software running on the 68020 on the GM board will appear as output on this device.
grin/	Directory containing the individual logical graphics input devices.
console	System console device.
syscon	Hard link to */dev/console*.
systty	Hard link to */dev/console*.
queue	Graphics queue device. Graphics programs call ''select'' on this device in order to be notified when there is input in their graphics queue. This device can't be actually read or written.
dials	Device for serial port connected to dial and button box.
keybd	Device for serial port connected to keyboard.
mouse	Device for serial port connected to mouse.
tablet	Device for serial port connected to digitizing tablet.
ttyd[*1-12*]	Serial ports 1–12.
ttyf[*1-12*]	Serial ports 1–12 for devices that understand hardware flow control.
ttym[*1-12*]	Serial ports 1–12 for modems.
*ttyq**	Pseudo tty devices; see *pty*(7).
zero	Zero device (infinite zeros on reads); see *zero*(7).

Appendix C

IRIX Kernel Error Messages

This appendix lists error messages you may receive from the IRIX operating system. The operating system handles error conditions according to their degree of severity and divides the error messages into three classes: NOTICE, WARNING, and PANIC.

- NOTICE messages provide information on the system status. These messages can sometimes help you anticipate problems before trouble occurs. See "NOTICE Messages" on page 832.
- WARNING messages indicate that the system might stop functioning unless you take corrective action. See "WARNING Messages" on page 833.
- PANIC messages indicate that a problem is severe enough that the system comes to a halt. The cause is usually a hardware problem or a problem in the kernel software. This type of message occurs occasionally; it should not cause much concern. If a particular PANIC message occurs repeatedly or predictably, however, contact your service representative. See "PANIC Messages" on page 833.

The error messages in this appendix are listed alphabetically in their respective classes. The messages that follow relate to tunable parameters. For other error messages, see the file */usr/include/sys/errno.h,* the *intro*(2) reference page, and the *Owner's Guide* for your system.

NOTICE Messages

- diskname: Swap out failed blkno x (page still in memory)

  A swap write failed. No action is necessary as the page is still in memory.

- Insufficient memory to lock/allocate # pages

  There is not enough memory available to open the requested pages.

- system call failed for process name

  There is insufficient free memory to execute the system call; reduce the system load and try again, or add more memory.

- "str" - swpuse count overflow

  More than 256 processes are sharing the same page of swap. A copy has been made. No action is required.

- useracc - no memory to lock page

  There is insufficient free memory to lock a user data page in memory to service a read or write system call to a raw device. Reduce the system load, reduce the size of the raw I/O buffer in the user program, or add more memory.

WARNING Messages

- iget - inode table overflow

  Each active file in the system requires an entry in the inode table, and this table has overflowed. Reduce the system load. If this happens frequently, increase the tunable parameter *ninode* using *systune*(1M) and follow any directions given by that utility to activate the change.

- mfree map overflow

  Fragmentation of some resource (such as message queues) contributed to the loss of some of the resource. No action necessary.

- out of physical memory, nbytes=#

  No more physical pages are available. Reduce the system load or add more memory.

- Paging Daemon (vhand) not running - NFS server down?

  The system determines that *vhand* is not executing, possibly because it is waiting for an I/O transfer to complete to an NFS server (especially if the NFS file system is hand mounted). No action should be necessary as the system will restart *vhand* when needed.

PANIC Messages

- bumpcnt - region count list overflow

  Indicates an unresolvable problem with the operating system. Reboot your system.

- getpages - pbremove

  Indicates an unresolvable problem with the operating system. Reboot your system.

- Ran out of action blocks

  A resource used by the multiprocessor kernel for inter-CPU interrupts has run out. If this happens frequently, use the *systune*(1M) command to increase the value of the parameter *nactions*.

- vfault - bad dbd_type

  The page being faulted in is not a recognized type: either demand fill, demand zero, in file or on swap (operating system error). Reboot your system and if the error persists, check your application and your disk.

Appendix D

IRIX sendmail Reference

The *sendmail* system implements a general purpose internetwork mail-routing facility under the UNIX operating system. It is not tied to any one transport protocol; its function is like a crossbar switch, relaying messages from one domain into another. In the process, it can also do a limited amount of message header editing to put the message into a format that is appropriate for the receiving domain. All *sendmail* operations are done under the control of a configuration file.

To provide customized services to each site, *sendmail* uses a configuration file that provides a high degree of flexibility. The unfortunate result is that the configuration file can seem complex and unapproachable. However, most sites fit one of a few basic configurations. The standard configuration file supplied with IRIX is designed to work in these configurations with a minimum number of site-specific modifications.

This appendix is organized into the following sections:

- "Overview" on page 836 introduces *sendmail.*
- "Basic Installation" on page 848 explains how to install *sendmail*.
- "Normal Operations" on page 849 provides information for day-to-day maintenance of your mail system. If your site has a relatively simple mail environment, this section may contain sufficient information for you to keep *sendmail* working correctly.
- "sendmail Command-Line Flags" on page 860 and "Tuning" on page 863 describe features used to monitor or adjust the operation of *sendmail*.
- "The Configuration File" on page 868 contains detailed information about the configuration file. Read this section if you will be writing your own configuration file.
- "Flags, Options, and Files" on page 887 provides complete lists of configuration options, support files, and command line, mailer, and debugging flags.

Overview

sendmail's implementation features aliasing, forwarding, automatic routing to network gateways, and flexible configuration.

In a simple network, each node has an address, and resources can be identified with a host-resource pair. For example, a mail system can refer to users with a host–user-name pair. Host names and numbers must be administered by a central authority, but user names can be assigned locally to each host.

In an internetwork, multiple networks with different characteristics and management must communicate. In particular, the syntax and semantics of resource identification change. You can handle certain simple cases by using improvised techniques, such as providing network names that appear local to hosts on other networks. However, the general case is extremely complex. For example, some networks require point-to-point routing, which simplifies the database update problem, because only adjacent hosts are entered into the system tables; others use end-to-end addressing. Some networks use a left-associative syntax; others use a right-associative syntax, causing ambiguity in mixed addresses.

Internetwork standards seek to eliminate these problems. Initially, these standards proposed expanding the address pairs to address triples, consisting of {network, host, resource}. Network numbers must be universally agreed upon; hosts can be assigned locally on each network. The user-level presentation was quickly expanded to address domains, composed of a local resource identification and a hierarchical domain specification with a common static root, as defined in RFC 1034. The domain technique separates the issue of physical versus logical addressing. For example, an address of the form "jane@iris1.company.com" describes only the logical organization of the address space.

sendmail bridges the gap between the world of totally isolated networks that know nothing of each other and the clean, tightly coupled world of unique network numbers. *sendmail* can accept old arbitrary address syntaxes, resolving ambiguities by using heuristics specified by the network administrator, as well as domain-based addressing. *sendmail* helps guide the conversion of message formats between disparate networks. In short, *sendmail* is designed to assist a graceful transition to consistent internetwork addressing schemes.

Design Goals

The design goals for *sendmail* included the following:

1. Message delivery should be reliable, guaranteeing that every message is correctly delivered or at least brought to the attention of a human for correct disposal; no message should ever be completely lost.
2. Existing software should be used to do actual message delivery whenever possible.
3. *sendmail* should be easy to expand to fairly complex environments.
4. Configuration should not be compiled into the code.
5. *sendmail* should let various groups maintain their own mailing lists, and let individuals specify their own forwarding, without modifying the system alias file.
6. Each user should be able to specify which mailer to execute to process mail being delivered for him. This feature allows users with specialized mailers that use a different format to build their environments without changing the system, and facilitates specialized functions (such as returning an “I am on vacation” message).
7. To minimize network traffic, addresses should be batched to a single host where possible, without assistance from the user.

System Organization

The original design goals for *sendmail* motivated the architecture illustrated in Figure D-1.

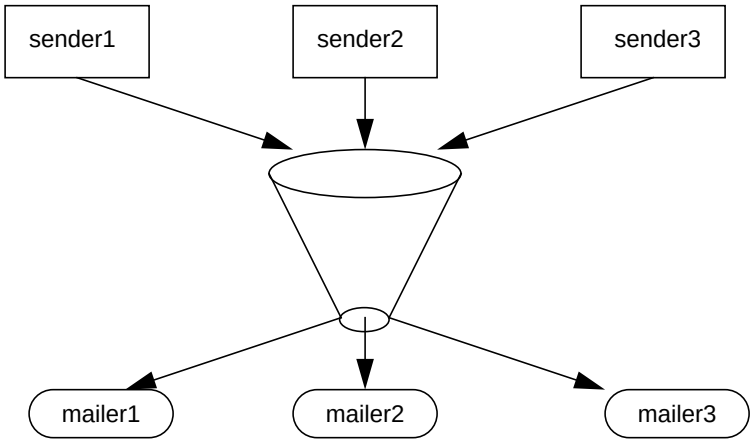


Figure D-1 *sendmail* System Structure

sendmail neither interfaces with the user nor does actual mail delivery. Rather, it collects a message generated by a user agent program such as Berkeley *Mail*, edits the message as required by the destination network, and calls appropriate mailers to do mail delivery or queueing for network transmission. The exception is mail sent to a file; in this case, *sendmail* delivers the mail directly.

This discipline allows the insertion of new mailers at minimum cost.

Because some of the senders may be network servers and some of the mailers may be network clients, *sendmail* can be used as an internetwork mail gateway.

sendmail Communications

sendmail communicates with the outside world to receive and send mail in any of three ways:

- by using the conventional UNIX argument vector/return status

With this standard UNIX technique, a list of recipients is sent in the argument vector and the message text is sent on the standard input. If problems occur, anything that the mailer prints is simply collected and sent back to the sender. The exit status from the mailer is collected after the message is sent and a diagnostic is printed if appropriate.

- by speaking SMTP over a pair of UNIX pipes

  The SMTP protocol defined in RFC 821 runs an interactive lock-step interface with the mailer. A subprocess is created, but no recipient addresses are passed to the mailer by means of the argument list. Instead, they are passed one at a time in commands sent to the process's standard input. Anything appearing on the standard output must be a reply code in a special format.

- by speaking SMTP over sockets

  This technique is similar to the previous technique, except that it uses the BSD socket mechanism and is exceptionally flexible. The mailer need not reside on the same machine as *sendmail,* but simply connects to a *sendmail* process on another machine.

How sendmail Works

To send a message, the sender program issues a request to *sendmail* by using one of the methods described in the preceding subsection. *sendmail* operates in two distinct phases. First, it collects and stores the message. Second, it sees that the message is delivered. If errors occur during the second phase, *sendmail* returns to the sender either a status code or a new message describing the error.

Argument Processing and Address Parsing

When *sendmail* is called, initial processing includes reading *sendmail.cf*, scanning arguments and processing option specifications. Next, *sendmail* collects recipient addresses, either from the command line or from the SMTP RCPT command, and creates a list of recipients. This step includes expanding alias and mailing lists. *sendmail* also tries to validate addresses by checking syntax and verifying local addresses. However, detailed checking of host names and addresses occurs during the delivery phase.

As local addresses are verified, *sendmail* performs any system aliasing or per-user forwarding (described in "Per-User Forwarding" on page 858). *sendmail* appends each address to the recipient list after parsing. When a name is aliased or forwarded, the old name is retained in the list and a flag is set that tells the delivery phase to ignore this recipient. This list is kept free from duplicates to prevent alias loops and duplicate messages to the same recipient, which might occur if a person is a member of two groups.

Message Collection

After initial processing, *sendmail* collects the message. The message should begin with a header; no other formatting requirements are imposed on the message except that it must be composed of text lines. (That is, *sendmail* does not accept binary data.) The message header is parsed and stored in memory; the message body (the text lines of the actual message) is saved in a temporary file.

To simplify the program interface, the message is collected even if no addresses are valid. In this case, the message is returned with an error.

Message Delivery

For each unique mailer and host in the recipient list, *sendmail* calls the appropriate mailer. Each mailer invocation sends to all users receiving the message on one host. Mailers that accept only one recipient at a time are handled properly.

The message is sent to the mailer by means of one of the same three interfaces used to submit a message to *sendmail.* Each copy of the message is prepended by a customized header. The mailer status code is caught and checked, and an error message given as appropriate. If the exit code does not conform to a system standard, *sendmail* sends a generic message ("Service unavailable").

Queueing for Retransmission

If the mailer returns a status indicating that it might be able to handle the mail later, *sendmail* queues the mail and tries again later.

Return to Sender

If errors occur during processing, *sendmail* returns the message to the sender for retransmission. The message can be mailed back or written in the file *dead.letter* in the sender's home directory. (Obviously, if the site giving the error is not the originating site, the only reasonable option is to return the message to the sender. There are many error disposition options, but they only affect the error message. The "return to sender" function always operates in one of these two ways.)

Message Header Editing

Some editing of the message header occurs automatically under the control of the configuration file. Header lines may be inserted and addresses rewritten to conform to the requirements of the receiving domain.

See "The Configuration File" on page 868 for more information on the configuration file.

Usage and Implementation

This section provides a more in-depth discussion of several topics discussed at the operational level in the preceding section.

Arguments

sendmail command arguments include both flags and addresses. Flags appear first and set various processing options. Address arguments are valid unless *sendmail* is running in SMTP mode, and follow the address syntax in RFC 822 for Internet address formats:

1. Anything in parentheses is ignored (treated as a comment).
2. Anything in angle brackets (< >) is preferred over anything else. This rule implements the Internet standard of choosing bracketed over nonbracketed addresses in cases like the following:

   `user name <`*machine-address* `>`

   In this example the electronic "machine-address" will be the recipient of the message rather than the human "user name."

3. Double quotation marks (") are used to quote a phrase; a backslash (\) is used to quote a character. Backslashes are more powerful: They cause otherwise equivalent phrases to compare differently. For example, *user* and "*user*" are equivalent, but *user* is different from either of the first two forms.

Parentheses, angle brackets, and double quotation marks must be properly balanced and nested. The rewriting rules control the rest of the parsing, except that some special processing occurs after local names are rewritten. This processing is described in the next subsection.

Message Redirection

sendmail uses three facilities to redirect mail: aliasing, forwarding, and inclusion. Aliasing applies system-wide. Forwarding allows each user to redirect incoming mail sent to his account. Inclusion directs *sendmail* to read a file for a list of addresses, and is normally used in conjunction with aliasing.

Aliasing

Aliasing maps names to address lists, using a system-wide file. This file is indexed to speed access. Only names that parse as local are allowed as aliases; this restriction guarantees a unique key (since there are no nicknames for the local host). You can suppress aliasing by prepending a backslash to the address. See "The Alias Database" on page 855 for more information.

Forwarding

If aliasing determines that a name is local and valid, the recipient's home directory is checked for the existence of a *.forward* file. If one exists, the message is *not* sent to the user, but rather to the list of users identified in the *.forward* file. The file may contain only one address; in this case, the feature implements network mail forwarding. See "Per-User Forwarding" on page 858 for more information.

Inclusion

Inclusion is specified in RFC 733 syntax:

`:include:` *pathname*

An address of this form reads the file specified by *pathname* and sends the message to all users listed in that file. RFC 822, which defines Internet addressing, does not define inclusion; therefore it should not appear in any message address field. Silicon Graphics does not support direct use of this feature, but uses it as a subset of aliasing. For example, an alias of the following form allows a project to maintain a mailing list without involving the system administration, even if the alias file is protected:

```
project: :include:/usr/project/userlist
```

When you change the contents of an :include: list, you do not have to rebuild the indexed alias database. (You will, however, have to rebuild the indexed alias database if the *pathname* changes.)

Mail to Files and Programs

A message can be sent to a file or program residing on the local host if the file or program name is the result of an alias lookup or is obtained from a user's *.forward* file.

Files provide archival storage of messages and are useful for project administration and history. Programs are useful recipients in a variety of situations. The *vacation*(1) program, which automatically responds to messages while you are on vacation, is an example of a recipient program.

If the initial parsing algorithm determines that an address is local (that is, that the address is not valid for another mailer), the address is scanned for two special cases:

- If prefixed by a vertical bar (`|`), the address is processed as a shell command.
- If prefixed by a slash (`/`), the address is processed as a file name instead of a login name.

In general, file-type recipients must be writable by everyone. However, if *sendmail* is running as *root* and the file has *setuid* or *setgid* bits set, then the message will be written to the file.

Message Collection

After all recipient addresses are parsed and verified, the message is collected. The message comes in two parts, a message header and a message body, separated by a blank line, as defined by RFC 822.

The header is formatted as a series of lines of the form

field-name: field-value

Field-value can be split across lines if each subsequent line starts with a space or a tab. Some header fields have special internal meaning and appropriate special processing. Other headers simply pass through. Header fields such as time stamps may be added automatically.

The body is a series of text lines that is completely uninterpreted and untouched, except that an extra dot is added to any line beginning with a dot when transmitted over an SMTP channel. (The extra dot is stripped by the receiver.)

The message is then placed in the mail queue. The mail queue resides in the directory */var/spool/mqueue* and consists of two queue files for each message. The queue control file describes the list of recipients built during initial address parsing, as well as other parameters. This control file is a series of tagged lines with each line describing a sender, a recipient, or some other salient parameter of the message. The queue control file also contains the complete message header. Associated with each control file is an unstructured queue data file that contains the complete body of the message.

Message Delivery

After the message has been collected and placed in the mail queue, *sendmail* attempts to deliver it. To implement message batching, prior to transmission, the list of recipients is ordered by receiving host. Under control of the configuration file, *sendmail* selects the appropriate mailer to use for connecting to each receiving host in turn.

After a connection is established, *sendmail* makes the per-mailer changes to the header and sends the result to the mailer. If any mail is rejected by the mailer, a flag is set to invoke the return-to-sender function after all delivery completes.

The interface to the mailer uses one of the techniques described in "sendmail Communications" on page 838.

Each recipient address to which the message is sent is marked so that *sendmail* can safely rescan the list. Mail to files and programs is detected during the scan of the recipient list.

After the message has been delivered to all recipients (or errors reported for all recipients to whom the message could not be delivered), the message is removed from the mail queue.

Queueing for Retransmission

If the mailer returns a "temporary failure" exit status, the message is left in the mail queue for attempted delivery at a later time.

Configuration

Almost all configuration information is read at run time from an ASCII file named */usr/lib/sendmail.cf* . This file encodes options, header declarations, mailer declarations, trusted user declarations, message precedences, address-rewriting rules, macro definitions, and class definitions.

To eliminate the need to read the configuration file each time *sendmail* is called, thus improving performance, *sendmail* can save a copy of its data space after the configuration file has been parsed. This technique creates what is known as a "frozen" configuration file. Subsequent calls to *sendmail* can use this frozen configuration directly, greatly reducing startup time.

The subsections that follow briefly describe the information in the configuration file. For a full description of the syntax and semantics of the configuration file, see "The Configuration File" on page 868.

Options

The options that can be set from the configuration file include the pathnames of various support files, timeouts, default modes, and so forth.

See the description of the Set Option command in "The Syntax" on page 868 and "Configuration Options" on page 888 for further details.

Header Declarations

Header declarations tell *sendmail* the format of known header lines.

Built into *sendmail* is the knowledge of a few header lines, such as the "From:" and "Date" lines. Most configured headers are automatically inserted in the outgoing message if they don't exist in the incoming message. See the description of the Define Header command in "The Syntax" on page 868 for details.

Mailer Declarations

Mailer declarations tell *sendmail* what mailers are available. The definition specifies the internal name of the mailer, the pathname of the program to call, some flags associated with the mailer, associated and separate sender and recipient address-rewriting rules, and an argument vector to be used on the call. See the description of the Define Header command in "The Syntax" on page 868 for details.

Trusted User Declarations

Trusted user declarations tell *sendmail* the names of users who are allowed to override the sender-address setting in messages. This should not be confused with anything having to do with Trusted IRIX/B, which is a B1-level trusted (secure) operating system. See the description of the Define Trusted Users command in "The Syntax" on page 868.

Message Precedence

Message precedence is set by the configuration file and is used by *sendmail* to determine the message priority adjustment associated with various

possible contents of the "Precedence:" header field. See the description of the Define Precedence command in "The Syntax" on page 868 for details.

Address Rewriting Rules

The heart of address parsing in *sendmail* consists of the address-rewriting rule sets. Each rule set is an ordered list of pattern-replacement rules. Rule sets perform the two main functions of rewriting address strings and selecting the appropriate mailer for the message.

The mechanism for textual rewriting of address strings enables *sendmail* to edit addresses into different formats. For example, consider an address of the following form:

```
widgets!buddy
```

This address might be rewritten as follows to conform to the domain syntax:

```
buddy@widgets.UUCP
```

With appropriate rule sets, translations can also be made in the other direction.

The special rule set 0 selects the correct mailer to handle a message based upon the recipient's address. For example, consider the following address:

```
johndoe@big.company.com
```

This address is parsed into a (mailer, host, user) triple:

```
{ether, big.company.com, johndoe@big.company.com}
```

The triple selects the mailer to use to deliver the message, the name of the host to deliver the message to next, and the address of the recipient.

sendmail uses different rule sets to rewrite sender and recipient addresses. Each mailer can have specific sender and recipient address-rewriting rules associated with it. Using these rules, *sendmail* can function as an internetwork mail router. See "The Syntax" on page 868 for information.

Macros

sendmail uses macros in three ways:

- A macro can provide unstructured textual information, such as the name that *sendmail* uses to identify itself in error messages.
- A macro can provide information to *sendmail* for use in creating fields such as argument vectors to mailers (for example, the name of the sender and the host and user of the recipient).
- A macro that is not used internally can be used as shorthand in the configuration file.

See the description of the Define Macros command in "The Syntax" on page 868 for information.

Classes

Used in the address-rewriting rules, classes define sets of textual information against which to test patterns to determine whether they are members. For example, a class might contains all known names for the local host. The address-rewriting rules might use this class to test the recipient hostname, determining if it is actually just another name for the local host. See the description of the Define Classes command in "The Syntax" on page 868 for more detail.

Basic Installation

Installing *sendmail* is a two-step process:

1. Install the *sendmail* files into the appropriate places with the proper permissions.

   This installation happens automatically when you install IRIX.

2. Modify the */usr/lib/sendmail.cf* configuration file, the file that *sendmail* reads when it starts up.

   The configuration file describes mailers, tells *sendmail* how to parse addresses and rewrite message headers, and sets various *sendmail* options. The standard configuration file shipped with IRIX supports a wide variety of mail configurations and should require only minimal modification.

Note: The supplied */usr/lib/sendmail.cf* file does not work "out of the box." Some modifications are required before *sendmail* will work. The next section describes typical modifications.

See "Customizing the sendmail.cf File" on page 657 for instructions on tailoring *sendmail* to your specific mail environment.

Normal Operations

This sections describes how *sendmail* operates under normal conditions.

Starting and Stopping the sendmail Daemon

Under rare circumstances, a user may need to stop or start the *sendmail* daemon manually. For example, to implement changes to the configuration file, you must stop all running *sendmail* processes, "refreeze" the configuration file, and restart the *sendmail* daemon before the new configuration will take effect. To simplify the task of starting and stopping *sendmail,* IRIX provides a shell script called */etc/init.d/mail.*

This script takes a single argument, either "start" or "stop," which starts or stops the *sendmail* daemon respectively. You must be superuser (root) to use this script.

When */etc/init.d/mail* is called with the "start" argument, it verifies the existence and permissions of various required components of the mail environment. If a required component such as the */var/spool/mqueue* directory is missing, the script creates it. For more complex components such as */usr/lib/aliases,* the script exits with a message.

When the */etc/init.d/mail* script is called with the "stop" argument, it kills all running *sendmail* processes with a SIGTERM signal.

Note: System startup includes an automatic call to the */etc/init.d/mail* script with the *start* argument. If system startup runs in "verbose" mode (that is, */etc/chkconfig* verbose on), the following message appears, verifying that *sendmail* has been started: `Mailer daemons: sendmail` For more information, examine the */etc/init.d/mail* script.

Freezing the Configuration File

A "frozen" configuration file (*sendmail.fc*) is an image of *sendmail*'s data space upon reading in the configuration file. After the *sendmail.fc* file is created, it is used in place of */usr/lib/sendmail.cf.* This process improves startup speed. The regular configuration file is effectively ignored once a frozen configuration file exists.

The following command creates a "frozen" configuration file; the *-bz* option is required:

```
/usr/lib/sendmail -bz
```

Note: If */usr/lib/sendmail.fc* exists, changes to */usr/lib/sendmail.cf* are not honored until you rebuild */usr/lib/sendmail.fc*. A common mistake is forgetting to re-freeze the configuration after making modifications to */usr/lib/sendmail.cf. sendmail* ignores the newly modified */usr/lib/sendmail.cf* and continues to use the old frozen configuration. (However, if a */usr/lib/sendmail.fc* file exists when *sendmail* is restarted by means of the */etc/init.d/mail* script, a new frozen configuration is created automatically.)

The frozen configuration file is ignored if *sendmail i*s started with a *-C* flag or if *sendmail* detects that */usr/lib/sendmail.fc* is out of date.

Caution: *sendmail*'s heuristics for detecting that the frozen configuration is out of date are not strong and should not be trusted.

Error Logging

The *syslogd*(1M) program supports the error log. A large amount of information can be logged. The log is structured in a succession of levels. The lowest level includes only extremely strange or serious situations. The highest level includes even the most mundane and uninteresting events. As a convention, log levels above ten are considered useful only for debugging purposes. For a complete description of log levels, see "Log Level" on page 867.

The Mail Queue

The mail queue contains messages that could not be delivered immediately. The messages are stored in various queue files that exist under the */var/spool/mqueue* directory. Normally, a *sendmail* subdaemon processes the messages in this queue periodically, attempting to deliver each message. (The */etc/init.d/mail* script starts the *sendmail* daemon so that it will fork a subdaemon every 15 minutes to process the mail queue.)

Each time *sendmail* processes the queue, it reads and sorts the queue, and then attempts to run all jobs in order. When *sendmail* attempts to run a job, it first checks to see if the job is locked. If the job is locked, *sendmail* ignores the job.

On occasion, you may need to process the queue manually. For example, if a major host is down for a period of time, the queue may become clogged. Although *sendmail* ought to recover gracefully once the host comes up, performance may be unacceptably bad in the interim.

Printing the Queue

You can print the contents of the queue using the *mailq* command or by specifying the *-bp* flag to *sendmail.* The printout includes a listing of the queue IDs, the size of each message, the date the message entered the queue, and the sender and recipients.

Forcing the Queue

The *-q* flag (with no value) forces *sendmail* to process the queue. It is sometimes useful to use the *-v* flag (verbose) also when running the queue manually, as follows:

```
/usr/lib/sendmail -q -v
```

In verbose mode, *sendmail* displays the SMTP chatter with other hosts as well as messages indicating any delivery errors and final message disposition.

Because of the locking algorithm, it is impossible for one job to freeze the queue. However, an uncooperative recipient host or a program recipient that never returns can consume many system resources. Unfortunately, there is no way to resolve this without violating the SMTP protocol used by *sendmail*.

In some cases, if a major host goes down for a couple of days, a prohibitively large queue may be created. As a result, *sendmail*'s will spend an inordinate amount of time sorting the queue. You can remedy this situation by moving the queue to a temporary location and creating a new queue. The old queue can be run later when the offending host returns to service.

Use the following commands to move the entire queue directory:

```
cd /var/spool
mv mqueue omqueue
mkdir mqueue
chmod 755 mqueue
```

Then kill the existing *sendmail* daemon (because it will still be processing in the old queue directory) and create a new daemon:

```
/etc/init.d/mail stop
/etc/init.d/mail start
```

To run the old mail queue, use the following command:

```
/usr/lib/sendmail -oQ/var/spool/omqueue -q
```

The *-oQ* flag specifies an alternate queue directory and the *-q* flag says to run every job in the queue once and then return. Use the *-v* (verbose) flag to watch what is going on. It may be necessary to run the old mail queue a number of times before all of the messages can be delivered.

When the queue is finally emptied, the directory can be removed:

```
rmdir /var/spool/omqueue
```

The Queue Files

All queue file names take this form:

x**f***AA99999*

Where *AA99999* is the ID for this file and *x* is a file type. Valid types include the following:

d	The data file. The message body (excluding the header) is kept in this file.
n	This file is created when an ID is being created. It is a separate file to ensure that no mail can ever be destroyed because of a race condition. The file should exist for no more than a few milliseconds at any given time.
q	The queue control file. This file contains the information necessary to process the job.
t	A temporary file. This is an image of the *qf* file when it is being rebuilt. It should be renamed to a *qf* file very quickly.
x	A transcript file, existing during the life of a session showing everything that happens during that session.

The *qf* file is structured as a series of lines each beginning with a code letter. The codes and lines are as follows:

C	The controlling user. This line defines the name of the controlling user for security purposes. If the message was forwarded, the line is set to the user name of the original message recipient. There can be no more than one of these lines in a *qf* file.
D	The name of the data file. There must only be one of these lines in a *qf* file.
E	An error address. If any such lines exist, they contain the addresses that should receive error messages.
H	A header definition. There may be any number of these lines in a *qf* file. The order is important, as the header lines appear in the final message in the order in which they appear in the *qf* file. These lines use the same syntax as header definitions in the configuration file.
M	A message. This line is printed by the *mailq* command and is generally used to store status information. It can contain any text.
P	The current message priority. This line is used to order the queue. Higher numbers mean lower priorities. The priority changes as the message sits in the queue. The initial priority depends on the message class and the size of the message.

R	A recipient address. This line will normally be completely aliased, but is actually re-aliased when the job is processed. There will be one line for each recipient in a *qf* file.
S	The sender address. There must only be one of these lines.
T	The job creation time. This is the time (in seconds since January 1, 1970) that the job was entered into the queue. This value is used to compute when to timeout the job.

The following example is a queue control file for a message sent from *jd@iris.company.com* to *bossman@company.com* and *buddy@company.com:*

```
P132
T670288862
DdfAA13557
S<jd@iris.company.com> R<bossman@company.com>
R<buddy@company.com>
Hdate: Fri, 29 Mar 91 23:21:00 GMT
HFrom: jd (John Doe)
Hfull-name: John Doe
Hsubject: This is an example message
Hmessage-id: <8209232249.AA13557@IRIS.COMPANY.COM>
Hreceived: by IRIS.COMPANY.COM (5.65+bind 1.5+ida/
900410.SGI)for
bossman@company.com id AA13557; Fri, 29 Mar 91 23:21:00 GMT
HTo: bossman@company.com, buddy
```

The lines in the example provide the following information:

- the message priority
- the submission time (in seconds since January 1, 1970)
- the name of the data file
- the person who sent the message
- the recipients
- the headers for the message

The Alias Database

The alias database is an *ndbm*(3B) database. The text form of the database is maintained in the file */usr/lib/aliases.* The alias are of this form:

name: *name*1 [, *name2*, ...]

For example, the following command delivers mail addressed to *jd* to *johndoe@company.com:*

```
jd:johndoe@company.com
```

Caution: Only the local part of addresses may be aliased. For example, the following command is wrong and will not have the desired effect: `jd@big.university.edu:jd@company.com`. *sendmail* consults the alias database only after deciding that the message (as originally addressed) should be delivered locally, and after it has rewritten the address to contain only the local part.

Alias continuation lines must start with a space or a tab. Blank lines and lines beginning with the number sign (#) are treated as comments.

If you are running NIS, *sendmail* can use the contents of the NIS alias database with the local */usr/lib/aliases* database by adding the following special alias to the */usr/lib/aliases* file:

```
+:+
```

This special alias tells *sendmail* to consult the NIS alias database if the alias cannot be found in the local alias database. (Notice that local alias therefore supersede NIS alias.)

Building the Alias Database

At startup *sendmail* automatically uses the ndbm (3B) routines to process the */usr/lib/aliases* file into the files */usr/lib/aliases.dir* and */usr/lib/aliases.pag*. Using these files to resolve alias is a technique that improves performance.

To rebuild the DBM version of the database without restarting *sendmail,* execute this command

```
newaliases
```

Executing this command is equivalent to giving *sendmail* the *-bi* flag:

```
/usr/lib/sendmail -bi
```

When building the DBM version of the database, *sendmail* checks the left-hand side of each entry to make sure that it is a local address. *sendmail* issues a warning for each entry in */usr/lib/aliases* with a non-local left-hand side. Such entries are not entered into the DBM version of the database.

If the NIS alias database is used with the local *usr/lib/aliases* database, the special "+:+" alias is entered into the DBM version of the database. If *sendmail* cannot find an alias in the DBM version of the database, it looks for the special "+:+" alias. If it finds the special alias, *sendmail* then queries the NIS alias database. This permits the global NIS alias database to change without having to rebuild the local alias database. However, the left-hand sides of the NIS alias are *not* checked by *sendmail* to ensure that they contain only local addresses.

If the configuration or the command line specifies the *D* option, *sendmail* will automatically try to rebuild the alias database when it is out of date.

sendmail rebuilds the alias database if either of the following conditions exists:

- The DBM version of the database is mode 666.
- *sendmail* is running *setuid* to root.

Auto-rebuild can be dangerous on heavily loaded machines with large alias files. If it takes more than five minutes to rebuild the database, there is a chance that several processes will start the rebuild process simultaneously.

Testing the Alias Database

You can test the alias database as described in "Verify Mode" on page 861.

Potential Problems

Problems can occur with the alias database, especially if a *sendmail* process accesses the DBM version before it is completely rebuilt. Two circumstances can cause this problem:

- One process accesses the database while another process is rebuilding it.
- The process rebuilding the database dies (because of being killed or a system crash) before completing the rebuild.

sendmail has two techniques to try to relieve these problems. First, to avoid the problem of a partially rebuilt database, *sendmail* ignores interrupts while rebuilding the database. Second, at the end of the rebuild it adds an alias of the following form (which is not normally legal):

```
@: @
```

Before *sendmail* accesses the database, it ensures that this entry exists. For this action to occur requires the *-a* option in the configuration file. Be sure to specify this unless you run *delivermail* in parallel with *sendmail.*

If the @ : @ entry does not exist, *sendmail* waits for it to appear. After the specified waiting period elapses, *sendmail* will force a rebuild itself. For this action to occur, the configuration file must include the *D* option. If the *D* option is not specified, a warning message is generated and *sendmail* continues.

Another alias problem can arise for systems incorporating the NIS alias database in */usr/lib/aliases* through the use of the "+:+" alias. If the NIS alias server goes down or is otherwise nonresponsive to NIS queries, *sendmail* will not see the alias normally obtained from the NIS server. This situation may result in mail's being returned, marked "User unknown."

List Owners

If an error occurs when mail is sent to a certain address (*x*, for example), *sendmail* looks for an alias of the following form to receive the error:

```
owner-x
```

This scheme is typically useful for a mailing list where a user mailing to the list has no control over the maintenance of the list itself; in this case the list maintainer would be the owner of the list. For example, the following would cause *jd@1company.com* to get the error that occurs when someone sends mail to *unix-hackers* and *sendmail* finds the phony user *nosuchuser* on the list.

```
unix-hackers: jd@company1.com, ed@big.university.edu,
nosuchuser,  jane@company2.com

owner-unix-hackers: jd@company1.com
```

Per-User Forwarding

As an alternative to the alias database, a user can put a file with the name *.forward* in his home directory. If the *.forward* file exists, *sendmail* redirects mail for that user to the list of recipients in the file. The recipients are separated by commas or newlines. For example, if the home directory for user *jane* has a *.forward* file with the following contents, any mail arriving for *jane* is redirected to the specified accounts:

```
zippy@state.edu

bongo@widgets.com
```

The *.forward* file also allows the user to redirect mail to files or programs. A *.forward* file with the following contents will redirect any incoming messages to *jd@company.com,* append a copy of the message to the file */var/tmp/mail.log,* and pipe a copy of the message to *stdin* of the */usr/bin/mymailer* program:

```
jd@company.com

/var/tmp/mail.log

|/usr/bin/mymailer
```

In general, file-type recipients must be writable by everyone. However, if *sendmail* is running as *root* and the file has *setuid* or *setgid* bits set, then the message will be written to the file.

Users can redirect mail to themselves in addition to sending it to other recipients. This is particularly useful if the users want to continue to receive mail in their own mailboxes while passing copies of each incoming message to some alternative destination. For example, say that the home directory for user *john* contains a *.forward* file with the following contents:

```
\john, |/usr/sbin/vacation
```

sendmail will behave as follows:

- It will send each incoming message to john's regular mailbox; the backslash (\) preceding the name indicates that no further aliasing is to occur.
- It will pipe a copy of each message to *stdin* of the */usr/sbin/vacation* program. (The vertical bar [|] is the standard UNIX pipe symbol.)

Special Header Lines

The configuration file defines special interpretations for several header lines. In others cases, *sendmail* defines special interpretations. This section describes these special header lines.

Return-Receipt-To

When this header appears in a message, a return-receipt message is sent to the specified addresses when the message is successfully delivered to a mailer with the *l* flag (for "local delivery") set in the mailer descriptor. The header is added by the program that presents the mail interface to the sender. For example, *mail_bsd*(1) is one such program.

Errors-To

If errors occur at any time during processing, this header directs error messages to the listed addresses rather than to the sender. The header is automatically added by *sendmail* when it expands a mailing list. The address of the list owner is inserted into the header. For more information, see the "List Owners" on page 857 topic contained in "The Alias Database" on page 855.

Apparently-To

If a message arrives with no recipients listed in a "To:," "Cc:," or "Bcc:" line, *sendmail* will add an "Apparently-To" header line for any recipients it knows. The appearance of an "Apparently-To" header instead of a standard recipient line notifies recipients that the list is not complete.

At least one recipient line is required under RFC 822.

sendmail Command-Line Flags

You can include one or more flags on the command line to tailor a *sendmail* session. This section describes some of the more frequently used flags. For a complete description of command-line flags, see "Flags, Options, and Files" on page 887.

Changing the Values of Configuration Options

The *-o* flag overrides an option in the configuration file. The override is for the current session only. In the following example, the *T* (timeout) option becomes two minutes for this session only:

```
/usr/lib/sendmail -oT2m
```

For a complete discussion of configuration options, see "Flags, Options, and Files" on page 887.

Delivery Mode

One configuration option frequently overridden on the command line is the *d* option, which specifies the *sendmail* delivery mode. The delivery mode determines how quickly mail is delivered:

i	deliver interactively (synchronously)
b	deliver in background (asynchronously)
q	queue only (don't deliver)

There are trade-offs. Mode *i* passes the maximum amount of information to the sender, but is rarely necessary.

Mode *q* puts the minimum load on your system, but if you use it, delivery may be delayed for up to the queue interval.

Mode *b* is probably a good compromise. However, in this mode, *sendmail* may initiate a large number of processes if you have a mailer that takes a long time to deliver a message.

Queue Mode

The *-q* flag causes *sendmail* to process the mail queue at regular intervals. The syntax is as follows, where *time* defines the interval between instances of queue processing:

```
-q [time]
```

Time is expressed in number of minutes: 15m sets the interval to 15 minutes. If *time* is omitted, *sendmail* processes the queue once and returns. The *-q* flag is often used in conjunction with daemon mode, described in the next subsection.

See "Timeouts and Intervals" on page 864 for a discussion of time interval specifications and formats.

Daemon Mode

To process incoming mail over sockets, a daemon must be running. The *-bd* flag causes *sendmail* to run in daemon mode. The *-bd* and *-q* flags can be combined in one call, as in the following example:

```
/usr/lib/sendmail -bd -q30m
```

This command causes *sendmail* to run in daemon mode and to fork a subdaemon for queue processing every half hour.

The script for starting *sendmail* that is provided with IRIX includes the following command line:

```
/usr/lib/sendmail -bd -q15m
```

Verify Mode

Using the *-bv* flag directs *sendmail* to validate addresses, aliases, and mailing lists. In this mode *sendmail* performs verification only. It does not try to collect or deliver a message. *sendmail* expands all aliases, suppresses duplicates, and displays the expanded list of names. For each name, *sendmail* indicates if it knows how to deliver a message to that destination.

Test Mode

The *-bt* flag places *sendmail* in test mode so that it describes how the current configuration rewrites addresses. Test mode is extremely useful for debugging modifications to the */usr/lib/sendmail.cf* configuration file. For more information, see the "Test Mode" on page 882 topic contained in "Relevant Issues" on page 881.

Debugging Flags

Several debugging flags are built into *sendmail.* Each flag includes a number and a level. The number identifies the debugging flag. The level, which defaults to 1, dictates how much information prints. A low level causes minimal information to print; a high level causes more comprehensive information to print. By convention, levels greater than 9 are "absurd," so much information prints that it is of limited value. Debugging flags use the following syntax:

-d *debug-list*

A debug list includes the flag number and the flag level, as shown in the following examples.

- Set flag 13 to level 1.

  ```
  -d13
  ```

- Set flag 13 to level 3.

  ```
  -d13.3
  ```

- Set flags 5 though 18 to level 1.

  ```
  -d5-18
  ```

- Set flags 5 through 18 to level 4.

  ```
  -d5-18.4
  ```

Many debugging flags are of little use to the average *sendmail* user. Some are occasionally useful for helping to track down obscure problems. "Flags, Options, and Files" on page 887 includes a complete list of debugging flags.

Using a Different Configuration File

The *-C* flag directs *sendmail* to use an alternate configuration file. For example, the following line directs *sendmail* to use the *test.cf* file instead of the default */usr/lib/sendmail.cf* file:

```
/usr/lib/sendmail -Ctest.cf
```

If the *-C* flag appears without a file name, *sendmail* uses the file *sendmail.cf* in the current directory. Thus, the *-C* flag directs *sendmail* to ignore any */usr/lib/sendmail.fc* (“frozen”) file that may be present.

Tuning

A number of configuration parameters are available for fine-tuning *sendmail* to the requirements of a specific site. Options in the configuration file set these parameters. For example, the string "*T3d*" sets the *T* (timeout) option to "3d" (three days).

Most options have default values that are appropriate for many sites. However, sites having very high mail loads may need to tune these parameters to fit the mail load. In particular, sites with a large volume of small messages that are delivered to multiple recipients may need to adjust the parameters for queue priorities.

The rest of this section describes the configuration parameters for the following tuning areas:

- timeouts and intervals
- forking during queue runs
- queue priorities
- load limiting
- delivery mode
- log level

Timeouts and Intervals

Time intervals use the following abbreviations:

s	seconds
m	minutes
h	hours
d	days
w	weeks

For example, *10m* represents 10 minutes, and *2h30m* represents two hours and 30 minutes.

Queue Interval

The argument to the *-q* flag specifies how often to process the mail queue. When the *sendmail* daemon is started with the */etc/init.d/mail* script, the queue interval is set to 15 minutes.

If *sendmail* runs in delivery mode *b*, messages are written to the queue only when they cannot be delivered (for example, when a recipient host is down). Therefore, the need to process the queue is limited and the queue interval value may be set quite high. The value is relevant only when a host that was down comes back up.

If *sendmail* runs in delivery mode *q*, the queue interval should be set to a low value, as it defines the longest time that a message sits in the local queue before being processed.

Read Timeouts

sendmail can time out when reading the standard input or when reading from a remote SMTP server. Technically, a timeout is not acceptable within the published protocols. However, setting the read timeout option to a high value (such as an hour) reduces the chance that a large number of idle daemons will pile up on a system. The read timeout option is *r*.

Message Timeouts

sendmail causes a queued message to time out after a specified time period. This feature ensures that the sender knows the message cannot be delivered. This default message timeout value is one week (seven days). The value is set with the *T* option.

The queue records the time of submission, rather than the time remaining until timeout. This approach enables *sendmail* to flush messages that have been hanging for a short period by running the queue with a short message timeout. The following example illustrates how to process the queue and flush any message that is one day old:

```
/usr/lib/sendmail -oT1d -q
```

Forking During Queue Runs

Setting the *Y* option causes *sendmail* to fork before each individual message when processing the queue. This technique prevents *sendmail* from consuming large amounts of memory, and may be useful in memory-poor environments. However, if the *Y* option is not set, *sendmail* keeps track of hosts that are down during a queue run, which can improve performance dramatically.

Queue Priorities

sendmail assigns a priority to every message when it is first instantiated. *sendmail* uses the priority and the message creation time (in seconds since January 1, 1970) to order the queue. The message with the lowest priority number is processed first. The algorithm that derives a message's priority uses the following information:

message size (in bytes)
: Small messages receive lower priorities than large messages, increasing the efficiency of the queue.

message class
: If the user includes a "Precedence:" field and value in a message, *sendmail* uses the value to select the message class from the configuration file. (Typical values might be "first-class" or "special-delivery.")

class work factor
: This factor is set in the configuration file with the *z* option; its default value is 1800.

number of recipients
: The number of recipients affects the load a message presents to the system. A message with a single recipient has a lower priority than one with a long recipient list.

recipient work factor
: This factor is set in the configuration file with the *y* option; its default value is 1000.

The priority algorithm is as follows:

```
priority=message_size-(message_class * z)+(num_recipients * y)
```

After assigning message priorities, *sendmail* orders the queue by using the following formula:

```
ordering = priority + creation_time
```

A message's priority can change each time an attempt is made to deliver it. The "work time factor" (set with the *Z* option) is a value that increments the priority, on the assumption that a message that has failed many times will tend to fail in the future.

Load Limiting

sendmail can queue (and not attempt to deliver) mail if the system load average exceeds a specified maximum. The *x* option defines this maximum limit. When the load average exceeds the maximum, *sendmail* tests each message's priority by using the following algorithm, where *q* is the value associated with the *q* option, and *x* is the value associated with the *x* option:

q / (load_average - *x* + 1)

In IRIX *sendmail,* the algorithm is modified slightly. The number of child processes forked by the *sendmail* daemon is added to the load average. Thus, a host with a load average of 1 but with six forked *sendmail* processes has an effective load average of 7.

After the final load average is calculated, *sendmail* compares it to the message priority for each message. If the priority is greater, *sendmail* sets the delivery mode to *q* (queue only).

The default value for the *q* option is 10000; each point of load average is worth 10000 priority points. The *X* option defines the load average at which *sendmail* refuses to accept network connections. Locally generated mail (including incoming UUCP mail) is still accepted.

Log Level

sendmail provides a comprehensive error- and event-logging capability. The *L* (log level) option in the configuration file determines the level of detail written to the log. The default value for the log level is 1; valid levels are as follows:

0	no logging
1	major problems only
2	message collections and failed deliveries
3	successful deliveries
4	messages being deferred (because a host is down, for example)
5	normal message queue-ups
6	unusual but benign incidents, such as trying to process a locked queue file
9	log internal queue ID to external message ID mappings; useful for tracing a message as it travels among several hosts
12	several messages of interest when debugging
16	verbose information regarding the queue

The Configuration File

This section begins with an overview of the configuration file, describes its semantics in detail, and includes hints on writing a customized version. Admittedly, the file's syntax is not easy to read or write. A more important design criterion is that the syntax be easy to parse, because it must be parsed each time *sendmail* starts up.

The Syntax

The configuration file is a series of lines, each of which begins with a character that defines the semantics for the rest of the line. A line beginning with a space or a tab is a continuation line (although the semantics are not well defined in many places). A blank line or a line beginning with the number symbol (#) is a comment.

Rewriting Rules—the S and R Commands

The rewriting rules are the core of address parsing. These rules form an ordered production system. During parsing, *sendmail* scans the set of rewriting rules, looking for a match on the left-hand side (ls) of the rule. When a rule matches, the address is replaced by the right-hand side (rhs) of the rule.

The rewriting rules are grouped into sets of one or more rules. Each set has an integer identifier called a *rule set number*. Each rule set acts like a subroutine: When the set is called, each rule is scanned in sequence until all rules have been scanned or until the rule set returns to the caller.

Some rule sets are used internally and have specific semantics. Others do not have specifically assigned semantics and can be referenced by mailer definitions or by other rule sets.

Each rule set begins with an S command, with the syntax

S*n*

where *n* is the rule set identifier, an integer between 0 and 99. If the same rule set identifier is used more than once in a configuration file, the last definition is the one used.

Each line within the rule set begins with an R command and defines a rewrite rule. R commands have the syntax

R*lhs rhs* [*comments*]

where the following conditions exist:

- The fields are separated by at least one tab character; embedded spaces with a field are acceptable.
- Any input matching the *lhs* pattern is rewritten according to the *rhs* pattern.
- Comments, if any, are ignored.

Define Macro–the D Command

The D command names a macro and defines its content. A macro name is a single ASCII character. Because *sendmail* defines several internal macros, care must be taken to avoid using their identifiers. If an identifier is re-used, it effectively redefines the macro. To avoid conflicts, the convention is that the name for any user-defined macro is an uppercase letter. Lowercase letters and special characters are reserved for system use. (A list of *sendmail*'s internally defined macros appears under the topic "Special Macros and Conditionals" on page 874 in "The Semantics" on page 873.

The Define Macro command takes one of the following forms:

D*x Value*

D*x*|*shell_command* [*arguments*]

where *x* is the name of the macro. The first form defines the macro to contain *Value.* The second form defines the macro as the first line seen on *stdout* of the specified shell command. The vertical bar must immediately follow the macro identifier; no white space is permitted. The remainder of the line is interpreted as arguments to the shell command.

Macros can be interpolated in most places by means of the escape sequence $x.

Caution: *sendmail* does not honor comments on macro definition lines. For example, the following line defines the *D* macro as "foo.com # my domain name": `DDfoo.com  # my domain name`

Define Classes–the C and F Commands

The C command defines classes of words to match on the left-hand side of rewriting rules, where a "word" is a sequence of characters. A class name is a single uppercase letter. A class might define a site's local names and be used to eliminate attempts to send to oneself. Classes can be defined either directly in the configuration file or by being read in from a pipe or other file. The sequence cannot contain characters used as "operators" in addresses. (Operators are defined with the $o macro.)

The C command takes one of the following forms:

`C`*X word1 [word2 . . .]*

`C`*X $x [$y ...]*

where *X* is the class name. The first form defines the class to match any of the named words. The second form reads the elements of the class from the expansion of the listed macros. The macros to be expanded must be leftmost in each token. Only one macro can be expanded per token. Any remainder in a token following a macro expansion will be added as a separate token.

The F command defines a file from which to read the elements of a class, and takes either of the following forms:

`F`*X file [format]*

`F`*X*`|`*shell_command [arguments]*

The first form reads the elements of the class from the named file. If an optional format argument is present, it is passed as the control string to an *sscanf*(3S) call and applied to each input line read from the named file, thus:

```
sscanf(const char *line, const char *format, char *new-element);
```

There must be no white space to the left of the file name.

The second form reads the elements of the class from *stdout* of the specified shell command. The entire contents of the line are taken to be a shell command followed by its arguments.

It is permissible to split class definitions across multiple lines. For example, the two forms:

```
CHmonet picasso
```

and

```
CHmonet
CHpicasso
```

are equivalent.

It is also permissible to define a class using both C and F commands. For example the following commands define class H containing *monet, picasso*, the expansion of the $w macro, and all strings from the file */var/tmp/foofile* appearing before the first number symbol (#) on each line:

```
CHmonet picasso $w
FH/var/tmp/foofile %[^#]
```

Caution: *sendmail* does not honor comments are not respected on class definition lines. For example, the following command defines the D class as containing "foo.com," "bar.com," "#," "my," "local," and "domains": `CD foo.com bar.com # my local domains`

Define Mailer–the M Command

The M command defines programs and interfaces to mailers. The format is as follows:

`M`*name*`, {`*field=value*`}*`

where *name* is the name of the mailer (used internally only) and the field=value pairs define attributes of the mailer. The asterisk (*) indicates that the preceding bracketed structure may be repeated 0 or more times. That is, there may be multiple *field=value* pairs. The following list defines valid field names:

Path	The pathname of the mailer
Flags	Special flags for this mailer; see "Flags, Options, and Files" on page 887 for a list of special flags.
Sender	A rewriting set for sender addresses.
Recipient	A rewriting set for recipient addresses.

Argv	An argument vector to pass to this mailer.
Eol	The end-of-line string for this mailer.
Maxsize	The maximum message length to this mailer.

Only the first character of the field name is checked.

Define Header–the H Command

The H command defines the format of header lines that *sendmail* inserts into a message. The command format is as follows:

`H[?`*mflags*`?]`*hname: htemplate*

Continuation lines for an H command line appear in the outgoing message. *sendmail* macro-expands the *htemplate* before inserting it into the message. If *mflags* (which must be surrounded by question marks) appear in the command line, at least one of the specified flags must also appear in the mailer definition or the header will not appear in the message. However, if a header appears in the input message, the header also appears in the output, regardless of these flags.

Some headers have special semantics, which are described in "The Semantics" on page 873.

Set Option–the O Command

Several *sendmail* options can be set by a command line in the configuration file. Each option is identified by a single character.

The format of the O command line is as follows:

`O`*o value*

The command sets option *o* to *value*. Depending on the option, *value* is a string, an integer, a Boolean (with legal values "t ," "T," "f," or "F" and a default of TRUE), or a time interval.

The K option is a new option available in IRIX *sendmail.* This option defines (K)eyed databases that are accessible by means of the IDA lookup operators. See "Configuration Options" on page 888 for a complete description of the K option, as well as a list of all configuration options.

Define Trusted Users–the T Command

Trusted users are permitted to override the sender address by using the *-f* flag. Typically, trusted users are limited to *root*, *uucp*, and *network.* On some systems it may be convenient to include other users. For example, there may be a separate *uucp* login for each host. Note, though, that the concept of trusted users in *sendmail* bears no relation to Trusted IRIX/B, or to the concept of trusted (secure) operating systems in general.

The format of the T command is as follows:

T*user1 user2...*

A configuration file can contain multiple T lines.

Define Precedence–the P Command

The P command defines values for the "Precedence:" field. The format of the command line is as follows:

P*name*=*num*

When *sendmail* matches the value in a message's "Precedence: " field with the *name* in a P command line, *sendmail* sets the message's class to *num.* A higher number indicates a higher precedence. Numbers less than zero indicate that error messages will not be returned to the sender. The default precedence is zero. For example, a list of precedences might be:

```
Pfirst-class=0
Pspecial-delivery=100
Pjunk=-100
```

The Semantics

This section describes the semantics of the configuration file, including special macros, conditionals, special classes, and the "error" mailer.

Special Macros and Conditionals

Macros are interpolated by means of the construct $*x*, where *x* is the name of the macro to be interpolated. Special macros are named with lowercase letters; they either have special semantics or pass information into or out of *sendmail*. Some special characters are also reserved, and are used to provide conditionals and other functions.

The following syntax specifies a conditional:

`$?`*x* *text1* `$|` *text2* `$.`

This example interpolates text1 if the macro *$x* is set, and text2 otherwise.

The "else" (`$|`) clause can be omitted.

The following macros *must b*e defined if information is to be transmitted into *sendmail*:

e	SMTP entry message, which prints out when STMP starts up
j	"official" domain name for this site; must be the first word of the *$e* macro
l	format of the UNIX "from" line; usually a constant
n	name of the daemon (for error messages); usually a constant
o	set of token operators in addresses
q	default format of sender address

The $o macro consists of a list of characters that are treated as tokens themselves and serve to separate tokens during parsing. For example, if @ were in the *$o* macro, then the input string "a@b" would scan as three tokens: a, @, and b.

Here are several examples of these macro definitions:

```
De$j Sendmail $v/$Z ready at $b

Dj$w

DlFrom $g $d

DnMAILER-DAEMON
```

```
Do.:%@!^=/[]
Dq$g$?x ($x)$.
```

An acceptable alternative format for the *$q* macro is "$?x$x $.<$g>" . This syntax corresponds to the following two formats:

```
jd@company.com (John Doe)
John Doe <jd@company.com>
```

Some macros are defined by *sendmail* for interpolation into *argv*'s for mailers or for other contexts. These macros are:

a	origination date in RFC 822 format
b	current date in RFC 822 format
c	hop count
d	date in UNIX (ctime) format
f	sender (“From”) address
g	sender address relative to the recipient
h	recipient host
i	queue ID
p	pid for *sendmail*
r	protocol used
s	sender’s host name
t	a numeric representation of the current time
u	recipient user
v	version number of *sendmail*
w	host name of this site
x	full name of the sender
z	home directory of the recipient

Macros *$a*, *$b*, and *$d* specify three dates that can be used. The *$a* and *$b* macros are in RFC 822 format. *$a* is the time extracted from the “Date:” line of the message; *$b* is the current date and time (used for postmarks). If no

"Date:" line appears in the incoming message, *$a* is also set to the current time. The *$d* macro is equivalent to the *$a* macro in UNIX (*ctime*) format.

The *$c* macro is set to the "hop count," the number of times this message has been processed. It can be determined by the *-h* flag on the command line or by counting the time stamps in the message.

The *$f* macro is the ID of the sender as originally determined; when a message is sent to a specific host, the *$g* macro is set to the address of the sender *relative to the recipient.* For example, if *jd* sends to *buddy@USomewhere.edu* from the machine *company.com*, the *$f* macro will be *jd* and the *$g* macro will be *jd@company.com.*

When a message is sent, the *$h*, *$u*, and *$z* macros are set to the host, user, and home directory (if local) of the recipient. The first two are set from the *$@* and *$:* part of the rewriting rules, respectively. The *$i* macro is set to the queue ID on this host; when included in the time-stamp line, it can be extremely useful for tracking messages.

The *$p* and *$t* macros are used to create unique strings (for example, for the "Message-Id:" field). The *$r* and *$s* macros are set to the protocol used to communicate with *sendmail* and the name of the sending host; these macros are not supported in the current version.

The *$v* macro is set to be the version number of *sendmail*, which normally appears in time stamps and is extremely useful for debugging. The *$w* macro is set to the name of this host, if it can be determined.

The *$x* macro is set to the full name of the sender, derived from one of these sources (in this order):

- a flag to *sendmail*
- the value of the "Full-name:" line in the header if it exists
- the comment field of a "From:" line
- for messages originating locally, the value from the */etc/passwd* file

Special Classes

The *w* class is set to be the set of all names this host is known by, and can be used to match local host names.

The Left-Hand Side

The left-hand side of rewriting rules contains a pattern. Words are simply matched directly. A dollar sign introduces the following meta-syntax:

$*	Match zero or more tokens.
$+	Match one or more tokens.
$-	Match exactly one token.
$=x	Match any token in class x.
$~x	Match any token not in class x.

If a match occurs, it is assigned to the symbol $*n* for replacement on the right-hand side, where *n* is the index in the lhs. For example, if the lhs:

```
$-@$+
```

is applied to the input:

```
jd@company.com
```

the rule will match, and the values passed to the rhs will be:

$1	jd
$2	company.com

The Right-Hand Side

When the left-hand side of a rewriting rule matches, the input is deleted and replaced by the right-hand side. Right-hand-side tokens are inserted exactly as they appear unless they begin with a dollar sign. The meta-syntax is:

$n	Substitute indefinite token n from lhs; substitute the corresponding value from a $+, $-, $*, $=, or $~ match on the lhs; can be used anywhere.
$[*hostname*$]	Canonicalize *hostname*; a host name enclosed between $[and $] is looked up by the *gethostbyname*(3N) routines and replaced by the canonical name. For example, $[frodo$] might become *frodo.fantasy.com* and $[[192.48.153.1]$] would become *sgi.com*. In the event that *gethostbyname*(3N) encounters an error, *hostname* will be returned unchanged.

$[*name*$:*default*$]

This is an extended version of the preceding construct, and provides a way to determine whether the canonicalization was successful. Using this syntax, the *default* is returned if the canonicalization step fails. For example, $[frodo$:FAIL$] would become FAIL if *gethostbyname*(3N) fails to canonicalize *frodo*.

$(*x key*$@*arg*$:*default*$)

Look up key in DBM or NIS database *x*. This is the IDA database lookup syntax; *x* corresponds to a database previously declared using the *K* option (described in "Configuration Options" on page 888) and *key* is the string that should be searched for in the database. The *arg* and *default* arguments are optional. The *default* is returned if the *key* was not found in the database. If neither the *default* nor a matching *key* is found, the whole expression expands to the value of *key*. However, if a result is found, it is used as the format string of a *sprintf*(3S) expression, with the *arg* as extra argument. Thus, database values with "%s" strings embedded in them can be useful when rewriting expressions. These values could typically be used with the *pathalias* program to expand routes without leaving *sendmail*.

${*x query*$:*default*$}

Look up query in DNS database *x*. This is the DNS database lookup syntax. The various values of *x* are internally defined and correspond to various DNS databases. The *default* argument is optional and will be returned in the event that *query* cannot be found in the specified database.

The values for *x* are:

@ Return MX record for *query*.

. As above, but use domain search rules.

: Return MR record for *query*.

? Return MB record for *query*.

c Expand *query* to its canonical name following MX records.

C As above, but use domain search rules.

h Like *${c* but for A records.

H As above, but use domain search rules.

When domain search rules are requested, *sendmail* sets the RES_DNSRCH flag when calling the resolver. See the *resolver*(3N) man page for further information.

$>*n* Call rule set *n*; causes the remainder of the line to be substituted as usual and then passed as the argument to rule set *n*. The final value of rule set *n* becomes the substitution for this construct.

Multiple calls can be embedded on the rhs. For example, >32>33$1 would make the substitution for $1 and pass the result to rule set 33. The result from rule set 33 would then be passed as the input to rule set 32. Finally, the entire construct would be replaced with the result from rule set 32.

Only embedded rule set calls in the form outlined above are supported. Rule sets calls cannot be arbitrarily placed within the rhs.

$#*mailer*$@*host*$:*user*
Resolve to mailer; the *$#* syntax should be used only in rule set 0. The syntax causes evaluation of the rule set to terminate immediately and signals sendmail that the address has completely resolved. This process specifies the {mailer, host, user} triple necessary to direct the mailer. If the mailer is local, the host part may be omitted. The mailer and host must be a single word, but the user may be a multi-part value.

An entire rhs may also be prefixed by a *$@* or a *$:* to control evaluation.

The *$@* prefix causes the rule set to return with the remainder of the rhs as the value. The *$:* prefix causes the rule to terminate immediately, but the rule set to continue; this can be used to avoid continued application of a rule. The prefix is stripped before continuing.

The *$@* and *$:* prefixes may precede a *$>* specification. For example,

```
R$+   $:$>7$1
```

matches anything, passes that to rule set 7, and continues; the *$:* is necessary to avoid an infinite loop.

Substitution occurs in the order described: Parameters from the lhs are substituted, host names are canonicalized, "subroutines" are called, and finally *$#*, *$@*, and *$:* are processed.

Semantics of Rewriting Rule Sets

There are five rewriting rule sets that have specific semantics. Figure D-2 shows the relationship among these rule sets.

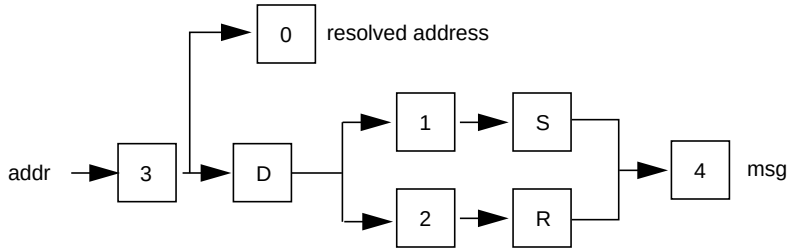


Figure D-2 Semantics of Rewriting Rule Sets

Rule set 3 should turn the address into canonical form. This form should have the following basic syntax:

```
local-part@host-domain-spec
```

If no "at" (@) sign is specified, then the host-domain-spec can be appended from the sender address (if the *C* flag is set in the mailer definition corresponding to the *sending* mailer). *sendmail* applies rule set 3 before doing anything with any address.

Next, rule set 0 is applied to an address that actually specifies recipients. The address must resolve to a {*mailer, host, user*} triple. The *mailer* must be defined in the mailer definitions from the configuration file. The *host* is

defined into the *$h* macro for use in the *argv* expansion of the specified mailer.

Rule sets 1 and 2 are applied to all sender and recipient addresses, respectively. They are applied before any specification in the mailer definition. They must never resolve.

Rule set 4 is applied to all addresses in the message. It is typically used to translate from internal to external form.

The "error" Mailer

The mailer with the special name "error" can be used to generate a user error. The (optional) host field is a numeric exit status to be returned, and the user field is a message to be printed. For example, the following entry on the rhs of a rule causes the specified error to be generated if the lhs match:

```
$#error$:Host unknown in this domain
```

This mailer is functional only in rule set 0.

Relevant Issues

This section discusses testing and debugging the rewrite rules and building mailer definitions.

Testing and Debugging the Rewrite Rules

As part of building or modifying a configuration file, you should test the new file. *sendmail* provides a number of built-in tools to assist in this task. The subsections that follow discuss tools and techniques for debugging the rewrite rules.

Using Alternative Configuration Files

Using the *-C* command-line flag causes *sendmail* to read an alternate configuration file. This feature is helpful during debugging because it permits modifications and testing on a separate copy of the configuration file from the one currently in use. This precaution eliminates the chance that a buggy configuration file will be used by an instance of *sendmail* that is trying

to deliver real mail. This flag also provides a convenient way to test any number of configuration files without fussy and potentially confusing renames. See "Using a Different Configuration File" on page 863 and "Command-Line Flags" on page 887 for more information.

Test Mode

Invoking *sendmail* with the *-bt* flag causes it to run in "test mode." For example, the following command invokes *sendmail* in test mode and causes it to read configuration file *test.cf:*

```
/usr/lib/sendmail -bt -Ctest.cf
```

In this mode, *sendmail* processes lines of the form:

rwsets address

where *rwsets* is the list of rewriting sets to use and *address* is an address to which to apply the sets. In test mode, *sendmail* shows the steps it takes as it proceeds, finally showing the final address.

A comma-separated list of rule sets causes sequential application of rules to an input. For example, the following command first applies rule set 3 to the value *monet@giverny*. Rule set 1 is applied to the output of rule set 3, followed similarly by rule sets 21 and 4:

```
3,1,21,4 monet@giverny
```

Note: Some versions of *sendmail*, including those provided with all versions of IRIX prior to IRIX 4.0, automatically apply rule set 3 to input before applying the requested rule set sequence. Versions of *sendmail* in IRIX 4.0 and later do *not* apply rule set 3; rule set 3 must be specifically requested.

The input and output of each rule set is displayed. For example, input of:

```
3,0 foo@bar
```

might result in output that looks like this:

```
rewrite: ruleset 3 input: foo @ bar
rewrite: ruleset 3 returns: foo < @ bar >
rewrite: ruleset 0 input: foo < @ bar >
rewrite: ruleset 30 input: foo < @ bar . com >
rewrite: ruleset 30 returns: foo < @ bar . com >
```

```
rewrite: ruleset 0 returns:
$# forgn $@ bar . com $: foo < @ bar . com >
```

This output indicates that, given the address foo@bar, rule set 0 will select the forgn mailer and direct it to connect to host bar.com, which will be told to send the mail on to foo@bar.com. Furthermore, rule set 0 "called" rule set 30 at one point while processing the address.

The -d21 Debugging Flag

The *-d21* debugging flag causes *sendmail* to display detailed information about the rewrite process. This flag is most useful when used with the test mode described in the preceding subsection. The most useful setting of this flag is *-d21.12*, which shows all rewrite steps. Higher levels of the *-d21* flag are rarely needed and create enormous amounts of output.

The Debugging Rewrite Rule

The standard */usr/lib/sendmail.cf* file supplied with IRIX includes a special "debugging" rewrite rule. This rule is defined as follows:

```
##########################################

# insert this handy debugging line wherever you have # problems

#R$*             $:$>99$1
```

Note that rule set 99 is an empty rule set that does nothing. Placing one or more (uncommented) copies of this rule anywhere within a rule set forces *sendmail* to display an intermediate rewrite result without using the *-d21* flag. The following test mode output illustrates the use of the debugging rewrite rule:

```
rewrite: ruleset 3 input: foo @ bar
rewrite: ruleset 3 returns: foo < @ bar >
rewrite: ruleset 0 input: foo < @ bar >
rewrite: ruleset 99 input: foo < @ bar . com >
rewrite: ruleset 99 returns: foo < @ bar . com >
rewrite: ruleset 99 input: foo < @ barcom >
rewrite: ruleset 99 returns: foo < @ barcom >
rewrite: ruleset 0 returns:
$# ether $@ barcom $: foo < @ barcom >
```

Note that somewhere between the first and second appearance of the debugging rewrite rule in rule set 0, the host name was mangled from

```
bar.com
```

to

```
barcom
```

Building Mailer Definitions

To add an outgoing mailer to a mail system, you must define the characteristics of the mailer. Each mailer must have an internal name. This name can be arbitrary, except that the names “local” and “prog” must be defined.

The pathname of the mailer must be given in the *P* field. If this mailer should be accessed by means of an IPC connection (socket), use the string "[IPC]" instead.

The *F* field defines the mailer flags. Specify an *f* or *r* flag to pass the name of the sender as an *-f* or *-r* flag respectively. These flags are passed only if they were passed to *sendmail*, so that mailers that give errors under some circumstances can be placated. If the mailer is not picky, just specify “-f $g” in the *argv* template. If the mailer must be called as *root*, use the *S* flag; this flag will not reset the userid before calling the mailer. (*sendmail* must be running *setuid* to root for this technique to work.)

If this mailer is local (that is, it will perform final delivery rather than another network hop), use the *l* flag. Quote characters (backslashes and quotation marks) can be stripped from addresses if the *s* flag is specified; if it is not, they are passed through. If the mailer is capable of sending to more than one user on the same host in a single transaction, use the *m* flag. If this flag is on, the *argv* template containing $u will be repeated for each unique user on a given host. The *e* flag will mark the mailer as being expensive, causing *sendmail* to defer connection until a queue run. (For this technique to be effective, you must use the *c* configuration option.)

An unusual case is the *C* flag: It applies to the mailer the message is received from, rather than the mailer being sent to. If this flag is set, the domain specification of the sender (that is, the *@host.domain* part) is saved and is appended to any addresses in the message that do not already contain a domain specification. For example, a message of the form:

```
From: jd@company.com
```

```
To: buddy@USomewhere.edu, jane
```

will be modified to:

```
From: jd@company.com
To: buddy@USomewhere.edu, jane@company.com
```

if the *C* flag is defined in the mailer corresponding to *jd@company.com.*

Other flags are described in "Mailer Flags" on page 892.

The *S* and *R* fields in the mailer description are per-mailer rewriting sets to be applied to sender and recipient addresses, respectively. These sets are applied after the sending domain is appended and the general rewriting sets (sets 1 and 2) are applied, but before the output rewrite (rule set 4) is applied. A typical usage is to append the current domain to addresses that do not already have a domain.

For example, a header of the form:

```
From: jd
```

might be changed to:

```
From: jd@company.com
```

or

```
From: company!jd
```

depending on the domain it is being shipped into.

These sets can also be used to do special-purpose output rewriting with rule set 4.

The *E* field defines the string to use as an end-of-line indication. A string containing only a newline is the default. The usual backslash escapes (\r, \n, \f, \b) can be used.

Finally, an *argv* template is given as the *E* field. It can have embedded spaces. If there is no *argv* with a *$u* macro in it, *sendmail* will speak SMTP to the mailer. If the pathname for this mailer is "[IPC]," the *argv* should be

```
IPC $h [ port ]
```

where *port* is the port number to connect to. This number is optional.

For example, the following specification specifies a mailer to do local delivery and a mailer for Ethernet delivery:

```
Mlocal, P=/bin/mail, F=EDFMlsmhu, S=10, R=20, A=mail -s -d $u

Mether, P=[IPC], F=mDFMhuXC, S=11, R=21, M=1000000, E=\r\n, \
    A=IPC $h
```

The first mailer is called "local" and is located in the file */bin/mail*. It has the following characteristics:

- It escapes lines beginning with "From" in the message with a ">" sign.
- It expects "Date:", "From:", and "Message-Id:" header lines.
- It does local delivery.
- It strips quotation marks from addresses.
- It sends to multiple users at once.
- It expects uppercase to be preserved in both host and user names.

Rule set 10 is to be applied to sender addresses in the message and rule set 20 is to be applied to recipient addresses. The *argv* to send to a message will be the word *mail*, the word *-s* , the word *-d* and words containing the name of the receiving user.

The second mailer is called "ether" and is connected with an IPC connection. It has the following characteristics:

- It handles multiple users at the same time.
- It expects "Date:", "From:", and "Message-Id" header lines.
- It expects uppercase to be preserved in both host and user names.
- It uses the hidden-dot algorithm of RFC 821.
- It rewrites addresses so that any domain from the sender address is appended to any receiver name without a domain.

Sender addresses are processed by rule set 11; recipient addresses, by rule set 21. There is a 1,000,000-byte limit on messages passed through this mailer.

The EOL string for this mailer is "\r\n" and the argument passed to the mailer is the name of the recipient host.

Flags, Options, and Files

This section contains information on the following topics:

- command-line flags
- configuration options
- mailer flags
- summary of support files
- debugging flags

Command-Line Flags

Flags must precede addresses. The flags are:

-bx	Set operation mode to x. Operation modes are:
a	Run in ARPANET mode.
d	Run as a daemon.
i	Initialize the alias database.
m	Deliver mail in the usual way (default).
p	Print the mail queue.
s	Speak SMTP on input side.
t	Run in test mode.
v	Just verify addresses, don't collect or deliver.
z	Freeze the configuration file.
	The special processing for the ARPANET includes reading the "From:" and "Sender:" lines from the header to find the sender, printing ARPANET-style messages (preceded by three-digit reply codes for compatibility with the FTP protocol) and ending lines of error messages with <CRLF>.

`-C`*file* Use a different configuration file. *sendmail* runs as the invoking user (rather than root) when this flag is specified.

`-d`*flag*`[-`*flag*`][.`*level*`]`
Set debugging flag (or range of flags) to the specified level. (The default is 1.) See "Debugging Flags" on page 896.

`-F`*name* Set the full name of the sender to *name.*

`-f` *name* Set the name of the "From" person (the sender of the mail). This flag is ignored unless the user appears in the list of "trusted" users, or the person the user is trying to become is himself.

`-h` *cnt* Set the "hop count" to cnt. The hop count is incremented every time the mail is processed. When it reaches a limit, the mail is returned with an error message, the victim of an aliasing loop.

`-n` Don't do aliasing.

`-o`*x value* Set configuration option *x* to the specified *value.* These options are described in section "Configuration Options" on page 888.

`-q`[*time*] Process the queued mail. If the time is given, *sendmail* will run through the queue at the specified interval to deliver queued mail; otherwise, it only runs once. See "Queue Mode" on page 861.

`-r` *name* An alternative and obsolete form of *-f.*

`-t` Read the header for "To:", "Cc:", and "Bcc:" lines, and send the message to everyone listed in those lists. The "Bcc:" line is deleted before sending. Any addresses in the argument vector are deleted from the send list.

`-v` Go into verbose mode: Alias expansions are announced, and so on.

Configuration Options

You can set the following options by using the *-o* flag on the command line or the *O* line in the configuration file. Many of these options cannot be specified unless the invoking user is trusted.

A*file*	Use the named file as the alias file. If no file is specified, use alias in the current directory.
a*N*	If set, wait up to *N* minutes for an "@:@" entry to exist in the alias database before starting up. If the entry does not appear in *N* minutes, rebuild the database (if the *D* option is also set) or issue a warning.
B*c*	Set the blank substitution character to *c*. Unquoted spaces in addresses are replaced by this character.
c	If an outgoing mailer is marked as being expensive, don't connect immediately. This option requires queueing.
d*x*	Deliver in mode *x*. Legal modes are:
i	Deliver interactively (synchronously).
b	Deliver in background (asynchronously).
q	Just queue the message (deliver during queue run).
D	Rebuild the alias database if necessary and possible. If this option is not set, *sendmail* will never rebuild the alias database unless you explicitly request it to do so (by using *sendmail -bi*).
e*x*	Handle errors by using mode *x*. The values for *x* are:
	p Print error messages (default).
	q Print no messages; just give exit status.
	m Mail back errors.
	w Write back errors (mail the errors if user not logged in).
	e Mail back errors and always give zero exit status.
F*mode*	Set the UNIX file mode to use when creating queue files and "frozen configuration" files.
f	Save UNIX-style "From" lines at the front of headers. Normally these lines are assumed to be redundant and are discarded.
g*n*	Set the default gid for running mailers to n.
H*file*	Specify the help file for SMTP.

`I`	Insist that the BIND name server be running to resolve host names and MX records. Treat ECONNREFUSED errors from the resolver as temporary failures. In general, you should set this option only if you are running the name server. Set this option if the */etc/hosts* file does not include all known hosts or if you are using the MX (mail forwarding) feature of the BIND name server. The name server is still consulted even if this option is not set, but *sendmail* resorts to reading */etc/hosts* if the name server is not available.
`i`	Do not interpret dots on a line by themselves as a message terminator.
`K`*x[%]file*	Define the internal keyed database *x* and associate it with the external database *file; x* is a single character database name. If the percent sign (%) is present, *file* is the name of an NIS database. If the percent signr is absent, *file* is the name of a *ndbm*(3B) database. After a keyed database is defined, you can use it to look up arbitrary strings in the right-hand side of rewriting rules, by using the IDA lookup operator *$($)*. The */usr/lib/aliases* file is automatically available if you are using the "@" database and should not be declared with the *K* option.
`L`*n*	Set the log level to n.
`M`*x value*	Set the macro *x* to *value.* This option can be used only from the command line.
`m`	Send to "me" (the sender) even if the sender is in an alias expansion.
`N`*netname*	Set the name of the home (local) network. If the name of a connecting host (determined by a call to *gethostbyaddr*(3N)) is unqualified (contains no dots), a single dot and *netname* will be appended to *sendmail*'s idea of the name of the connecting host. Later, the argument of the SMTP "HELO" command from the connecting host will be checked against the name of the connecting host as determined above. If they do not

	match, "Received:" lines are augmented by the connecting host name that *sendmail* has generated so that messages can be traced accurately.
`n`	Validate the right-hand side when building the alias database.
`o`	Assume that the headers may be in an old format, in which spaces delimit names. This option actually turns on an adaptive algorithm: If any recipient address contains a comma, parenthesis, or angle bracket, it will be assumed that commas already exist. If this flag is not on, only commas delimit names. Headers are always put out with commas between the names.
`P`*addr*	Add "postmaster" address *addr* to the "Cc:" list of all error messages.
`Q`*dir*	Use dir as the queue directory.
`q`*factor*	Use factor as the multiplier in the function to decide when to queue messages rather than attempting to send them. This value is divided by the difference between the current load average and the load average limit (*x* option) to determine the maximum message priority that will be sent. This value defaults to 10000.
`r`*time*	Cause a timeout on reads after *time* interval.
`S`*file*	Log statistics in the named *file.*
`s`	Be super-safe when running; that is, always instantiate the queue file, even if attempting immediate delivery. *sendmail* always instantiates the queue file before returning control to the client under any circumstances.
`T`*time*	Set the queue timeout to *time.* After this interval, messages that have not been successfully sent are returned to the sender.
`u`*n*	Set the default userid for mailers to n. Mailers without the S flag in the mailer definition run as this user.
`v`	Run in verbose mode.

x*LA*	Use *LA* as the system load average limit when deciding whether to queue messages rather than attempting to send them.
X*LA*	When the system load average exceeds LA, refuse incoming SMTP connections.
y*factor*	This fact*or* is multiplied by the number of recipients and added to the priority. Therefore, this value penalizes messages with large numbers of recipients.
Y	Deliver each job that is run from the queue in a separate process. Use this option if you are short of memory, since the default tends to consume considerable amounts of memory while the queue is being processed.
z*factor*	This fa*ctor* is multiplied by the message class (determined by the "Precedence:" field in the user header and the precedence declaration lines in the configuration file) and subtracted from the priority. Therefore, messages with a higher class will be favored.
Z*factor*	This f*actor* is added to the priority every time a message is processed. Therefore, this value penalizes messages that are processed frequently.

Mailer Flags

The following flags can be set in the *F* field of a mailer definition in the *sendmail.cf* file:

B	Don't wait for SMTP responses.

C	If mail is *received* from a mailer with this flag set, any addresses in the header that do not have an "at" sign (@) after being rewritten by rule set 3 have the "@domain" clause from the sender appended. This flag allows mail with headers of the form:
	`From: usera@hosta`
	`To: userb@hostb, userc`
	`to be rewritten automatically as:`
	`From: usera@hosta`
	`To: userb@hostb, userc@hosta`

D	This mailer wants a “Date:” header line.
E	Escape any lines beginning with “From” in the message with a “>” sign.
e	This mailer is expensive to connect to; usually, avoid connecting. Any necessary connection occurs during a queue run.
F	The mailer wants a “From:” header line.
f	The mailer wants a *-f from* flag, but only if this is a network forwarding operation. (That is, the mailer will give an error if the executing user does not have special permissions).
h	Uppercase should be preserved in host names for this mailer.
I	This mailer will be speaking SMTP to another *sendmail* and can use special protocol features. This option is not required; if it is omitted, the transmission still operates successfully, although perhaps not as efficiently as possible.
L	Limit the line lengths as specified in RFC 821.
l	This mailer is local; final delivery will be performed.
M	This mailer wants a “Message-Id:” header line.
m	This mailer can send to multiple users on the same host in one transaction. When a *$u* macro occurs in the *argv* part of the mailer definition, that field will be repeated as necessary for all qualifying users.
n	Do not insert a UNIX-style “From” line on the front of the message.
P	This mailer wants a “Return-Path:” line.
p	Use the return-path in the SMTP “MAIL FROM:” command rather than just the return address; although this usage is required in RFC 821, many hosts do not process return paths properly.
r	Same as *f*, but sends an *-r* flag.

S	Don't reset the userid before calling the mailer. Used in a secure environment where *sendmail* runs as root, this option could avoid forged addresses. This flag is suppressed if given from an "unsafe" environment (such as a user's *mail.cf* file).
s	Strip quotation characters from the address before calling the mailer.
U	This mailer wants UNIX-style "From" lines with the UUCP-style "remote from <host>" on the end.
u	Preserve uppercase characters in user names for this mailer.
V	Make all addresses UUCP !-relative to recipient/sender nodes. Addresses of the form r*ecipient_host!foo!bar* are rewritten as *foo!bar*. Addresses of the form *mumble!grumble* are rewritten as s*ender_host!mumble!grumble*.
X	This mailer wants to use the hidden-dot algorithm as specified in RFC 821; basically, any line beginning with a dot will have an extra dot prepended (to be stripped at the other end). This action ensures that a line containing a dot will not terminate the message prematurely.
x	This mailer wants a "Full-Name:" header line.

Support Files

This section provides a summary of the support files that *sendmail* creates or generates.

/usr/lib/sendmail
: The sendmail program.

/usr/lib/sendmail.cf
: The configuration file in textual form.

/usr/bsd/newaliases
: A link to */usr/lib/sendmail;* causes the alias database to be rebuilt. Running this program is equivalent to giving *sendmail* the *-bi* flag.

/usr/lib/sendmail.fc
: The configuration file represented as a memory image (the "frozen configuration").

/usr/lib/sendmail.hf
: The SMTP help file.

/usr/lib/sendmail.st
: A statistics file; need not be present.

/usr/lib/sendmail.killed
: A text file that contains the names of all known "dead" hosts (hosts that no longer exist or cannot receive mail for some reason).

/usr/lib/aliases
: The textual version of the alias file.

/usr/lib/aliases.{pag,dir}
: The alias file in *ndbm*(3B) format.

/var/spool/mqueue
: The directory in which the mail queue and temporary files reside.

*/var/spool/mqueue/qf**
: Control (queue) files for messages.

*/var/spool/mqueue/df**
: Data files.

*/var/spool/mqueue/tf**
: Temporary versions of the *qf* files, used during queue-file rebuild.

*/var/spool/mqueue/nf**
: A file used when a unique ID is created.

*/var/spool/mqueue/xf**
: A transcript of the current session.

/usr/bin/mailq
: Prints a listing of the mail queue; using this file is equivalent to using the *-bp* flag to *sendmail.*

/etc/init.d/mail
: Shell script for starting and stopping the *sendmail* daemon.

/bin/mail
: Program that *sendmail* uses as the "local" mailer.

Debugging Flags

The following list includes all known debugging flags. Flags that are especially useful are marked with an asterisk (*).

`0.1*`	Force daemon to run in foreground.
`0.4*`	Show known names for local host.
`0.15`	Print configuration file.
`0.44`	Have p*rintav*() print addresses of elements.
`1.1*`	Show mail "From" address for locally generated mail.
2.1*	Print exit status and envelope flags.
5.4	Print arguments to *tick*() calls.
5.5	Print arguments to *setevent*() and *clrevent*() calls.
5.6	Print event queue on *tick*() call.
6.1	Indicate call to *savemail*() or *returntosender*() error processing.
6.5	Trace states in *savemail()* state machine.
7.1*	Print information on envelope assigned to queue file.
7.2*	Print selected queue-file name.
7.20*	Print intermediate queue-file name selections.
8.1*	Print various information about resolver calls.
9.1*	Show results from *gethostbyaddr*() call.
10.1*	Print message delivery information.
11.1	Indicate call to *openmailer*().
12.1*	Display *remotename*() input and output.
13.1	s*endall*() - print addresses being sent to
13.3	s*endall()* - print each address in loop looking for failure.
13.4	s*endall()* - print who gets the error.
14.2	Indicate *commaize*() calls.
15.1	Indicate port or socket number used by *getrequests*().

15.2	Indicate when *getrequests*() forks or returns.
15.15	Set DEBUG socket option in *getrequests*().
16.1*	Indicate host, address, and socket being connected to in *makeconnection*().
16.14	Set DEBUG socket option in *makeconnection*().
18.1*	Show SMTP chatter.
18.100	Suspend *sendmail* after reading each SMTP reply.
20.1*	Display *parseaddr*() input and output.
21.2*	Show rewrite rule-set subroutine calls/returns and input/output, and display run-time macro expansions.
21.3*	Indicate rewrite subroutine call from inside rewrite rule.
21.4*	Display rewrite results.
21.10*	Indicate rule failures.
21.12*	Indicate rule matches and display address-rewrite steps.
21.15*	Show rewrite substitutions.
21.35	Display elements in pattern and subject.
22.36	Display *prescan*() processing.
22.45	Display more *prescan*() processing.
22.101	Display even more *prescan*() processing.
25.1*	Show “To” list designations.
26.1*	Show recipient designations/duplicate suppression.
26.6*	Show recipient password-match processing.
27.1*	Print alias and forward transformations and errors.
27.3	Print detailed *aliaslookup*() information.
30.1	Indicate End Of Headers when collecting a message.
30.2	Print arguments to *eatfrom*() calls.
30.3	Indicate when adding an “Apparently-To” header to the message.

31.6	Indicate call to *chompheader*() and header to be processed.
32.1	Display collected header.
33.1	Display *crackaddr*() input/output.
35.9*	Display macro definitions.
35.24	Display macro expansions.
36.5	Show symbol table processing.
36.9	Show symbol table hash function result.
37.1*	Display options as set.
37.2*	Show rewrite class loading.
40.1*	Indicate queueing of messages and display queue contents.
40.4*	Display queue control file contents.
40.5*	Display information about message-controlling user.
41.2	Indicate *orderq*() failure to open control file.
45.1	Indicate *setsender*() calls.
50.1	Indicate *dropenvelope*() calls.
51.4	Don't remove transcript files (*qxAAXXXXX* files).
52.1	Indicate call to *disconnect*(); print I/O file descriptors.
52.5	Don't do disconnect.
60.1*	Print information about alias database accesses.
61.1*	Print information about MX record lookups.

Appendix E

BIND Standard Resource Record Format

The Berkeley Internet Name Domain (BIND) server uses a specific record format for the name server data files. This appendix details BIND's standard resource record format by resource record type.

Standard Resource Record Format

The records in the name server data files are called *resource records*. The Standard Resource Record (RR) Format is specified in RFC 1035. The standard format of resource records is:

```
{name} {ttl} addr-class Record Type Record-specific data
```

The first field is the name of the domain record. It must always start in column 1. For some RRs the name may be left blank, in which case it becomes the name of the previous RR. The second field is an optional time-to-live field, which specifies how long this data will be stored in the database. When this field is blank, the default time-to-live value is specified in the Start of Authority (SOA) resource record (described later in this section). The third field is the address class. Currently only the *IN* class (for Internet hosts and addresses) is recognized. The fourth field identifies the type of resource record. Subsequent fields depend on the type of RR.

Case is preserved in names and data fields when they are loaded into the name server. Comparisons and lookups in the name server database are not case sensitive.

If you specify TTLs for resource records, it is important that you set them to appropriate values. The TTL is the amount of time (in seconds) that a resolver will use the data from your server before it asks your server again. If you set the value too low, your server will get loaded down with repeat requests. If you set it too high, information you change will not get distributed in a reasonable amount of time.

Most host information does not change much over long time periods. A good way to set up your TTLs is to set them at a high value, and lower the value if you know a change is coming soon. You might set most TTLs between a day (86400) and a week (604800). When you know some data is changing in the near future, set the TTL for that RR to a low value, between an hour (3600) and a day, until the change takes place. Then reset it to its previous value. All resource records with the same name, class, and type should have the same TTL value.

The following characters have special meanings in resource records:

(blank)	A blank or tab character in the name field denotes the current domain.
@	A free-standing "at" sign (@) in the name field denotes the current origin.
.	A free-standing period in the name field represents the root domain name.
\x	The backslash designates that the special meaning of the character x does not apply. The x represents any character other than a digit (0–9). For example, use \.to place a dot character in a label.
\DDD	Each D is a digit; the complete string is the octet corresponding to the decimal number described by DDD. The octet is assumed to be text and is not checked for special meaning.
( )	Parentheses enclose group data that crosses a line. In effect, newlines are not recognized within parentheses. This notation is useful with SOA and WKS records.
;	A semicolon precedes a comment; the remainder of the line is ignored.
*	An asterisk is a wildcard character.

Usually a resource record has the current origin appended to the name if the name is not terminated by a period (.). This scheme is useful for appending the current domain name to the data, such as workstation names, but can cause problems if you do not want the name to be appended. If the name is not in the domain for which you are creating the data file, end the name with a period. However, do not append the period to Internet addresses.

$INCLUDE

An include line has $INCLUDE starting in column 1 and is followed by a file name. This feature helps you use multiple files for different types of data. For example:

```
$INCLUDE /usr/etc/named.d/mailboxes
```

This line is a request to load the file */usr/etc/named.d/mailboxes.* The $INCLUDE command does not cause data to be loaded into a different zone or tree. It allows data for a given zone to be organized in separate files. For example, you might keep mailbox data separate from host data by using this mechanism.

$ORIGIN

$ORIGIN changes the origin in a data file. The line starts in column 1 and is followed by a domain origin. This feature is useful for putting more then one domain in a data file.

```
$ORIGIN Berkeley.EDU.
```

SOA – Start Of Authority

```
name {ttl} addr-class SOA   Source             Person-in-
charge
@    IN    SOA   ucbvax.Berkeley.EDU kjd.ucbvax.Berkeley.EDU.
                   (
                   1994021501;Serial
                   10800   ;Refresh
                   3600    ;Retry
                   3600000 ;Expire
                   86400   ;Minimum
                   )
```

The name is the name of the zone. It can be a complete domain name like “Berkeley.EDU.” or a name relative to the current $ORIGIN. The “at” sign (@) indicates the current zone name, taken from the “primary” line in the *named.boot* file or from a previous $ORIGIN line.

The Start of Authority record, SOA, designates the start of a zone. There should only be one SOA record per zone.

Source is the name of the host on which the master data file resides, typically the primary master server.

Person-in-charge is the mailing address for the person responsible for the name server. The mailing address is encoded in the form of a domain name where the "at" sign (@) separating the user name from the host name is replaced with a period. In the example above, *kjd.ucbvax.berkeley.edu* is the encoded form of *kjd@ucbvax.berkeley.edu*.

Serial is the version number of this data file, and should be incremented whenever data are changed. Do not use floating point numbers (numbers with a decimal point, such as 1.1). A useful convention is to encode the current date in the serial number. For example, *25 April 1994 edit #1* is encoded as:

```
1994042501
```

Increment the edit number if you modify the file more than once on a given day.

Refresh indicates how often, in seconds, a secondary name server is to check with the primary name server to see if an update is needed.

Retry indicates how long, in seconds, a secondary server is to retry after a failure to check for a refresh.

Expire is the maximum number of seconds that a secondary name server has to use the data before they expire for lack of getting a refresh.

Minimum is the default number of seconds to be used for the time-to-live field on resource records with no explicit time-to-live value.

NS – Name Server

```
{name}  {ttl}  addr-class  NS  Name server's_name
               IN         NS  ucbarpa.Berkeley.EDU.
```

The Name Server record, *NS*, lists the name of a machine that provides domain service for a particular domain. The name associated with the RR is the domain name and the data portion is the name of a host that provides the service. Workstations providing name service need not be located in the *named* domain. There should be one NS record for each master server

(primary or secondary) for the domain. If you use more than approximately 10 to 15 NS records for a zone, you may exceed DNS datagram size limits.

NS records for a domain must exist in both the zone that delegates the domain and in the domain itself. If the name server host for a particular domain is itself inside the domain, then a glue record is needed. A glue record is an Address (A) record that specifies the address of the server. Glue records are needed only in the server delegating the domain, not in the domain itself. For example, if the name server for domain SRI.COM is KL.SRI.COM, then the NS and glue A records on the delegating server look like this:

```
SRI.COM.     IN   NS      KL.SRI.COM.

KL.SRI.COM. IN   A       10.1.0.2
```

The administrators of the delegating and delegated domains should ensure that the NS and glue A records are consistent and remain so.

A – Address

```
{name}    {ttl}     addr-class   A      address
ucbvax              IN           A      128.32.133.1
                    IN           A      128.32.130.12
```

The Address record, *A,* lists the address for a given workstation The name field is the workstation name, and the address is the network address. There should be one A record for each address of the workstation.

HINFO – Host Information

```
{name}  {ttl}  addr-class  HINFO  Hardware          OS
               IN         HINFO  SGI-IRIS-INDY    IRIX
```

The Host Information resource record, *HINFO,* is for host-specific data. This record lists the hardware and operating system running at the listed host. Only a single space separates the hardware information and the operating-system information. To include a space in the workstation name, you must place quotation marks around the name. There should be one HINFO record for each host. See the file */usr/etc/named.d/README* for the current list of names for IRIS-4D Series workstations and servers. To learn the names for

other types of hardware and operating systems, refer to the most current "Assigned Numbers" RFC (RFC 1060 as of this writing).

WKS – Well-Known Services

```
{name} {ttl} addr-class WKS address    protocol services list
             IN         WKS 192.12.6.16  UDP   (who route
                                                timed domain)
             IN         WKS 192.12.63.16 TCP   (echo telnet
                                                chargen ftp
                                                smtp time
                                                domain bootp
                                                finger sunrpc)
```

The Well-Known Services record, *WKS*, describes well-known services supported by a particular protocol at a specified address. The list of services and port numbers comes from the list of services specified in */etc/services.* There should be only one *WKS* record per protocol per address.

CNAME – Canonical Name

```
aliases  {ttl}  addr-class  CNAME  Canonical name
ucbmonet        IN          CNAME  monet
```

The Canonical Name resource record, *CNAME*, specifies an alias for the official, or canonical, host name. This record should be the only one associated with the alias name. All other resource records should be associated with the canonical name, not with the alias. Any resource records that include a domain name as their value (such as NS or *MX*) should list the canonical name, not the alias.

Aliases are also useful when a host changes its name. In that case, it is usually a good idea to have a *CNAME* record so that people still using the old name gets to the right place.

PTR – Domain Name Pointer

```
name  {ttl}  addr-class  PTR  real name
6.130        IN          PTR  monet.Berkeley.EDU.
```

A Domain Name Pointer record, *PTR*, allows special names to point to some other location in the domain. The example of a *PTR* record given here is used

to set up reverse pointers for the special IN-ADDR.ARPA domain. This line is from the example *named.rev* file; see section 3.8.8. *PTR* names should be unique to the zone. Note the period (.) appended to the real name to prevent *named* from appending the current domain name.

MB – Mailbox

```
name  {ttl}  addr-class  MB  Machine
ben          IN          MB  franklin.Berkeley.EDU.
```

The Mailbox record, *MB,* lists the workstation where a user receives mail. The *name* field is the user's login. The machine field lists the workstation to which mail is to be delivered. Mailbox names should be unique to the zone.

MR – Mail Rename Name

```
name {ttl} addr-class MR corresponding_MB
Postmaster IN         MR  ben
```

The Mail Rename Name record, *MR,* lists aliases for a user. The *name* field lists the alias for the name listed in the last field, which should have a corresponding MB record.

MINFO – Mail Information

```
name  {ttl}  addr-class MINFO requests      maintainer
BIND         IN        MINFO  BIND-REQUEST  kjd.Berkeley.EDU
```

The Mail Information record, *MINFO,* creates a mail group for a mailing list. This resource record is usually associated with a Mail Group (*MG*) record, but can be used with a Mailbox (*MB*) record. The *name* is the name of the mailbox. The *requests* field is where mail such as requests to be added to a mail group should be sent. The *maintainer* is a mailbox that should receive error messages. This arrangement is appropriate for mailing lists when errors in members' names should be reported to someone other than the sender.

MG–Mail Group Member

```
{mail group name}  {ttl}    addr-class  MG  member name
                            IN         MG  Bloom
```

The Mail Group record, *MG*, lists members of a mail group. Here is an example for setting up a mailing list:

```
Bind   IN  MINFO    Bind-Request        kjd.Berkeley.EDU.
       IN  MG       Ralph.Berkeley.EDU.
       IN  MG       Zhou.Berkeley.EDU.
```

MX – Mail Exchanger

```
                                  preference mailer
name {ttl}     addr-class MX value  xchanger
Munnari.OZ.AU. IN         MX 10     Seismo.CSS.GOV.
*.IL.          IN         MX 10     CUNYVM.CUNY.EDU.
```

The Mail Exchanger record, *MX*, specifies a workstation that can deliver mail to a workstation that is not directly connected to the network. In the first example given here, Seismo.CSS.GOV. is a mail gateway that can deliver mail to Munnari.OZ.AU. Other systems on the network cannot deliver mail directly to Munnari. The two systems, Seismo and Munnari, can have a private connection or use a different transport medium. The preference value is the order that a mailer should follow when there is more then one way to deliver mail to a single workstation. See RFC 974 for more detailed information.

You can use a wildcard name containing an asterisk (*) for mail routing with an MX record. Servers on the network can state that any mail to a given domain is to be routed through a relay. In the second example given here, all mail to hosts in the domain IL is routed through CUNYVM.CUNY.EDU. This routing is done by means of a wildcard MX resource record, which states that *.IL has an MX of CUNYVM.CUNY.EDU.

RP – Responsible Person

```
owner {ttl} addr RP box_domain_name    TXT_domain_name
franklin    IN   RP franklin.berkeley.edu admin.berkeley.edu.
```

The Responsible Person record, *RP,* identifies the name or group name of the responsible person for a host. Often it is desirable to be able to identify the responsible entity for a particular host. Otherwise, when that host is down or malfunctioning, it is difficult to contact someone who can resolve the problem or repair the host.

The *mbox domain name* field is a domain name that specifies the mailbox for the responsible person. Its format in master files uses the DNS convention for mailbox encoding, identical to that used for the *Person-in-charge* mailbox field in the SOA record. In the example given here, the *mbox domain name* shows the encoding for ben@franklin.berkeley.edu. You can specify the root domain name (just ".") to indicate that no mailbox is available.

The last field is a domain name for which *TXT* RRs exist. You can perform a subsequent query to retrieve the associated *TXT* resource records at the *TXT* domain name. Retrieving the *TXT* records provides a level of indirection so that the entity can be referred to from multiple places in the DNS. You can specify the root domain name (just ".") for the *TXT* domain name to indicate that no associated *TXT* RR exists. In the example, "sysadmins.berkeley.edu." is the name of a *TXT* record that could contain some text with names and phone numbers.

The format of the *RP* record is class insensitive. Multiple *RP* records at a single name may be present in the database. They should have identical TTLs.

The *RP* record is experimental; not all DNS servers implement or recognize it.

TXT – Text

```
text-name   {ttl}   addr-class   TXT  text-data
location            IN          TXT  "Berkeley, CA"
```

The Text record, *TXT*, is used to hold descriptive text. The semantics of the text depends on the domain where it is found.

Index

Index

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z

We'd Like to Hear From You

As a user of Silicon Graphics documentation, your comments are important to us. They help us to better understand your needs and to improve the quality of our documentation.

Any information that you provide will be useful. Here is a list of suggested topics to comment on:

- General impression of the document
- Omission of material that you expected to find
- Technical errors
- Relevance of the material to the job you had to do
- Quality of the printing and binding

Please include the title and part number of the document you are commenting on. The part number for this document is 007-0603-100.

Thank you!

Three Ways to Reach Us

The **postcard** opposite this page has space for your comments. Write your comments on the postage-paid card for your country, then detach and mail it. If your country is not listed, either use the international card and apply the necessary postage or use electronic mail or FAX for your reply.

If **electronic mail** is available to you, write your comments in an e-mail message and mail it to either of these addresses:

- If you are on the Internet, use this address: techpubs@sgi.com
- For UUCP mail, use this address through any backbone site: *[your_site]*!sgi!techpubs

You can forward your comments (or annotated copies of manual pages) to Technical Publications at this **FAX** number:

415 965-0964