IRIX™ Admin:
Backup, Security, and Accounting

Document Number 007-2862-001

CONTRIBUTORS

Written by John Raithel
Illustrated by Dany Galgani
Edited by Christina Cary
Production by Chris Everett
Cover design and illustration by Rob Aguilar, Rikk Carey, Dean Hodgkinson, Erik Lindholm, and Kay Maitz

IRIX™ Admin: Backup, Security, and Accounting
Document Number 007-2862-001

Contents

List of Figures

List of Tables

IRIX Admin Manual Set

This guide is part of the *IRIX Admin* manual set, which is intended for administrators: those who are responsible for servers, multiple systems, and file structures outside the user's home directory and immediate working directories. If you find yourself in the position of maintaining systems for others or if you require more information about IRIX™ than is in the end-user manuals, these guides are for you. The *IRIX Admin* guides are available through the IRIS InSight™ online viewing system. The set comprises these volumes:

- *IRIX Admin: Software Installation and Licensing*—Explains how to install and license software that runs under IRIX, the Silicon Graphics® implementation of the UNIX® operating system. Contains instructions for performing miniroot and live installations using Inst, the command line interface to the IRIX installation utility. Identifies the licensing products that control access to restricted applications running under IRIX and refers readers to licensing product documentation.
- *IRIX Admin: System Configuration and Operation*—Lists good general system administration practices and describes system administration tasks, including configuring the operating system; managing user accounts, user processes, and disk resources; interacting with the system while in the PROM monitor; and tuning system performance.
- *IRIX Admin: Disks and Filesystems*—Describes how to add, maintain, and use disks and filesystems. Discusses how they work, their organization, and how to optimize their performance.
- *IRIX Admin: Networking and Mail*—Describes how to plan, set up, use, and maintain the networking and mail systems, including discussions of sendmail, UUCP, SLIP, and PPP.
- *IRIX Admin: Backup, Security, and Accounting*—Describes how to back up and restore files, how to protect your system's and network's security, and how to track system usage on a per-user basis.
- *IRIX Admin: Peripheral Devices*—Describes how to set up and maintain the software for peripheral devices such as terminals, modems, printers, and CD-ROM and tape drives. Also includes specifications for the associated cables for these devices.
- *IRIX Admin: Selected Reference Pages* (not available in InSight)—Provides concise reference page (manual page) information on the use of commands that may be needed while the system is down. Generally, each reference page covers one command, although some reference pages cover several closely related commands. Reference pages are available online through the man(1) command.

About This Guide

"About This Guide" includes brief descriptions of the contents of this guide and an explanation of typographical conventions used, and refers you to additional sources of information you might find helpful.

This guide is written for system and network administrators responsible for IRIX backups, security, or accounting. If you are responsible for your personal workstation only, refer to the *Personal System Administrator's Guide* first for this information.

What This Guide Contains

IRIX Admin: Backup, Security, and Accounting documents data backup and recovery, host and network security, and host resource auditing and accounting for IRIX computer sites. It contains the following chapters:

Part I

Part I of this guide comprises three chapters on the following backup and recovery topics:

- Chapter 1, "Planning a Backup Strategy"—discusses types of backup media, tools available, and ideas on implementing a backup strategy.
- Chapter 2, "Backup and Recovery Procedures"—provides detailed information on each of the backup tools available and gives examples of their use.
- Chapter 3, "Troubleshooting Backup and Recovery"—provides information on types of backup errors, and explains some common error messages.

Part II

Part II of this guide covers system and network security and contains two chapters:

- Chapter 4, "System Security"—discusses how to implement local system security.

- Chapter 5, “Network Security”—discusses how to implement local area network security and network firewalls.

Part III

Part III of this guide covers system accounting and auditing and contains the following two chapters:

- Chapter 6, “Administering the System Audit Trail”—describes how to audit all events on an IRIX system.
- Chapter 7, “System Accounting”—describes how to track system usage.

Conventions Used in This Guide

These type conventions and symbols are used in this guide:

Bold	Keywords and literal command-line arguments (options/flags).
Italics	Backus-Naur Form entries, command monitor commands, executable names, filenames, IRIX commands, manual/book titles, new terms, onscreen button names, tools, utilities, variable command-line arguments, and variables to be supplied by the user in examples, code, and syntax statements
`Fixed-width type`	Error messages, prompts, and onscreen text.
`Bold fixed-width type`	User input, including keyboard keys (printing and nonprinting); literals supplied by the user in examples, code, and syntax statements (*see also* <>)
ALL CAPS	Environment variables.
“”	(Double quotation marks) Onscreen menu items and references in text to document section titles
()	(Parentheses) Following IRIX commands—surround reference page (man page) section number
[]	(Brackets) Surrounding optional syntax statement arguments

<>	(Angle brackets) Surrounding nonprinting keyboard keys, for example, **<Esc>**, **<Ctrl-D>**
#	IRIX shell prompt for the superuser (*root*)
%	IRIX shell prompt for users other than superuser
>>	Command Monitor prompt

How to Use This Guide

You will probably use the parts of this document separately.

If you are responsible for backups, refer to Part I. Read Chapter 1 if you have yet to implement a backup policy, Chapter 2 to learn details on the use of a particular backup tool, and Chapter 3 if you are having trouble with backups.

If you are responsible for security, read Part II, Chapter 4 for details on configuring IRIX host security, and Chapter 5 if you are responsible for network security as well.

If you are responsible for system auditing, read Part III, Chapter 6.

If you are responsible for monitoring system usage, read Part III, Chapter 7.

Additional Resources

The following the books, and network and product resources are available to help you establish system and network security.

Books

The following books provide additional information on system and network security.

- Steven Bellovin and William Cheswick. *Firewalls and Internet Security*. Addison-Wesley. ISBN 0-201-63357-4, 1994.
- Douglas Comer. *Internetworking with TCP/IP*. Prentice-Hall, Inc. ISBN 0-13-468505-9, second edition, 1991
- David A. Curry. *UNIX System Security*. Addison-Wesley. ISBN 0-201-56327-4, 1992.

- Simson Garfinkle and Eugene Spafford. *Practical UNIX Security.* O-Reilly & Associates, Inc. ISBN 0-937175-72-2, 1991.

Internet Resources

Various resources addressing security are provided on the Internet itself. Pointers (URLs) are provided here rather than including the information in full, as the material is frequently updated.

Internet resources relating to system security include answers to frequently asked questions (FAQs) from various newsgroups; documents concerning the history, practice, and theory of security; bulletins on new security issues; interactive mailing lists discussing security issues, and so on. Listed below are pointers to some of these resources.

World Wide Web Resources for System Security

Here are some URLs (universal resource locators) that can connect you to information to various sources of security information on the World Wide Web (WWW):

- *http://www.sgi.com/*—Silicon Surf. A good starting point for finding information and products available for Silicon Graphics platforms.
- *ftp://sgi.sgigate.com:~ftp/security*—Security patches for Silicon Graphics products.
- *ftp://ftp.uni-paderborn.de/doc/FAQ/comp.security.unix/*—General UNIX security FAQs.
- *http://www.alw.nih.gov/Security/*—Links to a wide variety of security-related resources.
- *http://www.telstra.com.au/info/security.html*—Many links to general network security information including security-related mailing lists.
- *http://www.sei.cmu.edu/SEI/programs/cert.html*—The Computer Emergency Response Team (CERT) Coordination Center was established by the Advanced Research Projects Agency to coordinate information regarding security threats for Internet users.
- *http://ciac.llnl.gov/*—The U.S. Department of Energy Computer Incident Advisory Capability page has links to advisory bulletins, mailing lists, documents, and more.
- *ftp://ftp.tis.com/pub/firewalls/faq.current*—Firewall FAQ. Frequently asked questions and answers concerning firewalls.

- *ftp://ftp.uni-paderborn.de/doc/FAQ/comp.security.unix/*—General UNIX security FAQ.
- *http://www.alw.nih.gov/Security*—Links to a wide variety of security-related resources including multiple FAQs.
- *http://www-ns.rutgers.edu/www-security/index.html*—A home page for security issues related to the World Wide Web.
- *http://neptune.tis.com/Home/NetworkSecurity/Toolkit.html*

  A toolkit for network security including source code for proxies.
- *ftp://ftp.nec.com/pub/security/socks.cstc/*

  Where to begin for looking into SOCKS proxies. A FAQ, the proxies, and other information are accessible from this URL.

Note that URLs change and some of these may already be out of date. Use a good WWW search tool and search for various key words such as “security,” “network security,” and “firewall” to find others.

USENET News Groups

Here are some news groups you can subscribe to that can help you keep up-to-date on security issues:

- comp.security.unix—General discussion of UNIX-related security issues.
- comp.security.announce—Announcements regarding security-related products and services.
- comp.sys.sgi.admin—Discussion of administration issues for users of Silicon Graphics products.
- comp.sys.sgi.announce—Announcements of new products and services of interest to the users of Silicon Graphics products.
- comp.security.firewalls—General discussion of network firewall issues for all platforms.

Commercial and Free Products

Contact your Silicon Graphics sales representative for information on the Gauntlet™ for IRIX and other security-related products. Silicon Graphics also has Netscape™ products, which support secure Internet access through encrypting and proxying servers.

Some additional products that are available are mentioned below, but note that mention here does not imply any endorsement by Silicon Graphics, and configuration and support of these products is either supplied by their vendors or by you.

- *http://neptune.tis.com/Home/NetworkSecurity/Toolkit.html*

  A toolkit for network security including source code for proxies.

- *ftp://ftp.nec.com/pub/security/socks.cstc/*

  Where to begin for looking into SOCKS proxies. A FAQ, the proxies, and other information are accessible from this URL.

Connecting to the Internet

The issues can be complex and confusing when trying to find the best way to connect to the Internet. Look at the Welcome page for the WebFORCE™ Netscape Navigator™ for the local link *Connecting to the Internet*, which provides basic information and pointers to help you if you have yet to establish an Internet connection.

PART ONE

Backup

Part 1, *Backup*, contains the following chapters:

Chapter 1
Planning a Backup Strategy

Chapter 2
Backup and Recovery Procedures

Chapter 3
Troubleshooting Backup and Recovery

Chapter 1

Planning a Backup Strategy

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 contains the following sections:

When you are familiar with backup and have addressed the needs of your site, refer to Chapter 2 for detailed information on the backup utilities that you plan to use.

Types of Backup Media

Some of the common types of backup media supported on Silicon Graphics, Inc., systems include:

- 1/4" cartridge tape, 4-track
- 8 mm cartridge
- DAT
- DLT

In addition to backup devices attached to any particular system, backup devices of various types and capacities may be accessible through network connections. Refer to your owner's guide for information on locally accessible devices, and the appropriate vendor documentation for network-accessible device information.

Certain limitations or conditions described in this chapter might not apply to your specific media. For example, if you back up a 350 MB filesystem 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 "IRIX Admin: Peripheral Devices.")

Choosing a Backup Tool

The IRIX system provides a variety of backup tools, and you should use whichever ones work best for you. 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 backup utility that is common to all the workstations.

IRIX provides the following utilities for backing up your data:

- System Manager
- bru(1M)
- Backup(1M) and Restore(1M), which use *bru*
- dump(1M) and restore(1M)
- xfsdump(1M) and xfsrestore(1M) for XFS™ filesystems
- tar(1M)
- cpio(1M)
- dd(1M)

Optional products for Silicon Graphics systems are also available. IRIS NetWorker™ is a scalable, full-featured data management tool for data backup and recovery. You can use IRIS NetWorker to back up data on high-end servers, or centrally manage backups for all your network workstations and file servers. See your Silicon Graphics sales representative for optional backup solutions.

Backup tools can be viewed as filesystem-oriented programs, like *bru*, *Backup*, and *dump,* or as file- and directory-oriented programs, like *tar* and *cpio.* While these utilities are not actually limited to one or the other, they are generally more convenient when used in this way. 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, but you would not normally use *dd* to create backups.

Table 1-1 summarizes the backup utilities available with IRIX.

Table 1-1 Backup Utilities Summary

Utility	Summary Description	Reference
System Manager	Graphical interface to *bru* utility. Most convenient and probably best tool if you are backing up only your own IRIX host.	*Personal System Administration Guide*
bru	Automatic file compression, space estimates, integrity checking. Prompts for additional media. May not be available on all hosts in a heterogeneous environment.	bru(1) reference page and "bru" on page 19
Backup and *Restore*	A command line "front end" to the *bru* utility.	Backup(1) and Restore(1) reference pages and "Backup and Restore" on page 23
dump and *restore*	Supports incremental backups and interactive restores. Standard UNIX backup utilities good in heterogeneous environments (but cannot back up XFS filesystems).	dump(1M) and restore(1M) reference pages and "dump and restore" on page 26
xfsdump and *xfsrestore*	Supports incremental backups, interactive restores, and interrupt recovery. Use instead of *dump* and *restore* on XFS filesystems.	xfsdump(1M) and xfsrestore(1M) reference pages and "xfsdump and xfsrestore" on page 30
tar	Most common UNIX backup utility historically and in current distribution, making it portable and thus widely used in very heterogeneous computer environments.	tar(1) reference page and "tar" on page 62
cpio	Flexible and standard UNIX command generally combined in command line pipes with other commands.	cpio(1) reference page and "cpio" on page 65
dd	Standard UNIX command to read input and write output with optional conversions.	dd(1M) reference page and "dd" on page 67

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" toolchest 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 filesystem 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 filesystems, then an incremental backup system reduces the number of tapes you use.

The following sections discuss the different aspects of backing up data.

When to Back Up Data and What to Back Up

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 does not want to reenter.

Root Filesystems

On most systems, the root filesystem is fairly static. You do not need to back it up as frequently as the */usr* filesystem.

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 filesystem backup utilities, such as the System Manager or *bru*, on a regular schedule avoids these problems.

A reasonable approach is to back up the root partition once a month. In addition to regular backups, here are some specific times to back up a root filesystem:

- 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, back up the root filesystem regularly.

User Filesystems

The */usr* filesystem, which often contains both system programs (such as in */usr/bin*) and user accounts, is usually more active than a root filesystem. 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.

Treat the */var* filesystem similarly—it contains data such as the contents of users' mailboxes.

Incremental Backups

Incremental backups can use fewer tapes to provide the same level of protection as repeatedly backing up the entire filesystem. They are also faster than backing up every file on the system.

An incremental scheme for a particular filesystem looks something like this:

1. On the first day, back up the entire filesystem. 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 and 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.

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:

system_name`:/dev/tape`

If required, specify an account on the remote device:

user`@`*system_name*`:/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

```
echo “./bus.schedule” | cpio -ovcO guest@alice:/dev/tape
```

Automatic Backups

You can use the *cron* utility to automatically back up filesystems 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 piece of media. If all the data doesn't fit, then someone must manually change backup media.

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 * * * /usr/sbin/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 the 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 "IRIX Admin: System Configuration and Operation."

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. Set the write protect switch on tapes you plan to store as soon as a tape is written, but remember to unset it when you are ready to overwrite a previously-used tape.

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 filesystem, 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 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 filesystem 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, 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. Write "bad" on the tape case before you throw it out so that someone doesn't accidentally try to use it. 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.

Chapter 2

Backup and Recovery Procedures

This chapter provides examples of how to use the various backup and recover tools described in Chapter 1.

All of the utilities discussed in this chapter support more options than can be shown here, but the examples combined with the discussions in Chapter 1 should provide enough information for you to choose and begin to use the tools best suited for your environment. For a complete description of the options available with a particular tool, refer to the reference page for that tool (for example, tar(1) for the *tar* command).

This chapter is divided into the following sections:

General 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 filesystems, respectively.

   Also, use good-quality media. Considering the value of your data, use the best quality media you can afford.

3. Run fsck(1M) first on EFS filesystems (if you are backing up an entire filesystem) to make sure you do not create a tape of a damaged filesystem. You must unmount a filesystem 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*).

4. The default tape device for any drives you may have is */dev/tape*. If you do not use the default device, you must specify a device in your backup command line.

5. Label your backups. If you plan to reuse the media, use pencil. Include the date, time, name of the system, 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.

6. 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.

7. Write-protect your media after you make the backup.

8. 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 9 for information on safely storing your backups.

System Backups

To make a backup of your system on any system with a graphical user interface, bring up the System menu on the System Toolchest and select the Backup and Restore menu item. From the Backup and Restore 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 Backup and Restore window are the easiest to make and use, and are accessible from the Recover System option on 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.

To make a backup of your system using an IRIX command, use the Backup(1) command. Although the *Backup* command is a front-end interface to the bru(1) command, *Backup* also writes the disk volume header on the tape so that the Recover System option can reconstruct the boot blocks, which are not written to the tape using other backup commands. For more information, see the section “Backup and Restore” on page 23.

Recovery After System Corruption

If your root filesystem is damaged and your system cannot boot, you can restore your system from the Recover System option on the System Maintenance Menu. This is the menu that appears when you interrupt the boot sequence before the operating system takes over the system. To perform this recovery, you need two things:

- Access to a CD that contains the IRIX release on your system.
- A full system backup tape (beginning in the root directory (/) and containing all the files and directories on your system) created using the Backup and Restore window or the Backup command as described in the section “System Backups” on page 13.

If you do not have a full system backup made with the *Backup* command or Backup and Restore window—and your *root* or *usr* filesystems are so badly damaged that the operating system cannot boot—you have to reinstall your system software and then read your backup tapes (made with any backup tool you prefer) over the freshly installed software.

You may also be able to restore filesystems from the miniroot. For example, if your root filesystem has been corrupted, you may be able to boot the miniroot, unmount the root filesystem, and then use the miniroot versions of *restore*, *xfs_restore*, *Restore*, *bru*, *cpio*, or *tar* to restore your root filesystem. Refer to the following discussions of these commands for details on how to use them.

To recover from system corruption using the Recover System option on the System Maintenance Menu, follow these steps:

1. When you first start up your machine or press the Reset button on the system, this message appears:

   ```
   Starting up the system...
   ```

   Click the *Stop for Maintenance* button or press **<Esc>** to bring up the System Maintenance menu.

2. Click the Recover System icon in the System Maintenance menu, or type:

   4

   This System Recovery menu appears or you see a graphical equivalent:

   ```
                          System Recovery...

                  Press <Esc> to return to the menu.

   1) Remote Tape  2) Remote Directory  3) Local CD-ROM  4) Local Tape

   Enter 1-4 to select source type, <esc> to quit,
   or <enter> to start:
   ```

3. Enter the menu item number or click the appropriate drive icon for the IRIX release CD or software distribution directory you plan to use.

   Note: As of IRIX 6.2, the Remote Tape and Local Tape options on the System Recovery window are no longer usable because bootable (miniroot) software distribution tapes are no longer supported.

   - If you have a CD-ROM drive connected to your system, enter **3** or click the *Local CD-ROM* icon, then click *Accept* to start.

     You then see a notifier prompting you to insert the media into the drive. Insert the IRIX CD that came with your system, then click *Continue*.

   - If you don't have a CD-ROM drive, you can use a drive that is connected to another system on the network. At the System Recovery menu, enter **2** or click the *Remote Directory* icon.

When a notifier appears asking you for the remote hostname, type the system's name, a colon (**:**), and the full pathname of the CD-ROM drive, followed by **/dist**. For example, to access a CD-ROM drive on the system *mars*, you would type:

```
mars:/CDROM/dist
```

Click *Accept* on the notifier window, then click *Accept* on the System Recovery window.

On systems without graphics, you are prompted for the host as above, then you see this menu:

```
1) Remote Tape 2)[Remote Directory] 3) Local CD-ROM 4) Local Tape
      *a) Remote directory /CDROM/dist from server mars.

Enter 1-4 to select source type, a to select the source, <esc>
to quit,
or <enter> to start:
```

Press **<Enter>**.

- If you are using a remote software distribution directory, enter **2** or click the *Remote Directory* icon.

  When a notifier appears that asks you to enter the name of the remote host, type the system's name, a colon (**:**), and the full pathname of the software distribution directory. For example:

  ```
  mars:/dist/6.2
  ```

  Click *Accept* on the notifier window, then click *Accept* on the System Recovery window.

  On systems without graphics, you are prompted for the host as above, then you see this menu:

  ```
  1) Remote Tape 2)[Remote Directory] 3) Local CD-ROM 4) Local Tape
        *a) Remote directory /dist/6.2 from server mars.

  Enter 1-4 to select source type, a to select the source, <esc>
  to quit,
  or <enter> to start:
  ```

  Press **<Enter>**.

4. The system begins reading recovery and installation from the CD. It takes approximately five minutes to copy the information that it needs. After everything is copied from the CD or remote directory to the system disk you see messages including:

```
************************************************************
*                                                          *
*                     CRASH    RECOVERY                    *
*                                                          *
************************************************************

You may type  sh  to get a shell prompt at most questions

Checking for tape devices
```

   The next message asks for the location of the tape drive that you will use to read a system backup tape you created prior to the system crash using the Backup and Restore tool on the System menu of the System Toolchest or using the Backup(1) script.

5. If you have a local tape device, you see this message:

   `Restore will be from` *tapename*`.  OK? ([y]es, [n]o): [y]`

   tapename is the name of the local tape device. Answer **y** if this is the correct tape drive and **n** if is not.

6. If you have a remote (network) tape device, no tape device was found, or you answered "no" to the question in the previous step, you see this message:

   `Remote or local restore ([r]emote, [l]ocal): [l]`

   - If you answer "remote," you have chosen to restore from the network, and you are then asked to enter the following information: the hostname of the remote system, the name of the tape device on the remote system, the IP address of the remote system, and the IP address of your system. The IP address must consist of two to four numbers, separated by periods, such as 192.0.2.1
   - If you answer "local," you have chosen a tape device that is connected to your system, and you are then asked to enter the name of the tape device.

7. When you see the following message, insert your most recent full backup tape, then press **<Enter>**.

```
Insert the first Backup tape in the drive, then
press (<enter>, [q]uit (from recovery), [r]estart):
```

8. There is a pause while the program identifies the filesystems on the tape and attempts to mount those filesystems under */root*. Then you see this message:

   ```
   Erase all old filesystems and make new ones (y, n, sh): [n]
   ```

   You have three choices:

   - Answer **n** for no. After additional prompts confirming the filesystems to be read, the files on the tape are extracted. The version of each file on the tape replaces the version, if any, on the disk even if the version on the disk is newer.
   - Answer **y** for yes. After additional confirming prompts and prompts about filesystem types, the system erases all of the filesystems and copies everything from your backup tape to the disk.
   - Answer **sh** to escape to a shell. You are now in the miniroot environment and can investigate the damage to the system or attempt to save files that have been created or modified since the backup tape was created. After exiting the shell, you have the opportunity to remake filesystems and/or read the backup tape.

9. After reading the full backup tape, this prompt gives you the opportunity to read incremental backup tapes:

   ```
   Do you have incremental backup tapes to restore ([y]es, [n]o
   (none)): [n]
   ```

   Insert another tape and answer **y** if you have additional tape, answer **n** otherwise.

10. This prompt gives you the opportunity to reboot your system if recovery is complete, begin the crash recovery process again at the beginning, or re-read your first backup tape:

    ```
    Reboot, start over, or first tape again? ([r]eboot, [s]tart,
    [f]irst) [r]
    ```

    If you are ready to reboot, answer **r**, otherwise choose **start** or **first**.

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 "IRIX Admin: Peripheral Devices."

The method of changing the system default tape device is to relink */dev/nrtape* to the desired device. Use the following procedure:

1. Enter the commands:

   ```
   cd /dev
   ls -l nrtape
   ```

   You see something similar to this:

   ```
   crw-rw-rw-    2 root     sys      144,1251 Aug  4 01:29 nrtape
   ```

   The major device numbers is 144, and the minor device numbers is 65.

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
   ```

Repeat this procedure for the */dev/tape* device.

Most programs use */dev/nrtape* or */dev/tape* 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.

bru

The *bru* utility provides a convenient single interface to perform functions that can only be accomplished using multiple utilities otherwise. For example, options to *bru* allow you to estimate the storage space a backup requires, compress the backup data, and examine backup archives. Refer to the bru(1) reference page for details on all *bru* defaults and options.

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* is the most widely accepted format. Refer to Table 1-1 for an overview of various backup utilities including *bru*. If your site has predominately IRIX workstations, and you don't need to move filesystem 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.

Saving Data With bru

With *bru*, it is a simple matter to estimate the space a backup requires, back up filesystems or individual directories and files, and compress backups as described in this section.

Note: XFS and *bru*: The **-K** option has been added to the *bru* command for use with files larger than 2 GB. If the **-K** option is not used, *bru* skips any files that it cannot compress to less than 2 GB and issues a warning. Note that use of this option can create *bru* archives that are not usable on non-XFS systems. The **-K** option can only be used in combination with the **-Z** (use 12-bit LZW file compression) option.

Estimating Space Required for Backup

Use the **-e** option with *bru* for an estimate of how much space is required for an archive, for example:

```
bru -e /usr
```

returns how much space is required for the */usr* filesystem backup.

Backing Up a Filesystem 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* filesystem:

```
bru -c /usr
```

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 filenames from another file:

```
bru -c - < listfile
```

where the file *listfile* is a list of file names to be backed-up.

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* filesystem that have been modified on or after November 26, 1990:

```
bru -c -n 26-Nov-90 /usr
```

Saving Files Using Data Compression

You can compress files as they are archived. Use the **-Z** flag:

```
bru -Z /usr
```

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 filenames and sizes before creating the archive.

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* filesystem:

   ```
   bru -c
   ```

2. Each day, back up the files that have changed since the previous daily backup:

   ```
   bru -c -n 25-Nov-95 /usr
   bru -c -n 26-Nov-95 /usr
   bru -c -n 27-Nov-95 /usr
   bru -c -n 28-Nov-95 /usr
   bru -c -n 29-Nov-95 /usr
   bru -c -n 30-Nov-95 /usr
   bru -c -n 1-Dec-95 /usr
   ```

3. Every week, back up the files that have changed since the last weekly backup:

   ```
   bru -c -n 25-Nov-95 /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.

Examining bru Archives

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. Refer to bru(1) for more information.

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 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 filesystem.

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 while sending the output to */dev/null*.

Restoring bru Archives

This section describes how to restore *bru* backups of filesystems and individual files.

Restoring a Filesystem 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 Individual Files With bru

To restore an individual file, type:

bru -x *filename*

If the file already exists on the filesystem, *bru* compares its modification date with that of the copy on tape. If the version of the file in the filesystem is more recent than the one on tape, *bru* does not extract the archived file. Note that *filename* must exactly match what *bru -t* displays.

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
```

Backup and Restore

The *Backup* and *Restore* utilities are front end interfaces to *bru*. They support remote hostname 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(1) reference pages.

If you are planning to use the System Maintenance menu *Recovery* option, 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*.

Estimating Space Required for Backup

Use the **-e** option with *bru* for an estimate of how much space is required for an archive, for example:

```
bru -e /usr
```

returns the amount of space required to back up the */usr* filesystem.

Saving Data with Backup

With *Backup*, you can back up files, directories, whole filesystems, and full systems on local or remote devices. Full system backups include the ability to recover a damaged volume header and also to back up only those files modified since a previous backup.

The syntax for the *Backup* command is:

Backup *[***-h** *hostname] [***-t** *device] [***-i***] directory_name* | *filename*

To back up an entire disk to the default tape device, enter:

```
Backup /
```

This *Backup* command archives the entire system. 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 names 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.

You can make a backup relative to the last full system backup by entering:

```
Backup -i /
```

To back up a specific filesystem, enter the directory name of the filesystem. For example, to back up the *usr* filesystem, enter the following:

```
Backup /usr
```

To use a remote tape drive, use the **-h** *hostname* option:

```
Backup -h guest@alice.cbs.tv.com:/dev/tape /usr/people/ralph
```

This would back up the directory */usr/people/ralph* on the */dev/tape* device on the host alice.cbs.tv.com. You must have at least *guest* login privileges on the remote system in order to use a remote tape drive.

To back up a file, enter the filename. For example:

```
Backup people.tar.Z
```

Files (and directories) are stored relative to the current directory if the backup is made with a relative pathname as shown in this example. Relative pathnames are those that do not begin with a slash (/) character. Pathnames that begin with a slash are known as *absolute* pathnames. For example, */usr/bin/vi* is an absolute pathname. The leading slash indicates that the pathname begins at the root directory of the system. In contrast, *work/special.project/chapter1* is a relative pathname since the lack of a leading slash indicates that the path begins with a directory name in the current directory.

Restoring Backup Archives With Restore

The *Restore* command is a shell script that uses *bru* to extract files from a backup (see "bru" on page 19). You can also use *Restore* to read tapes made using the graphical System Manager (see the *Personal System Administration Guide*).

Restoring a Filesystem With Restore

You can recover multivolume backups with *Restore*. Enter:

```
Restore
```

and you are prompted to insert the tape into the drive.

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 login privileges for the given account in order to extract data from a remote drive.

Files are restored into the current directory if the backup was made with relative pathnames. Relative pathnames are those that do not begin with a slash (/) character. Pathnames that begin with a slash are known as *absolute* pathnames. For example, */usr/bin/vi* is an absolute pathname. The leading slash indicates that the pathname begins at the root directory of the system. In contrast, *work/special.project/chapter1* is a relative pathname 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 pathname on the disk are overwritten during a restore operation even if they are more recent than the files on tape. You must be especially careful, then, if you are restoring files with absolute pathnames, because regardless of your current working directory, the file is restored where the pathname indicates.

For example, if the file you are restoring was backed up as */etc/passwd* and you are in the directory */tmp*, the file you restore overwrites the */etc/passwd* file. If the file you are restoring was backed-up as *passwd*, then restore the *passwd* file into */tmp*.

dump and restore

The *dump* and *restore* programs are standard filesystem backup utilities used on many UNIX systems. These commands are only used with EFS filesystems. Refer to “xfsdump and xfsrestore” on page 30 to dump and restore XFS filesystems. The *dump* program makes incremental backups of entire filesystems.

Use *restore* to retrieve files from a *dump* archive. With *restore*, you can restore an entire filesystem or specific files. It also has an interactive mode that lets you browse the contents of an archive, select specific files, and restore them.

Saving Data With dump

This section describes how to perform backups with the *dump* utility.

Saving a Filesystem 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** option to indicate an update.

This command backs up all files on the */usr* filesystem:

```
dump 0 /dev/usr
```

Note: The 0 in the example specifies the increment level. Refer to the next section for an explanation of level numbers.

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 filesystem.
- 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* filesystem:

```
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.

Restoring dump Archives With restore

This section describes how to use the *restore* command to restore files and filesystems backed up with the *dump* command.

Restoring a Filesystem With restore

Use *restore* to recover files and filesystems 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
```

If your root filesystem is damaged and needs to be completely restored, you should probably reinstall the system, then rebuild it by extracting selected files from backup tapes. You can also restore the root filesystem by booting the miniroot, unmounting the root filesystem, and then using *restore* in the miniroot to restore the root filesystem.

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 filesystem 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.

xfsdump and xfsrestore

This section describes how the *xfsdump* and *xfsrestore* utilities work and how to use them to back up and recover data on XFS filesystems. (The xfsdump(1M) and xfsrestore(1M) reference pages provide online information on these utilities.) Table 2-1 and Table 2-2 summarize when to use *xfsdump* and *xfsrestore* and when to use their EFS counterparts, dump(1M) and restore(1M).

Table 2-1 Filesystems and Dump Utilities

For a Filesystem of Type	Dump It Using
EFS	*dump*
XFS	*xfsdump*

Table 2-2 Filesystems and Restore Utilities

For a Dump Made Using	Restore It Using	On a Filesystem of Type
dump	*restore*	EFS or XFS
xfsdump	*xfsrestore*	EFS or XFS

Note than you can restore data in either EFS or XFS filesystems, but must use the restore utility that corresponds with the dump utility used to make the backup.

Features of xfsdump and xfsrestore

The *xfsdump* and *xfsrestore* utilities fully support XFS filesystems. With *xfsdump* and *xfsrestore*, you can back up and restore data using local or remote drives. You can back up filesystems, directories, and/or individual files, and then restore filesystems, directories, and files independently of how they were backed up. *xfsdump* also allows you to back up "live" (mounted, in-use) filesystems.

With *xfsdump* and *xfsrestore*, you can recover from intentional or accidental interruptions—this means you can interrupt a dump or restore at any time, and then resume it whenever desired. With *xfsrestore*, you can restore *xfsdump* data onto EFS filesystems. (*xfsdump* backs up mounted XFS filesystems only.) *xfsdump* and *xfsrestore* support incremental dumps, and multiple dumps can be placed on a single media object. The utilities can automatically divide a dump among multiple drives, and can restore a dump from multiple drives. This allows you to perform faster dumps and restores.

xfsdump and *xfsrestore* support XFS features including 64-bit inode numbers, file lengths, holes, and user-selectable extent sizes. They support multiple media types, all IRIX-supported file types (regular, directory, symbolic link, block and character special, FIFO, and socket), and retain hard links. *xfsdump* does not affect the state of the filesystem being dumped (for example, access times are retained). *xfsrestore* detects and bypasses media errors and recovers rapidly after encountering them. *xfsdump* does not cross mount points, local or remote.

xfsdump optionally prompts for additional media when the end of the current media is reached. Operator estimates of media capacity are not required and *xfsdump* also supports automated backups. *xfsdump* maintains an extensive online inventory of all dumps performed. Inventory contents can be viewed through various filters to quickly locate specific dump information. *xfsrestore* supports interactive operation, allowing selection of individual files or directories for recovery. It also permits selection from

among backups performed at different times when multiple dumps are available. Dump contents may also be viewed noninteractively.

Media Layout

The following section introduces some terminology and then describes the way *xfsdump* formats data on the storage media for use by *xfsrestore*.

Terminology

This section introduces terminology used in the rest of this chapter.

While *xfsdump* and *xfsrestore* are often used with tape media, the utilities actually support multiple kinds of media, so in the following discussions, the term *media object* is used to refer to the media in a generic fashion. The term *dump* refers to the result of a single use of the *xfsdump* command to output data files to the selected media object(s). An instance of the use of *xfsdump* is referred to as a *dump session*.

The dump session sends a single *dump stream* to the media object(s). The dump stream may contain as little as a single file or as much as an entire filesystem. The dump stream is composed of *dump objects*, which are:

- one or more *data segments*
- an optional *dump inventory*
- a *stream terminator*

The data segment(s) contains the actual data, the dump inventory contains a list of the dump objects in the dump, and the stream terminator marks the end of the dump stream. When a dump stream is composed of multiple dump objects, each object is contained in a *media file*. Some output devices, for example standard output, do not support the concept of media files—the dump stream is only the data.

Possible Dump Layouts

The simplest dump, for example the dump of a small amount of data to a single tape, produces a data segment and a stream terminator as the only dump objects. If the optional inventory object is added, you have a dump such as that illustrated in Figure 2-1. (In the data layout diagrams in this section, the optional inventory object is always included.)

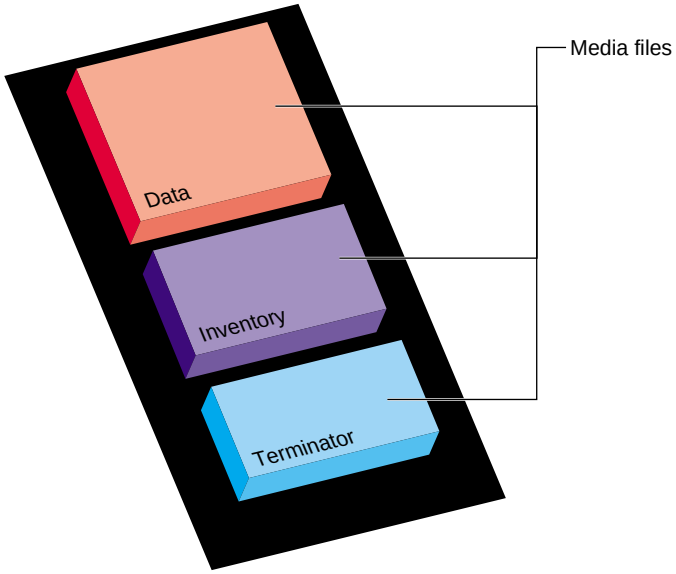

Figure 2-1 Single Dump on Single Media Object

You can also dump data streams that are larger than a single media object. The data stream can be broken between any two media files including data segment boundaries. (The inventory is never broken into segments.) In addition, if multiple drives are specified, the dump is automatically broken into multiple streams. The *xfsdump* utility prompts for a new media object when the end of the current media object is reached. Figure 2-2 illustrates the data layout of a single dump session that requires two media objects on each of two devices.

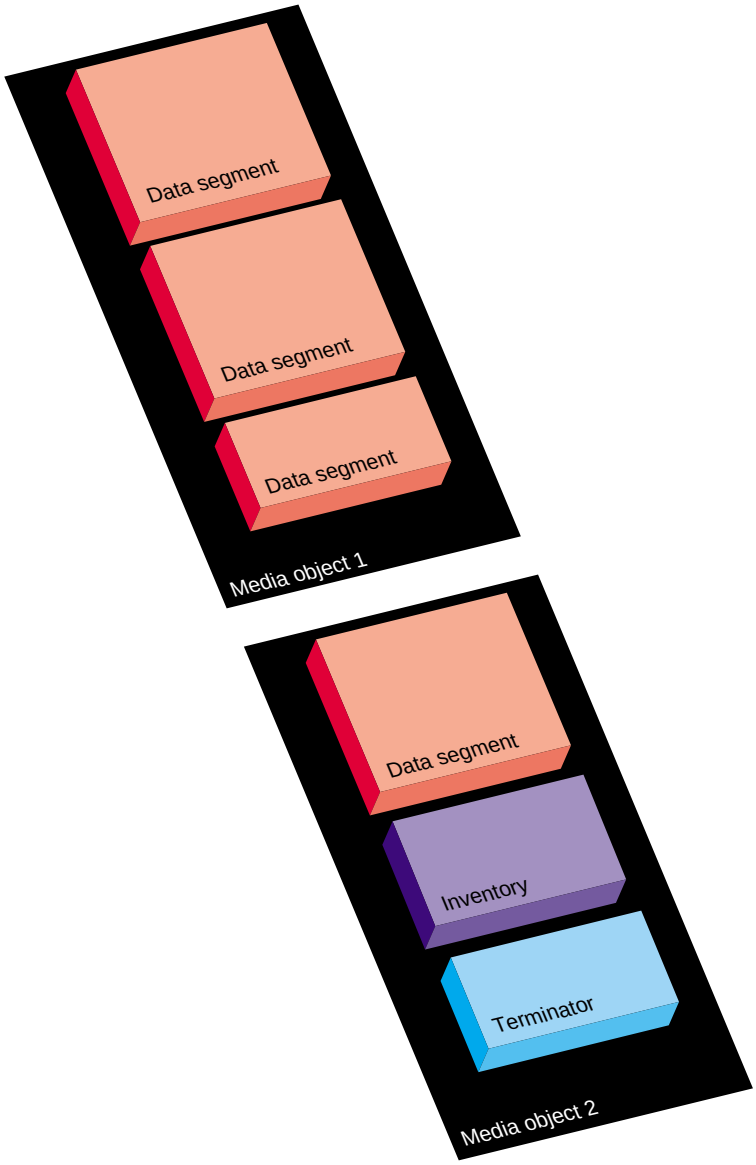

Figure 2-2 Single Dump on Multiple Media Objects

The *xfsdump* utility also accommodates multiple dumps on a single media object. When dumping to tape, for example, the tape is automatically advanced past the existing dump session(s) and the existing stream terminator is erased. The new dump data is then written, followed by the new stream terminator[1]. Figure 2-3 illustrates the layout of media files for two dumps on a single media object.

[1] For drives that do not permit termination to operate in this way, other means are used to achieve the same effective result.

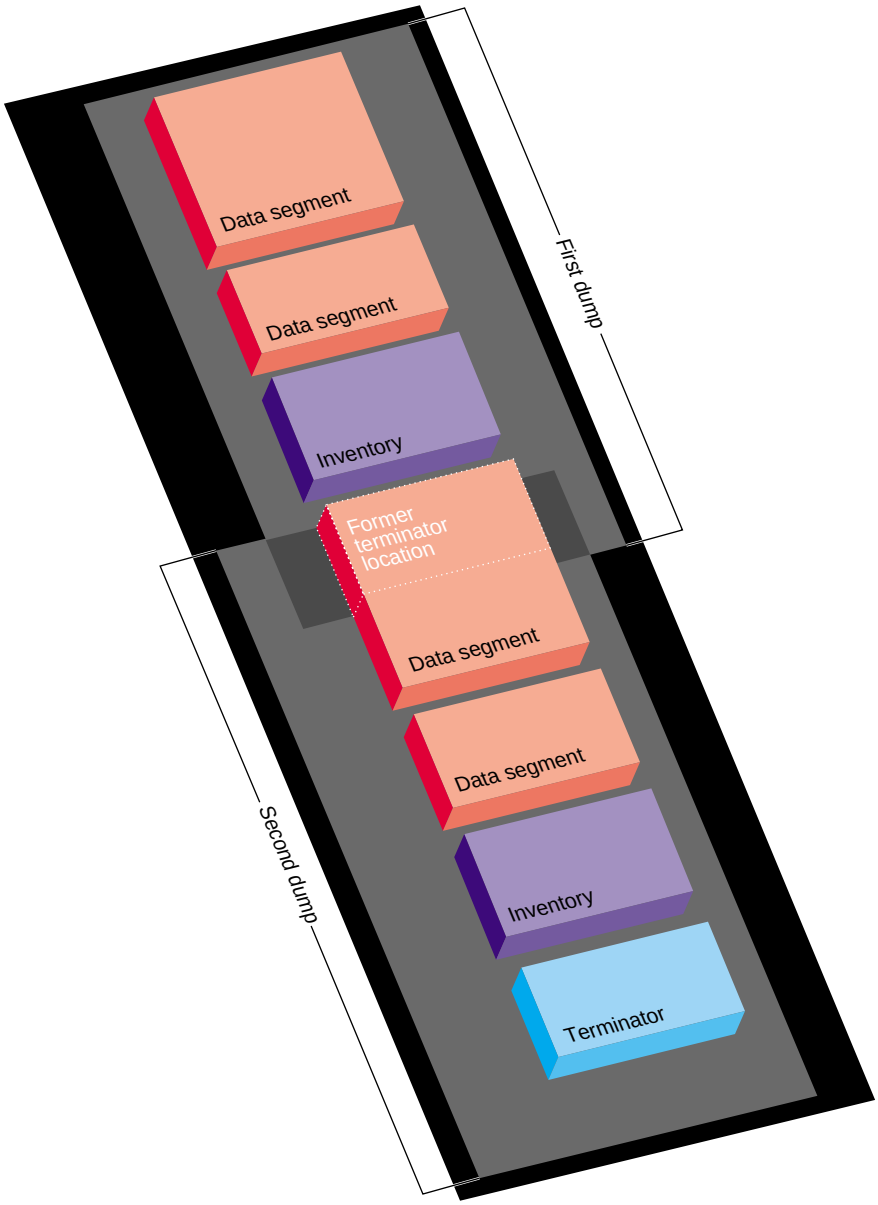

Figure 2-3 Multiple Dumps on Single Media Object

Figure 2-4 illustrates a case in which multiple dumps use multiple media objects. If media files already exist on the additional media object(s), the *xfsdump* utility finds the existing stream terminator, erases it, and begins writing the new dump data stream.

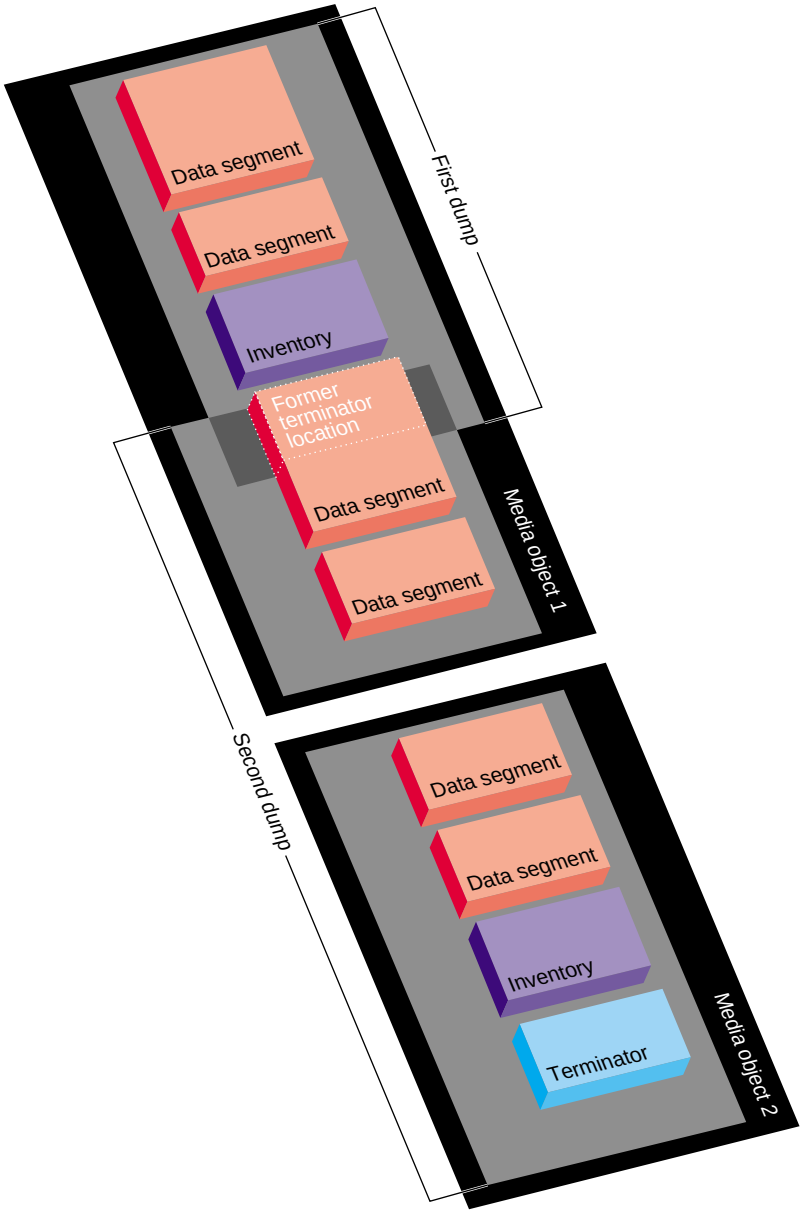

Figure 2-4 Multiple Dumps on Multiple Media Objects

Saving Data With xfsdump

This section discusses how to use the *xfsdump* command to back up data to local and remote devices. You can get a summary of *xfsdump* syntax with the **–h** option:

```
# xfsdump -h
xfsdump: version X.X
xfsdump: usage: xfsdump [ -f <destination> ... ]
              [ -h (help) ]
              [ -l <level> ]
              [ -p <seconds between progress reports> ]
              [ -s <subtree> ... ]
              [ -v <verbosity {silent, verbose, trace}> ]
              [ -A (don't dump extended file attributes) ]
              [ -B <base dump session id> ]
              [ -E (pre-erase media) ]
              [ -F (don't prompt) ]
              [ -I (display dump inventory) ]
              [ -J (inhibit inventory update) ]
              [ -L <session label> ]
              [ -M <media label> ... ]
              [ -O <options file> ]
              [ -R (resume) ]
              [ -T (don't timeout dialogs) ]
              [ -Y <I/O buffer ring length> ]
              [ - (stdout) ]
              [ <source (mntpnt|device)> ]
```

You must be the superuser to use *xfsdump*. Refer to the xfsdump(1M) reference page for details.

Specifying Local Media

You can use *xfsdump* to back up data to various media. For example, you can dump data to a tape or hard disk. The drive containing the media object may be connected to the local system or accessible over the network.

Following is an example of a level 0 dump to a local tape drive. Note that dump level does not need to be specified for a level 0 dump. (Refer to "Incremental Backups With dump" on page 27 for a discussion of dump levels.)

```
# xfsdump -f /dev/tape -L testers_11_21_94 -M test_1 /disk2
xfsdump: version 2.0 - type ^C for status and control
xfsdump: level 0 dump of cumulus:/disk2
xfsdump: dump date: Wed Oct 25 16:19:13 1995
xfsdump: session id: d2a6123b-b21d-1001-8938-08006906dc5c
xfsdump: session label: “testers_11_21_94”
xfsdump: ino map phase 1: skipping (no subtrees specified)
xfsdump: ino map phase 2: constructing initial dump list
xfsdump: ino map phase 3: skipping (no pruning necessary)
xfsdump: ino map phase 4: skipping (size estimated in phase 2)
xfsdump: ino map phase 5: skipping (only one dump stream)
xfsdump: ino map construction complete
xfsdump: preparing drive
xfsdump: creating dump session media file 0 (media 0, file 0)
xfsdump: dumping ino map
xfsdump: dumping directories
xfsdump: dumping non-directory files
xfsdump: ending media file
xfsdump: media file size 16777216 bytes
xfsdump: dumping session inventory
xfsdump: beginning inventory media file
xfsdump: media file 1 (media 0, file 1)
xfsdump: ending inventory media file
xfsdump: inventory media file size 4194304 bytes
xfsdump: writing stream terminator
xfsdump: beginning media stream terminator
xfsdump: media file 2 (media 0, file 2)
xfsdump: ending media stream terminator
xfsdump: media stream terminator size 2097152 bytes
xfsdump: I/O metrics: 3 by 2MB ring; 14/22 (64%) records streamed; 145889B/s
xfsdump: dump complete: 141 seconds elapsed
```

In this case, a session label (**–L** option) and a media label (**–M** option) are supplied, and the entire filesystem is dumped. Since no verbosity option is supplied, the default of *verbose* is used, resulting in the detailed screen output. The dump inventory is updated with the record of this backup because the **-J** option is not specified.

Following is an example of a backup of a subdirectory of a filesystem. In this example, the verbosity is set to *silent*, and the dump inventory is not updated (**–J** option):

```
# xfsdump -f /dev/tape -v silent -J -s people/fred /usr
```

Note that the subdirectory backed up (*/usr/people/fred*) was specified relative to the filesystem, so the specification did not include the name of the filesystem (in this case, */usr*). Since */usr* may be a very large filesystem and the **-v silent** option was used, this could take a long time during which there would be no screen output.

Specifying a Remote Tape Drive

To back up data to a remote tape drive, use the standard remote system syntax, specifying the system (by hostname if supported by a name server or IP address if not) followed by a colon (:), then the pathname of the special file.

Note: For remote backups, use the variable block size tape device if the device supports variable block size operation; otherwise, use the fixed block size device (see intro(7)).

The following example shows a subtree backup to a remote tape device:

```
# xfsdump -f magnolia:/dev/tape -L mag_10-95 -s engr /disk2
xfsdump: version 2.0 - type ^C for status and control
xfsdump: level 0 dump of cumulus:/disk2
xfsdump: dump date: Wed Oct 25 16:27:39 1995
xfsdump: session id: d2a6124b-b21d-1001-8938-08006906dc5c
xfsdump: session label: “mag_10-95”
xfsdump: ino map phase 1: parsing subtree selections
xfsdump: ino map phase 2: constructing initial dump list
xfsdump: ino map phase 3: pruning unneeded subtrees
xfsdump: ino map phase 4: estimating dump size
xfsdump: ino map phase 5: skipping (only one dump stream)
xfsdump: ino map construction complete
xfsdump: preparing drive
xfsdump: positioned at media file 0: dump 0, stream 0
xfsdump: positioned at media file 1: dump 0, stream 0
xfsdump: positioned at media file 2: dump 0, stream 0
xfsdump: stream terminator found
xfsdump: creating dump session media file 0 (media 0, file 2)
xfsdump: dumping ino map
xfsdump: dumping directories
xfsdump: dumping non-directory files
xfsdump: ending media file
xfsdump: media file size 6291456 bytes
xfsdump: dumping session inventory
xfsdump: beginning inventory media file
xfsdump: media file 1 (media 0, file 3)
xfsdump: ending inventory media file
```

```
xfsdump: inventory media file size 4194304 bytes
xfsdump: writing stream terminator
xfsdump: beginning media stream terminator
xfsdump: media file 2 (media 0, file 4)
xfsdump: ending media stream terminator
xfsdump: media stream terminator size 2097152 bytes
xfsdump: I/O metrics: 3 by 2MB ring; 12/22 (55%) records streamed; 99864B/s
xfsdump: dump complete: 149 seconds elapsed
```

In this case, */disk2/engr* is backed up to the variable block size tape device on the remote system *magnolia*. Existing dumps on the tape mounted on magnolia were skipped before recording the new data.

Note: The superuser account on the local system must be able to *rsh* to the remote system without a password. For more information, see hosts.equiv(4).

Backing Up to a File

You can back up data to a file instead of a device. In the following example, a file (*Makefile*) and a directory (*Source*) are backed up to a dump file (*monday_backup*) in */usr/tmp* on the local system:

```
# xfsdump -f /usr/tmp/monday_backup -v silent -J -s \
people/fred/Makefile -s people/fred/Source /usr
```

You may also dump to a file on a remote system, but note that the file must be in the remote system's */dev* directory. For example, the following command backs up the */usr/people/fred* subdirectory on the local system to the regular file */dev/fred_mon_12-2* on the remote system *theduke*:

```
# xfsdump -f theduke:/dev/fred_mon_12-2 -s people/fred /usr
```

Alternatively, you could dump to any remote file if that file is on an NFS-mounted filesystem. In any case, permission settings on the remote system must allow you to write to the file.

Refer to the section "Using xfsdump and xfsrestore to Copy Filesystems" on page 62 for information on using the standard input and standard output capabilities of *xfsdump* and *xfsrestore* to pipe data between filesystems or across the network.

Reusing Tapes

When you use a new tape as the media object of a dump session, *xfsdump* begins writing dump data at the beginning of the tape without prompting. If the tape already has dump data on it, *xfsdump* begins writing data after the last dump stream, again without prompting.

If, however, the tape contains data that is not from a dump session, *xfsdump* prompts you before continuing:

```
# xfsdump -f /dev/tape /test
xfsdump: version X.X - type ^C for status and control
xfsdump: dump date: Fri Dec 2 11:25:19 1994
xfsdump: level 0 dump
xfsdump: session id: d23cc072-b21d-1001-8f97-080069068eeb
xfsdump: preparing tape drive
xfsdump: this tape contains data that is not part of an XFS dump
xfsdump: do you want to overwrite this tape?
type y to overwrite, n to change tapes or abort (y/n):
```

You must answer **y** if you want to continue with the dump session, or **n** to quit. If you answer **y**, the dump session resumes and the tape is overwritten. If you do not respond to the prompt, the session eventually times out. Note that this means that an automatic backup, for example one initiated by a *crontab* entry, will not succeed—unless you specified the **-F** option with the *xfsdump* command, which forces it to overwrite the tape rather than prompt for approval.

Erasing Used Tapes

Erase preexisting data on tapes with the *mt erase* command. Make sure the tape is not write-protected.

For example, to prepare a used tape in the local default tape drive, enter:

```
# mt -f /dev/tape erase
```

Caution: This erases all data on the tape, including any dump sessions.

The tape can now be used by *xfsdump* without prompting for approval.

Incremental and Resumed Dumps

Incremental dumps are a way of backing up less data at a time but still preserving current versions of all your backed-up files, directories, and so on. Incremental backups are organized numerically by levels from 0 through 9. A level 0 dump always backs up the complete filesystem. A dump level of any other number backs up all files that have changed since a dump with a lower dump level number.

For example, if you perform a level 2 backup on a filesystem one day and your next dump is a level 3 backup, only those files that have changed since the level 2 backup are dumped with the level 3 backup. In this case, the level 2 backup is called the *base dump* for the level 3 backup. The base dump is the most recent backup of that filesystem with a lower dump level number.

Resumed dumps work in much the same way. When a dump is resumed after it has been interrupted, the remaining files that had been scheduled to be backed up during the interrupted dump session are backed up, and any files that changed during the interruption are also backed up. Note that you must restore an interrupted dump as if it is an incremental dump (see "Cumulative Restores With xfsrestore" on page 57).

In the following example, a new tape is used and the level 0 dump is the first dump written to it:

```
# xfsdump -f /dev/tape -l 0 -M Jun_94 -L week_1 -v silent /usr
```

A week later, a level 1 dump of the filesystem is performed on the same tape:

```
# xfsdump -f /dev/tape -l 1 -L week_2 /usr
```

The tape is forwarded past the existing dump data and the new data from the level 1 dump is written after it. (Note that it is not necessary to specify the media label for each successive dump on a media object.)

A week later, a level 2 dump is taken:

```
# xfsdump -f /dev/tape -l 2 -L week_3 /usr
```

and so on, for the four weeks of a month in this example, the fourth week being a level 3 dump (up to nine dump levels are supported). Refer to "Cumulative Restores With xfsrestore" on page 57 for information on the proper procedure for restoring incremental dumps.

You can interrupt a dump session and resume it later. To interrupt a dump session, type the interrupt character (typically **<CTRL-C>**). You receive a list of options which allow you to interrupt the session, change verbosity level, or resume the session.

In the following example, *xfsdump* is interrupted after dumping approximately 37% of a filesystem:

```
# xfsdump -f /dev/tape -M march95 -L week_1 -v silent /disk2

 ======================== status and control dialog ==========================

status at 16:49:16: 378/910 files dumped, 37.8% complete, 32 seconds elapsed

please select one of the following operations
1: interrupt this session
2: change verbosity
3: display metrics
4: other controls
5: continue (default) (timeout in 60 sec)
 -> 1

please confirm
1: interrupt this session
2: continue (default) (timeout in 60 sec)
 -> 1
interrupt request accepted

 -------------------------------- end dialog ---------------------------------

xfsdump: initiating session interrupt
xfsdump: dump interrupted prior to ino 1053172 offset 0
```

You can later continue the dump by including the **–R** option and a different session label:

```
# xfsdum -f /dev/tape -R -L week_1.contd -v silent /disk2p
```

Any files that were not backed up before the interruption, and any file changes that were made during the interruption, are backed up after the dump is resumed.

Note: Use of the **–R** option requires that the dump was made with a dump inventory taken, that is, the **–J** option was not used with *xfsdump*.

Examining xfsdump Archives

This section describes how to use the *xfsdump* command to view an *xfsdump* inventory.

The *xfsdump* inventory is maintained in the directory */var/xfsdump* created by *xfsdump*. You can view the dump inventory at any time with the *xfsdump –I* command. With no other arguments, *xfsdump –I* displays the entire dump inventory. (The *xfsdump -I* command does not require root privileges.)

The following output presents a section of a dump inventory.

```
# xfsdump -I | more
file system 0:
        fs id:          d23cb450-b21d-1001-8f97-080069068eeb
        session 0:
                mount point:    magnolia.abc.xyz.com:/test
                device:         magnolia.abc.xyz.com:/dev/rdsk/dks0d3s2
                time:           Mon Nov 28 11:44:04 1994
                session label:  ""
                session id:     d23cbf44-b21d-1001-8f97-080069068eeb
                level:          0
                resumed:        NO
                subtree:        NO
                streams:        1
                stream 0:
                        pathname:       /dev/tape
                        start:          ino 4121 offset 0
                        end:            ino 0 offset 0
                        interrupted:    YES
                        media files:    2
                        media file 0:
                                mfile index:    0
---more---
```

Notice that the dump inventory records are presented sequentially and are indented to illustrate the hierarchical order of the dump information.

You can view a subset of the dump inventory by specifying the level of depth (1, 2, or 3) that you want to view. For example, specifying depth=2 filters out a lot of the specific dump information as you can see by comparing the previous output with this:

```
# xfsdump -I depth=2
file system 0:
        fs id:          d23cb450-b21d-1001-8f97-080069068eeb
        session 0:
                mount point:    magnolia.abc.xyz.com:/test
                device:         magnolia.abc.xyz.com:/dev/rdsk/dks0d3s2
                time:           Mon Nov 28 11:44:04 1994
                session label:  ""
                session id:     d23cbf44-b21d-1001-8f97-080069068eeb
                level:          0
                resumed:        NO
                subtree:        NO
                streams:        1
        session 1:
                mount point:    magnolia.abc.xyz.com:/test
                device:         magnolia.abc.xyz.com:/dev/rdsk/dks0d3s2
               .
               .
               .
```

You can also view a filesystem-specific inventory by specifying the filesystem mount point with the *mnt* option. The following output shows an example of a dump inventory display in which the *depth* is set to *1*, and only a single filesystem is displayed:

```
# xfsdump -I depth=1,mnt=magnolia.abc.xyz.com:/test
filesystem 0:
        fs id:          d23cb450-b21d-1001-8f97-080069068eeb
```

Note that you can also look at a list of contents on the dump media itself by using the **–t** option with *xfsrestore*. (The *xfsrestore* utility is discussed in detail in the following section.) For example, to list the contents of the dump tape currently in the local tape drive, type:

```
# xfsrestore -f /dev/tape -t -v silent | more
xfsrestore: dump session found
xfsrestore: session label: "week_1"
xfsrestore: session id: d23cbcb4-b21d-1001-8f97-080069068eeb
xfsrestore: no media label
xfsrestore: media id: d23cbcb5-b21d-1001-8f97-080069068eeb
do you want to select this dump? (y/n): y
selected
one
A/five
people/fred/TOC
people/fred/ch3.doc
people/fred/ch3TOC.doc
people/fred/questions
A/four
people/fred/script_0
people/fred/script_1
people/fred/script_2
people/fred/script_3
people/fred/sub1/TOC
people/fred/sub1/ch3.doc
people/fred/sub1/ch3TOC.doc
people/fred/sub1/questions
people/fred/sub1/script_0
people/fred/sub1/script_1
people/fred/sub1/script_2
people/fred/sub1/script_3
people/fred/sub1/xdump1.doc
people/fred/sub1/xdump1.doc.backup
people/fred/sub1/xfsdump.doc
people/fred/sub1/xfsdump.doc.auto
people/fred/sub1/sub2/TOC
---more---
```

Restoring xfsdump Archives With xfsrestore

This section discusses the *xfsrestore* command, which you must use to view and extract data from the dump data created by *xfsdump*. You can get a summary of *xfsrestore* syntax with the **–h** option:

```
# xfsrestore -h
xfsrestore: version X.X
xfsrestore: usage: xfsrestore [ -a <alt. workspace dir> ... ]
                    [ -e (don't overwrite existing files) ]
                    [ -f <source> ... ]
                    [ -h (help) ]
                    [ -i (interactive) ]
                    [ -n <file> (restore only if newer than) ]
                    [ -o (restore owner/group even if not root) ]
                    [ -p <seconds between progress reports> ]
                    [ -r (cumulative restore) ]
                    [ -s <subtree> ... ]
                    [ -t (contents only) ]
                    [ -v <verbosity {silent, verbose, trace}> ]
                    [ -A (don't restore extended file attributes) ]
                    [ -C (check tape record checksums) ]
                    [ -D (restore DMAPI event settings) ]
                    [ -E (don't overwrite if changed) ]
                    [ -F (don't prompt) ]
                    [ -I (display dump inventory) ]
                    [ -J (inhibit inventory update) ]
                    [ -L <session label> ]
                    [ -N (timestamp messages) ]
                    [ -O <options file> ]
                    [ -P (pin down I/O buffers) ]
                    [ -Q (force interrupted session completion) ]
                    [ -R (resume) ]
                    [ -S <session id> ]
                    [ -T (don't timeout dialogs) ]
                    [ -U (unload media when change needed) ]
                    [ -V (show subsystem in messages) ]
                    [ -W (show verbosity in messages) ]
                    [ -X <excluded subtree> ... ]
                    [ -Y <I/O buffer ring length> ]
                    [ -Z (miniroot restrictions) ]
                    [ - (stdin) ]
                    [ <destination> ]
```

Use *xfsrestore* to restore data backed up with *xfsdump*. You can restore files, subdirectories, and filesystems—regardless of the way they were backed up. For example, if you back up an entire filesystem in a single dump, you can select individual files and subdirectories from within that filesystem to restore.

You can use *xfsrestore* interactively or noninteractively. With interactive mode, you can peruse the filesystem or files backed up, selecting those you want to restore. In noninteractive operation, a single command line can restore selected files and subdirectories, or an entire filesystem. You can restore data to its original filesystem location or any other location in an EFS or XFS filesystem.

By using successive invocations of *xfsrestore*, you can restore incremental dumps on a base dump. This restores data in the same sequence it was dumped.

Simple Restores With xfsrestore

A simple restore is a non-cumulative restore (for information on restoring incremental dumps, refer to "Cumulative Restores With xfsrestore" on page 57). An example of a simple, noninteractive use of *xfsrestore* is:

```
# xfsrestore -f /dev/tape /disk2
xfsrestore: version 2.0 - type ^C for status and control
xfsrestore: searching media for dump
xfsrestore: preparing drive
xfsrestore: examining media file 0

 ========================== dump selection dialog ===========================

the following dump has been found on drive 0

hostname: cumulus
mount point: /disk2
volume: /dev/rdsk/dks0d2s0
session time: Wed Oct 25 16:59:00 1995
level: 0
session label: "tape1"
media label: "media1"
file system id: d2a602fc-b21d-1001-8938-08006906dc5c
session id: d2a61284-b21d-1001-8938-08006906dc5c
media id: d2a61285-b21d-1001-8938-08006906dc5c

restore this dump?
1: skip
2: restore (default)
 -> 2
this dump selected for restoral

 -------------------------------- end dialog --------------------------------

xfsrestore: using online session inventory
xfsrestore: searching media for directory dump
xfsrestore: reading directories
xfsrestore: directory post-processing
xfsrestore: restoring non-directory files
xfsrestore: I/O metrics: 3 by 2MB ring; 9/13 (69%) records streamed; 204600B/s
xfsrestore: restore complete: 104 seconds elapsed
```

In this case, *xfsrestore* went to the first dump on the tape and asked if this was the dump to restore. If you had entered **1** for "skip," *xfsrestore* would have proceeded to the next dump on the tape (if there was one) and asked if this was the dump you wanted to restore.

You can request a specific dump if you used *xfsdump* with a session label. For example:

```
# xfsrestore -f /dev/tape -L Wed_11_23 /usr
xfsrestore: version X.X - type ^C for status and control
xfsrestore: preparing tape drive
xfsrestore: dump session found
xfsrestore: advancing tape to next media file
xfsrestore: dump session found
xfsrestore: restore of level 0 dump of magnolia.abc.xyz.com:/usr created Wed Nov
23 11:17:54 1994
xfsrestore: beginning media file
xfsrestore: reading ino map
xfsrestore: initializing the map tree
xfsrestore: reading the directory hierarchy
xfsrestore: restoring non-directory files
xfsrestore: ending media file
xfsrestore: restoring directory attributes
xfsrestore: restore complete: 200 seconds elapsed
```

In this way you recover a dump with a single command line and do not have to answer **y** or **n** to the prompt(s) asking you if the dump session found is the correct one. To be even more exact, use the **-S** option and specify the unique session ID of the particular dump session:

```
# xfsrestore -f /dev/tape -S \
d23cbf47-b21d-1001-8f97-080069068eeb /usr2/tmp
xfsrestore: version X.X - type ^C for status and control
xfsrestore: preparing tape drive
xfsrestore: dump session found
xfsrestore: advancing tape to next media file
xfsrestore: advancing tape to next media file
xfsrestore: dump session found
xfsrestore: restore of level 0 dump of magnolia.abc.xyz.com:/test resumed Mon
Nov 28 11:50:41 1994
xfsrestore: beginning media file
xfsrestore: media file 0 (media 0, file 2)
xfsrestore: reading ino map
xfsrestore: initializing the map tree
xfsrestore: reading the directory hierarchy
xfsrestore: restoring non-directory files
xfsrestore: ending media file
xfsrestore: restoring directory attributes
xfsrestore: restore complete: 229 seconds elapsed
```

You can find the session ID by viewing the dump inventory (see "Examining xfsdump Archives" on page 47). Session labels might be duplicated, but session IDs never are.

Restoring Individual Files with xfsrestore

On the *xfsrestore* command line, you can specify an individual file or subdirectory to restore. In this example, the file *people/fred/notes* is restored and placed in the */usr/tmp* directory (that is, the file is restored in */usr/tmp/people/fred/notes*):

```
# xfsrestore -f /dev/tape -L week_1 -s people/fred/notes \
/usr/tmp
```

You can also restore a file "in place" that is, restore it directly to where it came from in the original backup. Note, however, that if you do not use a **-e**, **-E**, or **-n** option, you overwrite any existing file(s) of the same name.

In the following example, the subdirectory *people/fred* is restored in the destination */usr*—this overwrites any files and subdirectories in */usr/people/fred* with the data on the dump tape:

```
# xfsrestore -f /dev/tape -L week_1 -s people/fred /usr
```

Network Restores with xfsrestore

You can use standard network references to specify devices and files on the network. For example, to use the tape drive on a network host named *magnolia* as the source for a restore, you can use the command:

```
# xfsrestore -f magnolia:/dev/tape -L 120694u2 /usr2
xfsrestore: version X.X - type ^C for status and control
xfsrestore: preparing tape drive
xfsrestore: dump session found
xfsrestore: advancing tape to next media file
xfsrestore: dump session found
xfsrestore: restore of level 0 dump of magnolia.abc.xyz.com:/usr2 created Tue
Dec 6 10:55:17 1994
xfsrestore: beginning media file
xfsrestore: media file 0 (media 0, file 1)
xfsrestore: reading ino map
xfsrestore: initializing the map tree
xfsrestore: reading the directory hierarchy
xfsrestore: restoring non-directory files
xfsrestore: ending media file
xfsrestore: restoring directory attributes
xfsrestore: restore complete: 203 seconds elapsed
```

In this case, the dump data is extracted from the tape on *magnolia*, and the destination is the directory */usr2* on the local system. Refer to the section "Using xfsdump and xfsrestore to Copy Filesystems" on page 62 for an example of using the standard input option of *xfsrestore*.

Interactive Restores With xfsrestore

Use the **–i** option of *xfsrestore* to perform interactive file restoration. With interactive restoration, you can use the commands *ls*, *pwd*, and *cd* to peruse the filesystem, and the *add* and *delete* commands to create a list of files and subdirectories you want to restore. Then you can enter the *extract* command to restore the files, or *quit* to exit the interactive restore session without restoring files. (The use of "wildcards" is not allowed with these commands.)

The following screen output shows an example of a simple interactive restoration.

```
# xfsrestore -f /dev/tape -i -v silent .
xfsrestore: dump session found
xfsrestore: no session label
xfsrestore: session id:    d23cbeda-b21d-1001-8f97-080069068eeb
xfsrestore: no media label
xfsrestore: media id:      d23cbedb-b21d-1001-8f97-080069068eeb
do you want to select this dump? (y/n): y
selected

 --- interactive subtree selection dialog ---

the following commands are available:
        pwd
        ls [ { <name>, ".." } ]
        cd [ { <name>, ".." } ]
        add [ <name> ]
        delete [ <name> ]
        extract
        quit
        help
 -> ls
            4122 people/
            4130 two
            4126 A/
            4121 one
 -> add two
 -> cd people
 -> ls
            4124 fred/
 -> add fred
 -> ls
    *       4124 fred/
 -> extract

 ---------------- end dialog ----------------
```

In the interactive restore session above, the subdirectory *people/fred* and the file *two* were restored relative to the current working directory (“.”). Note that an asterisk (*) in your *ls* output indicates your selections.

Cumulative Restores With xfsrestore

Cumulative restores sequentially restore incremental dumps to re-create filesystems and are also used to restore interrupted dumps. To perform a cumulative restore of a filesystem, begin with the media object that contains the base level dump and recover it first, then recover the incremental dump with the next higher dump level number, then the next, and so on. Use the **–r** option to inform *xfsrestore* that you are performing a cumulative recovery.

In the following example, the level 0 base dump and succeeding higher level dumps are on */dev/tape*. First the level 0 dump is restored, then each higher level dump in succession:

```
# /usr/tmp/xfsrestore -f /dev/tape -r -v silent .

 ========================== dump selection dialog ===========================

the following dump has been found on drive 0

hostname: cumulus
mount point: /disk2
volume: /dev/rdsk/dks0d2s0
session time: Wed Oct 25 14:37:47 1995
level: 0
session label: “week_1”
media label: “Jun_94”
file system id: d2a602fc-b21d-1001-8938-08006906dc5c
session id: d2a60b26-b21d-1001-8938-08006906dc5c
media id: d2a60b27-b21d-1001-8938-08006906dc5c

restore this dump?
1: skip
2: restore (default)
 -> <Enter>
this dump selected for restoral

 -------------------------------- end dialog --------------------------------

#
```

Next, enter the same command again. The program goes to the next dump and again you select the default:

```
# xfsrestore -f /dev/tape -r -v silent .

 ========================== dump selection dialog ===========================

the following dump has been found on drive 0

hostname: cumulus
mount point: /disk2
volume: /dev/rdsk/dks0d2s0
session time: Wed Oct 25 14:40:54 1995
level: 1
session label: “week_2”
media label: “Jun_94”
file system id: d2a602fc-b21d-1001-8938-08006906dc5c
session id: d2a60b2b-b21d-1001-8938-08006906dc5c
media id: d2a60b27-b21d-1001-8938-08006906dc5c

restore this dump?
1: skip
2: restore (default)
 -> <Enter>
this dump selected for restoral

 ------------------------------- end dialog --------------------------------
#
```

You then repeat this process until you have recovered the entire sequence of incremental dumps. The full and latest copy of the filesystem will then have been restored. In this case, it is restored relative to “.”, that is, in the directory you are in when the sequence of *xfsrestore* commands is issued.

Restore an interrupted dump just as if it were an incremental dump. Use the **-r** option to inform *xfsrestore* that you are performing an incremental restore, and answer **y** and **n** appropriately to select the proper “increments” to restore (see “Cumulative Restores With xfsrestore” on page 57).

Note that if you try to restore an interrupted dump as if it were a non-interrupted, non-incremental dump, the portion of the dump that occurred before the interruption is restored, but not the remainder of the dump. You can determine if a dump is an interrupted dump by looking in the online inventory.

Here is an example of a dump inventory showing an interrupted dump session (the crucial fields are in bold type):

```
# xfsdump -I depth=3,mobjlabel=AugTape,mnt=indy4.xyz.com:/usr
file system 0:
        fs id:          d23cb450-b21d-1001-8f97-080069068eeb
        session 0:
                mount point:    indy4.xyz.com.com:/usr
                device:         indy4.xyz.com.com:/dev/rdsk/dks0d3s2
                time:           Tue Dec  6 15:01:26 1994
                session label:  "180894usr"
                session id:     d23cc0c3-b21d-1001-8f97-080069068eeb
                level:          0
                resumed:        NO
                subtree:        NO
                streams:        1
                stream 0:
                        pathname:       /dev/tape
                        start:          ino 4121 offset 0
                        end:            ino 0 offset 0
                        interrupted:    YES
                        media files:    2
        session 1:
                mount point:    indy4.xyz.com.com:/usr
                device:         indy4.xyz.com.com:/dev/rdsk/dks0d3s2
                time:           Tue Dec  6 15:48:37 1994
                session label:  "Resumed180894usr"
                session id:     d23cc0cc-b21d-1001-8f97-080069068eeb
                level:          0
                resumed:        YES
                subtree:        NO
                streams:        1
                stream 0:
                        pathname:       /dev/tape
                        start:          ino 4121 offset 0
                        end:            ino 0 offset 0
                        interrupted:    NO
                        media files:    2
.
.
.
```

From this it can be determined that session 0 was interrupted and then resumed and completed in session 1.

To restore the interrupted dump session in the example above, use the following sequence of commands:

```
# xfsrestore -f /dev/tape -r -L 180894usr .
# xfsrestore -f /dev/tape -r -L Resumed180894usr .
```

This restores the entire */usr* backup relative to the current directory. (You should remove the *housekeeping* directory from the destination directory when you are finished.)

Interrupting xfsrestore

In a manner similar to *xfsdump* interruptions, you can interrupt an *xfsrestore* session. This allows you to interrupt a restore session and then resume it later. To interrupt a restore session, type the interrupt character (typically **<CTRL-C>**). You receive a list of options, which include interrupting the session or continuing.

```
# xfsrestore -f /dev/tape -v silent /disk2

 ========================== dump selection dialog ===========================

the following dump has been found on drive 0

hostname: cumulus
mount point: /disk2
volume: /dev/rdsk/dks0d2s0
session time: Wed Oct 25 17:20:16 1995
level: 0
session label: “week1”
media label: “newtape”
file system id: d2a602fc-b21d-1001-8938-08006906dc5c
session id: d2a6129e-b21d-1001-8938-08006906dc5c
media id: d2a6129f-b21d-1001-8938-08006906dc5c

restore this dump?
1: skip
2: restore (default)
 -> 2
this dump selected for restoral

 -------------------------------- end dialog --------------------------------


 ======================== status and control dialog ==========================
```

```
status at 17:23:52: 131/910 files restored, 14.4% complete, 42 seconds elapsed

please select one of the following operations
1: interrupt this session
2: change verbosity
3: display metrics
4: other controls
5: continue (default) (timeout in 60 sec)
 -> 1

please confirm
1: interrupt this session
2: continue (default) (timeout in 60 sec)
 -> 1
interrupt request accepted

 -------------------------------- end dialog --------------------------------

xfsrestore: initiating session interrupt
```

Resume the *xfsrestore* session with the **–R** option:

```
# xfsrestore -f /dev/tape -R -v silent /disk2
```

Data recovery continues from the point of the interruption.

housekeeping and orphanage Directories

The *xfsrestore* utility can create two subdirectories in the destination called *housekeeping* and *orphanage*.

The *housekeeping* directory is a temporary directory used during cumulative recovery to pass information from one invocation of *xfsrestore* to the next. It must not be removed during the process of performing the cumulative recovery but should be removed after the cumulative recovery is completed.

The orphanage directory is created if a file or subdirectory is restored that is not referenced in the filesystem structure of the dump. For example, if you dump a very active filesystem, it is possible for new files to be in the non-directory portion of the dump, yet none of the directories dumped reference that file. A warning message is displayed, and the file is placed in the *orphanage* directory, named with its original inode number and generation count (for example, 123479.14.).

Using xfsdump and xfsrestore to Copy Filesystems

You can use *xfsdump* and *xfsrestore* to pipe data across filesystems or across the network with a single command line. By piping *xfsdump* standard output to *xfsrestore* standard input you create an exact copy of a filesystem.

For example, to make a copy of */usr/people/fred* in the */usr2* directory, enter:

```
# xfsdump -J -s people/fred - /usr | xfsrestore - /usr2
```

To copy */usr/people/fred* to the network host *magnolia*'s */usr/tmp* directory:

```
# xfsdump -J -s people/fred - /usr | rsh magnolia \
xfsrestore - /usr/tmp
```

This creates the directory */usr/tmp/people/fred* on *magnolia*.

Note: The superuser account on the local system must be able to *rsh* to the remote system without a password. For more information, see hosts.equiv(4).

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.

Note: XFS and tar: The **-K** option is used with the tar(1) command for files larger than 2 GB. If the **-K** option is not used, *tar* skips any files larger than 2 GB and issues a warning. Note that use of this option can create *tar* archives that are not usable on non-XFS systems. The **-K** option cannot be used with the **-O** option, which creates *tar* archives formatted in an older, pre-POSIX format.

Saving Data With tar

This section describes how to backup files with the *tar* command, and how you can back up files that have been modified since a certain specific time, or relative to the last backup (incrementally).

Backing Up Files With tar

To back up individual files with *tar*, use the command:

`tar c` *file*

Saving Files by Modification Date

The *tar* command does not have the capability of saving files by modification date built in. However, you can use the *find* command to archive files that have not been modified in a particular number of days:

`find /usr -mtime 5 -local -type f -o -type` *othertypes* **`-print | tar cv -`**

The *find* command locates regular, local (non-NFS) files that have not been modified in five days. The *find* command sends its output to the *tar* command.

Incremental Backups With tar

Although *tar* does not have a built-in mechanisms for incremental backups, you can use other system commands to accomplish this task.

The following example uses the same incremental scheme presented in the preceding section to back up the */usr* filesystem. It uses the *find* command to determine which files to archive:

1. Go to the top of the filesystem that you want to back up. For example:

   ```
   cd /usr
   ```

2. Create a complete backup of the filesystem:

   ```
   tar cv .
   ```

3. Each day, back up the files that have changed since the previous daily backup:

   ```
   find /usr -mtime 1 -local -print | tar cvf -
   ```

4. Every week, back up the files that have changed since the last weekly backup:

   ```
   find /usr -mtime 7 -local -type f -print | tar cvf -
   ```

5. At the end of four weeks, perform a complete backup and start the process over.

Examining tar Archives

For *tar* archives, use the **v** keyword for verbose listing of the archive contents:

```
tar tv
```

You can compare files that are archived with the original 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 2-3.

Table 2-3 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

Restoring tar Archives

To recover individual files from a *tar* archive, specify the name of the files on the command line:

tar xv *file1 file2 directory/file3*

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 filesystem.

Note: XFS and cpio: Use the **-K** option with the cpio(1) command for files larger than 2 GB. If the **-K** option is not used, *cpio* skips any files larger than 2 GB and issues a warning. Note that use of this option can create *cpio* archives that are not usable on non-XFS systems. The **-K** option can be used only with the **-o** (output) option. The **-K** option cannot be used the **-c** option (which creates *cpio* archives with ASCII headers), or with the **-H** option (used to specify various header formats).

Saving Data With cpio

This section describes how to back up files with the *cpio* command, and how you can back up files that have been modified since a certain specific time, or relative to the last backup (incrementally).

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

The *cpio* command does not have the capability of saving files by modification date built in. However, you can use the *find* command to archive files that have not been modified in a particular number of days:

```
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.

Incremental Backups With 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 presented in the preceding section to back up the */usr* filesystem. It uses the *find* command to determine which files to archive:

1. Go to the top of the filesystem that you want to back up. For example:

   ```
   cd /usr
   ```

2. Create a complete backup of the filesystem:

   ```
   cpio -oLp .
   ```

3. Each day, back up the files that have changed since the previous daily backup:

   ```
   find /usr -mtime 1 -print | cpio -pdL
   ```

4. Every week, back up the files that have changed since the last weekly backup:

   ```
   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.

Examining cpio Archives

For *cpio* archives, use the following command to obtain a verbose listing:

```
cpio -itvI /dev/tape
```

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.

Restoring cpio Archives

To recover individual files from a *cpio* archive, specify the name of the file(s) on the command line:

`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.

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
- specify input and output block size
- copy a specific number of blocks
- perform various data conversions such as byte swapping

Refer to the dd(1M) reference page for details on the use of the *dd* command.

Chapter 3

Troubleshooting Backup and Recovery

From time to time you might experience backup failures. It is vitally important that you determine the cause of the failure. Most often, the failure is 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.

This chapter contains the following sections:

Unreadable Backups

The reasons a backup 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.

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

The mt(1M) command with these options tells you the block size used to write the tape. Set the block size correspondingly (or larger) when you use *dd* to read the tape. For example, if the block size 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 lowercase
- ucase–map alphabetics to uppercase

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 .
```

Note that you could also pipe the *dd* output to another local or remote tape drive (if available) if you do not need or want to create a disk file.

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 the 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 73.

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 filesystem.

If you used incremental backups, such as from *backup* or *bru*:

1. Make a complete backup of the current state of the filesystem. If you successfully recover the filesystem, 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 filesystem 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 filesystem 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.

Backup and Recovery Error Messages

Following are some of the possible error messages you may see that indicate problems with a backup or recovery.

```
unix: dks0d1s0: Process [tar] ran out of disk space
```

This error, or similar errors reporting a shortage of disk space, may occur if you are backing up data to a disk partition that does not have enough free space left to contain the data to be backed up.

Such errors may likewise occur in data restores if the data being recovered does not fit on the destination disk partition. Note that if you are uncompressing data that was compressed for backup, the uncompressed data could easily require twice as much space as the compressed data.

You may wish to add disk space, reclaim disk space, repartition existing disk space (see "IRIX Admin: Disks and Filesystems"), or redesign your backup procedure, for example, to use data compression (see "Saving Files Using Data Compression" on page 20).

```
unix: ec0: no carrier: check Ethernet cable

unix: NFS write error 151 on host garfield

unix: NFS2 getattr failed for server some.host.name: Timed out
```

These and similar network errors only represent a problem if you are using network resources (for example, a remote tape or disk drive) in your backup or recovery procedure. If this is the case, reestablish proper network connections (see "IRIX Admin: Networking and Mail") and either verify that your backup or recovery was successful or reinitiate it.

```
unix: Tape 3: Hardware error, Non-recoverable

mediad: Could not access device /dev/rmt/tps0d6nr, Device busy

unix: Tape 3: requires cleaning

unix: Tape 3: Unrecoverable media error

unix: NOTICE: SCSI tape #0, 6 had 1 successful retried commands (0% of
r/w)

unix: NOTICE: SCSI tape #0,7 Incompatible media when reading
```

These are all examples of tape access errors. Depending on whether you were trying to back up or recover data, the system encountered a problem writing or reading the tape. Be sure there is a tape in the drive indicated in the error message, and that it is not set on write-protect if you are attempting a backup. (Also, tape drives should be periodically cleaned according to manufacturer instructions.)

If these are not the problem, test the tape for read and/or write capabilities using one or more of the backup and recover utilities. Note that a media error can occur anywhere on a tape; to verify the tape, write and read the entire tape. You can also select "Run Confidence Tests" from the System toolchest and double-click on the Tape Drive test.

If you have any doubts about the quality of the tape you're using (for example, it is getting old), copy it to a new tape (if it still has good data) and discard it. If you are using a tape drive that you have not used before, verify that the tape type is compatible with the new drive. Run the mt(1) command to reset the tape drive. Run the hinv(1M) command to determine if the tape drive is recognized by the system.

A “device already in use” or “device busy” error probably means that some other program was using the tape drive when you tried to access it.

PART TWO

Security

Part II, *Security,* contains the following chapters:

Chapter 4
System Security

Chapter 5
Network Security

Chapter 4

System Security

This chapter deals with maintaining the security of your local computer system. Once you have initially established the security of a system, you can expand your secure area to include the network. But until you have local security, there is no point in trying to establish security over a larger area.

In addition, security is never established *finally*. That is, security is a dynamic process, requiring that you understand the issues, keep up to date on them, and continually monitor your system with the many tools available. It is the intention of this chapter and the next to give you the information you need to establish a security policy and begin its implementation.

Chapter 5 discusses *network* security issues, and is intended for use after the issues discussed in this chapter have been addressed to your satisfaction. If you are not connecting to the Internet or any large network scheme but only establishing or connecting to a small, local area network, you should still read the first part of Chapter 5, "Local Area Network Access" on page 107.

This chapter includes the following sections:

- "Accounts Shipped Without Passwords" on page 103 lists the user accounts in */etc/passwd* that do not contain passwords on the software as shipped.

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 you as the system administrator to balance the needs and concerns of the user community.

It is one thing to secure an isolated IRIX system, another to secure a local area network, and still another to secure a site that is connected to external networks such as the Internet. This chapter deals primarily with taking steps to secure an isolated system, but many of these same steps must also be taken before undertaking the more ambitious job of securing a network. For information regarding network security issues, refer to Chapter 5.

Standard Security Features

IRIX 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:

- file ownership split into three classes—*owner*, *group*, and *other*—permits the owner of files to specify who is allowed access to the files
- permissions split into three categories—*read*, *write*, and *execute*—permits the owner of files to specify the degree of access users may have to the files
- 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 in addition to using 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.
- 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.
- 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.

  For information about choosing passwords, see "Choosing Passwords" on page 84.
- 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, change critical passwords, including dialup passwords, whenever anyone leaves the organization. Former employees should not have access to workstations at the site.

- Systems with dialup ports should have special dialup accounts and passwords. This is very important for sites that have common-use accounts, as discussed above. Refer to the discussion on */etc/d_passwd* in "Second (Dialup) Passwords" on page 87.

  However, even with this added precaution, do 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)), take precautions to isolate access to a specific gateway workstation. Refer to "Network Security and Firewalls" on page 111 for details on connecting to external networks.

- 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 must know another person's password and therefore has full access to his or her account.

  Note: The file */var/adm/sulog* contains a log of all successful and unsuccessful attempts to use the *su* command (if it is enabled in */etc/default su*).

- 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 / (root), */bin*, */usr/bin*, and */etc* and the files in them are not writable except by the owner. This also prevents trojan horse attacks.
- 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*. Also, refer to the xlock(1) reference page for information on locking your local X display.
- Sensitive data files should be encrypted. The crypt(1) command, together with the encryption capabilities of the editors (*ed* and *vi*), provides some protection for sensitive information.
- 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 "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 with 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 hard copy map of the network to make it easier to spot unauthorized taps. Another way to make this sort of attack less likely is to use fiber-optic (FDDI) network hardware, which is much more difficult to tap. For details on configuring network software securely, refer to Chapter 5.

System security under IRIX is primarily dependent on system login accounts and passwords. Proper administration, user education, and use of the facilities provided 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 yield more security if the basic features are not used or are compromised by user actions*. Also, periodically log in with anonymous FTP to *sgigate.sgi.com* and look in the directory *~ftp/security* for any security patches for your system.

Note: If you are using NFS or NIS on your system, see the discussions in "Disabling NIS (YP)" on page 124 and "Limiting NFS Access" on page 125.

Password Administration

This discussion of password administration includes the following sections:

- "Choosing Passwords" on page 84 discusses how to choose more secure passwords and avoid easily guessed ones.
- "PROM Passwords" on page 85 discusses use of PROM passwords.
- "Second (Dialup) Passwords" on page 87 discusses how to associate a second password (often called system or dialup passwords) with specific tty lines.
- "Creating a Shadow Password File" on page 89 describes the use of a shadow password file to hide even the encrypted passwords contained in the standard password file.
- "Password Aging" on page 90 shows how to force users to choose new passwords at specified intervals.
- "Using pwck to Check the Password File" on page 93 introduces a useful tool that performs some useful password file checks.

Managing passwords is also described in "IRIX Admin: System Configuration and Operation."

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.

Also, 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.

More secure passwords are:

- long (the first eight characters are recognized)
- 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
- have all of these bulleted features

Easily guessed 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 other obvious information
- obvious—for example, "top secret," "secret," "private," "password," "friend," "key," "god," "me," and so on

PROM Passwords

Your system has a facility that allows you to require a password from users who attempt to gain access to the Command (PROM) Monitor. This gives you greater control over who may perform system administration tasks.

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. Alternatively, the intruder could simply remove the hard disk from your system and install it on another system 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.

Note that if you forget your PROM password, but you still know your *root* password, you can reset the PROM password on most systems 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.

To assign a new PROM password if you have forgotten it, first clear the existing PROM password from IRIX with the *nvram* command, and then assign a new one with the *passwd* command from the PROM monitor.

Clearing the PROM Password Using nvram

To clear 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 “”`

   Your PROM password is now cleared.

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. Type the *passwd* command and press **`<Enter>`**:

   ```
   passwd
   ```

   You see the prompt:

   ```
   Enter new password:
   ```

5. Enter the password you want for your system and press **`<Enter>`**. 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 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.

Refer to "Choosing Passwords" on page 84 for help in selecting a good 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 *clogin* 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
   ```

   All possible names for ports should be listed including links. Write the file and exit from 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). Return the account to this state when you are finished assigning a system password. (To

save an account's existing password, copy the password field of the account—the second field in */etc/passwd*—and replace it when you are finished with this procedure.)

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 **<Enter>**. The string "2themoon" is not displayed as you type it. Next you see:

```
Re-enter password:
```

Enter the string "2themoon" again and then press **<Enter>**. 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." (What you see may be different, even with the same password, depending on the "seed" the system uses to encrypt the password.)

5. Edit the file */etc/d_passwd*. In the file, place lines in the format:

   shell:password:

   shell is the command interpreter (shell) you wish to have a password, and *password* is the encrypted password. Make sure that all "shells" used in */etc/passwd* (the seventh and final field) are listed in this file, including those for UUCP, PPP, SLIP, and so on.

   For example, this command 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.

Note that you must make similar entries for any other login shells used on your system such as */bin/ksh*, */usr/local/bin/bash*, and */usr/bin/tcsh*.

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 using 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 are in effect. All standard password tools work transparently with shadow passwords. The difference should not be noticeable to your users, except that they cannot see the encrypted passwords in the */etc/passwd* file.

One difference in system operation is that older applications cannot get the proper value of *pw_passwd* from the getpwent(3C) and getpwnam(3C) library calls. This primarily affects "screen saver" programs, unless they have root privileges.

Note: Shadow passwords work differently with NIS. See the shadow(4) reference page for details on the use of shadow passwords with NIS.

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 new passwords every time their old ones expire. IRIX does not provide a utility that determines whether users are choosing from a set of passwords and, if so, then forces them to choose completely different passwords.

Note: Password aging is not supported for NIS entries (see passwd(4)).

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 with the following syntax:

passwd -x *max name*

where *max* is the maximum number of days the password is valid for the user *name*. For example, this command forces user *alice* to change her password every two weeks (14 days):

```
passwd -x 14 alice
```

If you set *max* to 0, the user must change her password when she next logs in, 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 4-1 lists the password aging codes and their meanings.

Table 4-1 Password Aging Character Codes

Character	Number of Weeks
. (period)	0 (zero)
/ (slash)	1
0 through 9	2 through 11

Table 4-1 (continued) Password Aging Character Codes

Character	Number of Weeks
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 example 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.`

Using pwck 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 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 89) 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.

Login and Account Administration

This section describes how to control special and login accounts. Special accounts are used by the system to perform specific system functions, and login accounts are user accounts allowing general-purpose system access.

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 has obtained access to one of the accounts, or was able to start a daemon on your system, could partially breach security. Partially, because 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. Either assign passwords to these accounts, or lock them using one of the methods described in "Locking Unused Logins" on page 94.

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 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 unless the system is a print server.

Locking Unused Logins

If a login is not used or needed, disable (lock) the login. You should not remove the account, though, because of the danger of reusing the UID in the future. User ID numbers are meant to be permanently associated with the person who used the account. If you reuse 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 damage your system. 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, the current entry in */etc/passwd* for the user *jones* might look like this:

```
jones:6.D/N3ZFGmq7U:3333:10:Jeremiah Jones:/usr/people/jones:/bin/tcsh
```

Enter the following command:

```
passwd -l jones
```

and the entry becomes:

```
jones:*LK*:3333:10:Jeremiah Jones:/usr/people/jones:/bin/tcsh
```

This command changes the password field of the entry in */etc/passwd* for account *jones* to `*LK*`. This blocks all logins to that account.

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 some unused logins in this manner. Be sure to check your */etc/passwd* file to be sure all logins have passwords or are disabled.

System Login Options

You can set the following login options to enhance security:

- Restrict *root* logins to a specific device, typically the system console.
- Specify the number of times an attempt to log in can fail before the login process is disabled.
- If the login process is disabled, specify how long before it can be resumed.

- Specify whether to maintain a log of logins and what information to store: all logins or only those that were unsuccessful.
- Specify whether to force a user who does not have a password to choose one immediately upon logging in.
- Specify whether or not to display to the user on login the date and time that user last logged in.

Login options are set in the file */etc/default/login,* which is a normal text file. The file contains one option specification per line. The options are described in the rest of this section.

Because the login procedure is your system's main defense against unauthorized access, login options are important. 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).

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 the login(1) reference page for further details.

Note that the visual login process clogin(1) does not provide these security options. To use the login security functions, you must turn off *clogin* and use the standard login processes, getty(1) and login(1). Use *chkconfig* to turn off the *visuallogin* and *xdm* configuration variables. See "IRIX Admin: System Configuration and Operation" 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 *clogin*.

Restricting root Logins

You can restrict *root* logins to a single device, forcing *root* users to either use that device or use the *su* command (thereby leaving a trail in */var/adm/sulog*). For example, edit */etc/default/login* to include the following line to restrict root logins to the system console:

```
CONSOLE=/dev/console
```

Note: Do not name */dev/syscon* or */dev/systty* as the device!

Maximum Login Attempts (MAXTRYS)

MAXTRYS is the number of times you let a login attempt fail before suspending the login. 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/default/login*. Place a line similar to this in the file:

```
MAXTRYS=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 (the DISABLETIME variable described in the next section), thus preventing further login attempts on that line for a while. The system default delay (DISABLETIME) is 20 seconds.

Following is an example login attempt that is disabled after three retries:

```
login: guest
password:
Login incorrect
login: guest
password:
Login incorrect
login: guest
password:
Login incorrect
```

At this point, no further login prompts are displayed until the period of time specified by DISABLETIME has passed.

Length of Time to Disable a Line (DISABLETIME)

Use this option along with the *MAXTRYS* option. To set the number of seconds after a certain number of unsuccessful login attempts that a line is disabled, edit the file */etc/default/login* and add a line similar to this:

```
DISABLETIME=30
```

This disables a line for 30 seconds. You can choose any value you consider appropriate for your system. The system default is 20 seconds.

Recording Login Attempts

You can record both successful and unsuccessful login attempts in the file */var/adm/SYSLOG*. To record all attempts to log in, place this line in the file */etc/default/login*:

```
SYSLOG=ALL
```

To record only unsuccessful attempts, place this line in the *login* file:

```
SYSLOG=FAIL
```

A large number of failed logins, especially with the same account name, may indicate that someone is trying to break into that account and thus into the system.

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/default/login*:

```
PASSREQ
```

Or, insert the following entry instead:

```
MANDPASS=YES
```

to prevent users from logging in if they do not already have a password.

Displaying the Last Login Time

Users can help maintain system security by noticing unauthorized use of their accounts. By default, the most recent login date, time, and the name of the terminal line (*tty* name)

or remote host from which the user logged in is displayed on login. 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*.

Users can stop the last login information from being displayed by having a *.hushlogin* file in their home directory, but they should be discouraged from doing so. Remind them periodically to look at the information each time they log in for any unusual information.

Set-UID and Set-GID Permissions

The set user identification (set-UID) and set group identification (set-GID) permissions must be used very carefully. When a user runs an executable file that has either of these permissions, 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-UID Files 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, 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  user  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 anyone in the group *user* can execute */usr/jbond/bin/sh* and become the superuser.

Checking for Set-UIDs in the root Filesystem

The following command line reports all files with a set-UID for the *root* filesystem (not just those owned by *root*) on EFS filesystems:

```
ncheck -s /dev/root | xargs ls -ld | cut -f2 | grep -v ~/dev/
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* filesystem. The output of the modified *ncheck* is then used as an argument to the *ls* command. The filesystem 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
```

For XFS filesystems, use the *find* command:

```
find / -perm -4000 -print
```

Checking Set-UIDs in Filesystems Other Than root

This example uses the *ncheck* command to examine the *usr* filesystem (*/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/people/jbond/bin/sh* should be investigated. This program is the only one 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 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          /usr/people/jbond/bin/sh
```

General File and Directory Permissions

Be conservative when establishing or changing permission bit settings on all files and directories. The safest settings do not allow write access, but where this is not possible, it may be possible to limit write access to the owner of the file or directory, or at least just to the owner and the group.

You should not be running the *rfindd* daemon, because it allows external access to your file, directory, and permissions listing. Use *chkconfig* to turn this daemon off if it is on. Refer to rfindd(1M) and chkconfig(1M) for more information.

Refer to the chmod(1) reference page for a discussion on setting the sticky bit on such directories as */tmp* to restrict removal and renaming of files.

The following files and directories are universally available for read and write access on IRIX as shipped. Depending on your site requirements, you may wish to change the permissions on these files to be more restrictive.

Caution: 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. Below is a partial list of such 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

Caution: Creating passwords on historically open accounts, such as *lp*, may cause certain related applications or operations to fail.

Security File and Command Reference

This section summarizes in two tables some IRIX files and commands that establish and control security. Table 4-2 lists the IRIX files concerned with security and Table 4-3 lists security-related commands.

Table 4-2 IRIX Security Files

File	Purpose	Reference
/etc/default/login	Control login actions	login(1)
/etc/default/su	Define *su* command defaults	su(1M)
/etc/passwd	Store password and account information	passwd(1), passwd(4)
/etc/shadow	Hide password information	shadow(4), pwconv(1M)
/var/adm/sulog	Log *su* command usage	su(1M)
/var/adm/SYSLOG	Log system messages	syslogd(1M)

Table 4-3 IRIX Security Commands

Command Example	Purpose	Reference
`arp -a`	Display current ARP entries	arp(1M), arp(7P)
`crypt` *password*	Encode/decode input/output	crypt(1)
`last`	Indicate last logins of users and terminals	last(1)
`ncheck(1M)`	Generate pathnames from i-numbers	ncheck(1M)
`passwd`	Change password	passwd(1), passwd(4)
`ps -elf`	Display a full, long list of every process currently running	ps(1)

Table 4-3 (continued) IRIX Security Commands

Command Example	Purpose	Reference
pwck	Report inconsistencies in */etc/passwd* file	pwck(1M), passwd(4)
sar	System activity reporter	sar(1), sar(1M)
satd	Reliably save the system audit trail	satd(1M) and "Placing the Audit Files" on page 143
vi -x	Edit encrypted file	vi(1), crypt(1)
w	Display users logged in with current activity	w(1)
who	Display users logged in, their tty, and time of login	who(1)

Chapter 5

Network Security

This chapter discusses various ways to make your network more secure. In general, you may need to establish policies regarding network access within a trusted local group, and other policies regarding access to and from external, untrusted networks such as the Internet.

This chapter contains the following sections:

- "Local Area Network Access" on page 107 discusses security issues to consider when creating or connecting to a local area network.
- "Network Security and Firewalls" on page 111 introduces issues concerned with how to build a "firewall," or barrier, between your local system or site and external, untrusted networks such as the Internet.
- "Hardware Configuration" on page 115 summarizes network hardware design configurations from the point of view of security.
- "IRIX Configuration" on page 119 describes the details of how to configure an IRIX host to serve as a firewall.
- "Internal Network Configuration" on page 127 summarizes issues relating to configuring Sendmail and DNS on your firewall and internal network.

Note: This chapter assumes you already have taken measures to secure your host system(s) as described in Chapter 4.

Local Area Network Access

Within your local area network, you may be able to allow a degree of internetwork access that is not possible or desirable with networks outside of your control. This section discusses the use of network host and user permission files to control access within your local network.

Controlling Network Access

Three files that help you control access to a host within your network are:

/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 and their encrypted passwords.

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 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 a non-privileged account such as *guest*. Note also that a *.rhosts* file with a system name "localhost" allows *su* to work without requiring passwords. Refer to su(1M) for more information.

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 automatically disregarded for security reasons.

You may wish to disallow use of *.rhosts* files altogether if connecting to an untrusted network (you can add the **-l** option to the *rshd* invocation in */etc/inetd.conf* and thereby disallow these files. See rshd(1M) for more information). The more secure configurations for such connections are as discussed later in this chapter under "Network Security and Firewalls" on page 111. For complete information about the */etc/hosts.equiv* and *.rhosts* files, see the hosts.equiv(4) reference page.

Limiting X11 Access

With the X Window System™, workstations can run client programs transparently on other hosts on the network. This access is completely independent of controls such 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 filename, 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. (This is the default.)

Next, the *xhost* command is run from the file */usr/lib/X11/xdm/Xsession*. In the default *Xsession* file, *xhost* allows 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*.hosts* file.

xhost Command

The *xhost* command 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.

For example, to deny other hosts access to the X server comment out the *xhost* line in */var/X11/xdm/Xsession* and */var/X11/xdm/Xsession.dt*:

```
# Gives anyone on any host access to this display
/usr/bin/X11/xhost +
```

to look like this:

```
# Gives anyone on any host access to this display
# /usr/bin/X11/xhost +
```

The *xhost* command can also be used interactively. 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 advise users not to use *xhost* +, or you may delete the command from the system if it is a perceived security risk.

X*.xhost File

You can selectively allow access to remote hosts by listing their names in the */etc/X*.hosts* file. For example, if the file */etc/X0.hosts* contains the following line, the remote host *bronx* is the only workstation allowed to access the local server for server 0:

```
bronx
```

In the above example, all other hosts are denied access to the local server—assuming you do not have a conflicting xhost command in the */var/X11/xdm/Xsession or /var/X11/xdm/Xsession.dt* file. The *xhost* command overrides the configuration file *X*.hosts*. To alter the default system configuration, you must not only modify the configuration file, but also change the *xhost* command in the */var/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 have to wait until connections are established with each of the hosts before the server is started.

For even better security (just commenting out *xhost +* still allows local programs to connect to the X server), you can enable X authority. To do this, change the `DisplayManager*authorize` entry in */var/X11/xdm/xdm-config* to say:

```
DisplayManager*authorize: on
```

This makes *xdm* generate "magic cookies" (put in each user's *$HOME/.Xauthority* file), which are then required for any X client to connect to the X server. This provides a good means of X server access control. (Note that this may already be the default on your system.)

For more information about X security and authorization, see the xsecurity(1), xhost(1), xauth(1), xserver(1), and X(1) reference pages.

Local inetd Services

The *inetd* process controls a number of network services that you may or may not want to support on your local area network. You can limit which services you offer and log access to those services by editing the */etc/inetd.conf* file. See "Limiting inetd Services" on page 121, but note that that discussion refers to limiting these services when connecting to an untrusted network. You may wish to be more lenient in your configuration of *inetd* if you are concerned only with determining policy for a trusted local network.

Network Security and Firewalls

After establishing your host and site security policies, you may want to connect your site to external networks such as the Internet. This section is concerned with establishing such a connection with an "untrusted" network in which you do not control security. This requires special consideration of the interface between your internal, trusted network and the external network. This interface, if you take steps to provide for security, is called a "firewall" and is the subject of this section. The remainder of the discussion refers specifically to connecting to the Internet, but you can also apply it to connecting to any untrusted network.

What is the Internet?

The Internet is a vast, connected network of heterogeneous computer resources, spanning the globe and growing daily. Increasingly, individuals and organizations are finding access to the Internet to be of importance for a wide variety of services and resources pertinent to their businesses and other interests, including electronic mail, access to vast information archives, and keeping abreast of current developments in a host of areas.

Undoubtedly the most recent spur to the growth of interest in Internet access is the development of the World Wide Web, which provides for both a "friendly" graphical interface to Internet resources and a standardized means of presenting and accessing them. Products designed for this market, such as WebFORCE, allow their users to establish an Internet presence that can be accessed from around the world.

The Internet presents ways to share data that you want to share, but you must take measures to protect data that you want protected. This section addresses an important aspect of this internetworked accessibility: the need to establish and maintain the security of local computers and computer networks. Specifically, computer sites have a need and a right to determine the privacy and safety of their data from competitive interests as well as outright software vandalism.

Network Security Issues

If you are connecting to the Internet, you should configure your connection so that you do not unwittingly risk the exposure or corruption of important data. You should know exactly which (if any) data you are making publicly accessible, and you should guard against the possibility of unwanted intruders gaining access to your site. The Internet has many known (and some famous) instances of unwanted intrusions, vandalism, and so on, and acknowledging and taking measures to prevent such acts is the best way to ensure that your Internet presence is a pleasurable and profitable one.

While it is beyond the scope of this chapter to detail particular instances of malicious or criminal activity on computer networks, a great deal of such information is available on the Internet itself, and makes for useful reading for those responsible for computer security (refer to "Additional Resources" on page xxi for pointers to additional information).

In general, you need to establish a line of defense between your trusted computer resources (your *internal* network) and the computer resources publicly accessible

through the Internet (the *external* network). This line of defense should shield you from direct, external access, and it may be as simple as a single router or computer host or as complex as multiple routers and an entire computer network. (This section is concerned with establishing the secure firewalls possible with a computer host or network, not with the limited firewall protection of a router-only configuration.) Behind this line, you choose the degree to which you want to allow internal, trusted users access to the Internet, and the degree to which external users can access internal resources.

What Is a Firewall?

The line between the external world of untrusted hosts and the internal world of trusted hosts is established by creating a firewall. A firewall is a combination of computer hardware and software that allows you to restrict interactions with the external network (often the Internet) to the degree you desire. The simple formula is the more access you allow, the greater the security concerns; the greater the restrictions you place on access, the easier it is to monitor and maintain security. The tradeoff is one of ease of use versus peace of mind. For system and network administrators, this often translates as balancing the wishes of users with the needs and capacities of the administrator(s).

An example of a simple firewall is shown in Figure 5-1 on page 114. In this illustration, a single computer host is configured with two network interfaces to become what is known as a dual-homed host—a host with a presence on each of two different networks. When it is configured as described in this chapter, it represents a single, controlled obstruction between your internal network and the Internet where you can focus your security efforts. In this chapter, the term firewall host refers to an IRIX host configured for network security. (Gauntlet™ for IRIX is an example of a commercial firewall implementation for IRIX—see your sales representative for details.)

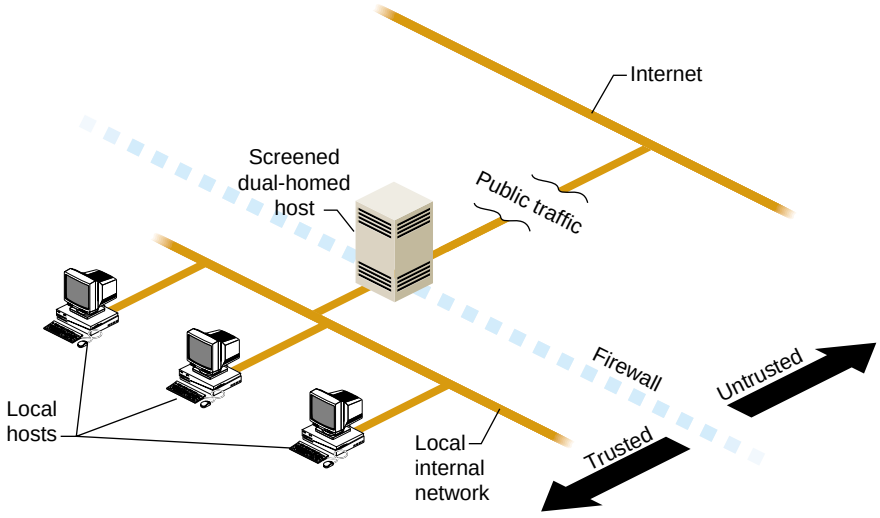

Figure 5-1 A Simple Firewall Environment

The firewall does not in any way restrict interactions on your internal network. Local hosts may share resources in the same way they did before connecting to the firewall. What is different now is how these hosts may interact with external sites as determined by your site policy—your policy determines how much or how little interaction is allowed. "Internal Network Configuration" on page 127 presents some scenarios of how you might configure a network with a dual-homed host.

Firewall Design Philosophy

The key to administering network security is the firewall. While there are important issues concerning internal security, those issues are the same regardless of whether or not you are connected to the Internet. (Refer to Chapter 4 for information on host security and "Local Area Network Access" on page 107 for information on local network security.)

Regarding the firewall itself:

- Limit users—if possible, limit users to the sole administrator of the system. If additional users are necessary, refer to Chapter 4 for a discussion of issues regarding password protection and educating users.

- Limit services—the more services you allow, the more possible security holes you present. In addition and in general, the more complex the software providing these services, the more chance for compromise, and the newer the software, the less chance it has been well tested in the "real world."
- Monitor the system—this document helps you configure the IRIX software of your firewall to maintain log files that can provide information on accesses to your firewall host, including time of access and unsuccessful access attempts. Also, make use of the many standard IRIX tools such as w(1), ps(1), and so on that give you snapshots of current system activities (see Table 4-3 on page 104 for a list of security-related commands).
- do not run applications on the firewall—any additional software, besides containing possible security weaknesses, further complicates the software environment, making security control more problematic.

World Wide Web Issues

There is the same security issue inherent in accessing software on the World Wide Web that has always been an issue when acquiring software from any unknown or untrusted source. When a user clicks on a browser button for a network resource, what is invoked is unknown. A click, for example, could download an executable file with a potential for damage. Users should be aware of this issue. If this is a serious concern at your site, you may consider isolating and limiting those hosts having World Wide Web access.

Refer to "Additional Resources" on page xxi for a pointer (URL) to additional information on security issues related to the World Wide Web.

Hardware Configuration

This section discusses how to configure network hardware to serve as the hardware portion of a firewall solution. (For information on how to configure Silicon Graphics software in a firewall solution, refer to "IRIX Configuration" on page 119.) Only setups that include an IRIX host as part of the solution are discussed, as router-only solutions tend to be too limited. A firewall host has the advantages of permitting and restricting specific applications, maintaining log files, and adding authentication to network access.

Routers and Firewalls

The firewall host is typically combined with a router, whether provided as part of your connection to your Internet service provider or added by you to your private configuration.

Routers, if properly configured, provide a certain degree of security by filtering IP packets. You can use your IRIX host as an IP packet filter as described in the ipfilterd(1M) reference page. Usually, routers are complete hardware devices that provide high-speed IP packet filtering. While many routers can be configured to provide IP packet-level security, they do not support such features as *proxies* and *authentication*.

Proxies are proxy servers, which provide for application specific control of network resources.[1] Authentication is a technique you can employ to require users to verify that they are who they say they are. To add these features and more, you must have a network hardware configuration such as the IRIX host setups described in the following sections.

You can use IP packet filtering *and* application-level controls by combining routers with firewalls. When using a router with a firewall host, configure it to allow traffic only to the firewall host. You should filter out:

- ICMP[2] redirects not from the router
- IP packets specifying the loose source routing option
- external packets claiming to be from the internal network (known as "spoofing"; see *http://www.msen.com/~emv/tubed/spoofing.html*).

Consult with your Internet service provider to determine the packet filtering options available for your Internet connection. You can also add routers to your firewall configuration as described in the next section, and then configure your routers with additional filtering options (refer to the router vendor documentation for details). (See also "Packet Filtering Gateways," in *Firewalls and Internet Security*, by Cheswick and Bellovin, referenced in "Additional Resources" on page xxi.)

---

[1] For example, the Netscape Proxy Server™ offers application proxies for several common network services including World Wide Web HTTP servers.

[2] Internet Control Message Protocol

Configuring SGI Hardware for Use as a Firewall

This section discusses general hardware configuration issues for the basic setup of a dual-homed host acting as the firewall, and then presents the “screened host” and “screened subnet” firewall configurations.

Dual-Homed Host Firewall

You can configure your Silicon Graphics host hardware for use in a firewall by making it a dual-homed gateway—that is, giving it two network connections. Figure 5-1 on page 114 illustrates the general idea of using a dual-homed host as the firewall.

Creating a dual-homed host may involve, for example, adding an additional Ethernet controller board, or you may already have two Ethernet connections. For specific information on the network hardware in your system, refer to your system documentation.

Screened Host Gateway

A screened host scenario uses a router to screen traffic between the Internet and the external network connection of the firewall host. Routers vary, but in general, they screen IP packets for certain addresses or settings that they have been programmed to disallow. They can further limit traffic to a few ports of the firewall host. No traffic is allowed from the outside to any other host on the internal network. This is the typical connection to the Internet in which the router is provided by the Internet service provider. Figure 5-2 on page 118 illustrates the basic screened host scenario.

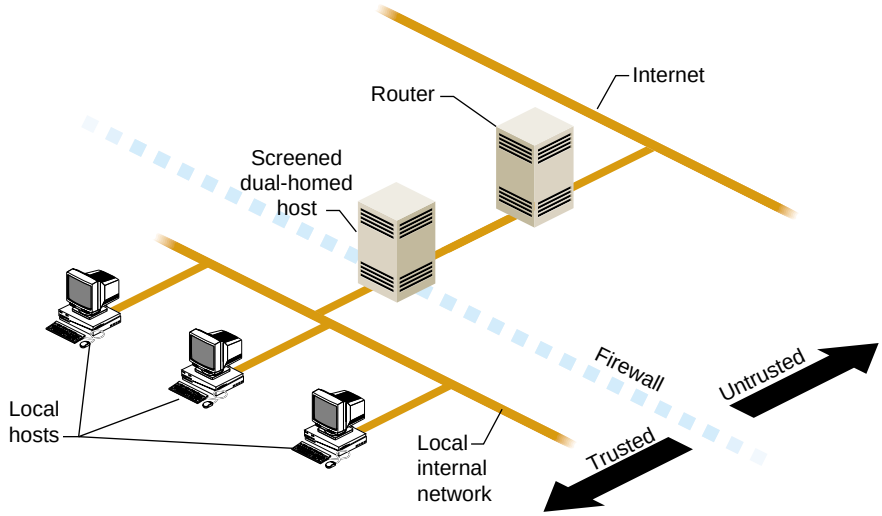

Figure 5-2 Screened Host

An additional level of complexity—and flexibility—is added when you expand the screened host scenario to a screened network scenario. The basic design remains the same, but the screened network receives all external traffic. Both the Internet and the internal network have access to the screened network, but traffic involving the internal network must still pass through the firewall host. This is useful for sites that want to make multiple servers available to the Internet and yet maintain a secure internal network. You could, for example, use one of the public hosts as your WWW server and another as an FTP server, depending on what you want to make available and the relative CPU loads expected.

Figure 5-3 on page 119 illustrates a screened subnet.[1]

[1] The "screened subnet" is sometimes called a "demilitarized zone" ("DMZ") or "red zone."

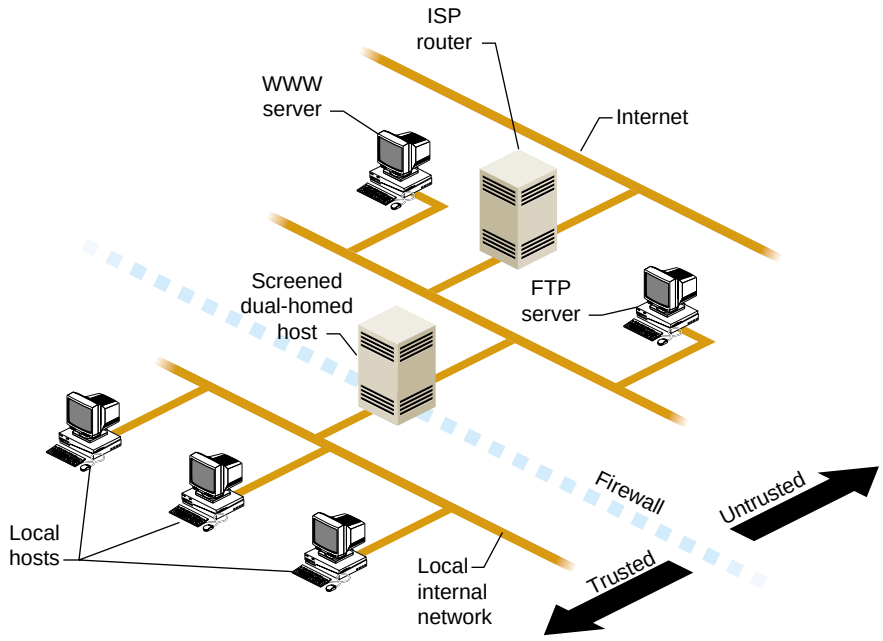

Figure 5-3 Screened Subnet

In the situation shown in Figure 5-3 on page 119, you continue to concentrate your security efforts on the single firewall host. Remember though, that your servers outside of the firewall are more easily compromised as they are protected only by a router. Keep your private data on the internal network and forward important data collected on the public servers to an internal host. (Details on software configuration are discussed in the next section.)

IRIX Configuration

This section discusses the basic network addressing configuration required on a firewall host, and then provides details on configuring IRIX software to tighten security on the host.

Note: Unless specified otherwise, all the software changes discussed in this section are to be performed on the firewall host.

Network Software Setup on a Dual-Homed Host

A dual-homed host is configured in network software as if it were two hosts, each with a different network address and, optionally, a different name. Use separate IP addresses for the two (or more) network interfaces (see "IRIX Admin: Networking and Mail").

Tightening Security in IRIX

This section discusses various modifications you can make to the IRIX operating system software to provide increased network security. Some of these changes are highly desirable on a firewall; others are more a matter of personal choice depending on the level of security you feel is necessary. The issues discussed include why the changes must or might be made.

The following discussion of changes made to the firewall host software also applies to any host made publicly accessible, such as the WWW server and FTP server shown in the screened subnet example in Figure 5-3 on page 119.

Note: Do not connect your hardware to the external network until you make the changes described in this section. When you have finished the procedures, reboot your firewall system to ensure that all changes take effect. *Many of these changes do not take effect until the system is rebooted*.

Disable Forwarding of IP Packets

By default, IRIX forwards IP packets on machines with more than one network hardware interface. You must edit a kernel configuration file, run *autoconfig*, and then reboot to disable this default.

Follow this procedure to turn off automatic IP packet forwarding:

1. As root, edit the file */var/sysgen/master.d/bsd*, changing the value of *ipforwarding* to 0:

   Change the line

   ```
   int ipforwarding = 1;
   ```

   to

   ```
   int ipforwarding = 0;
   ```

2. Save the modified */var/sysgen/master.d/bsd* file and exit from the editor.

3. Run *autoconfig* with the **-f** option:

   ```
   # autoconfig -f
   ```

   This creates a */unix.install* file, which becomes the new */unix* after the system is rebooted.

4. Reboot your system (see reboot(1M)).

5. To verify that IP packet forwarding has been disabled after your system comes back up, use the *netstat* command:

   ```
   # netstat -s -p ip | grep forwarding
   ```

   You should see the following:

   ```
    0 packets forwarded (forwarding disabled)
   ```

   If you do not see this message, repeat steps 1 through 5 until you do. (Be sure that your root filesystem has enough disk space so that the */unix.install* file is being created correctly. See autoconfig(1M) for more information.)

Limiting inetd Services

When your system starts up, the *inetd* process reads the */etc/inetd.conf* file for a list of TCP/IP services to support. Comment out services listed in this file that are not very secure or that you are not using.

Note: These services are being disabled on the firewall only. Services that are commented out in the system files on the firewall may still be available on your internal network—you just can't use them on the firewall host.

1. Edit the file */etc/inetd.conf*, and add the **#** symbol at the beginning of the following lines to comment them out (some may have already been commented out):

```
exec        stream  tcp     nowait  root    /usr/etc/rexecd         rexecd
bootp       dgram   udp     wait    root    /usr/etc/bootp          bootp
rstatd/1-3  dgram   rpc/udp wait    root    /usr/etc/rpc.rstatd     rstatd
walld/1     dgram   rpc/udp wait    root    /usr/etc/rpc.rwalld     rwalld
rusersd/1   dgram   rpc/udp wait    root    /usr/etc/rpc.rusersd    rusersd
rquotad/1   dgram   rpc/udp wait    root    /usr/etc/rpc.rquotad    rquotad
bootparam/1 dgram   rpc/udp wait    root    /usr/etc/rpc.bootparamd bootparam
ypupdated/1 stream  rpc/tcp wait    root    /usr/etc/rpc.ypupdated  ypupdated
rexd/1      stream  rpc/tcp wait    root    /usr/etc/rpc.rexd       rexd
```

   In other words, they should look like this:

```
#exec        stream  tcp     nowait  root    /usr/etc/rexecd         rexecd
```

```
#bootp       dgram   udp     wait    root    /usr/etc/bootp          bootp
#rstatd/1-3  dgram   rpc/udp wait    root    /usr/etc/rpc.rstatd     rstatd
#walld/1     dgram   rpc/udp wait    root    /usr/etc/rpc.rwalld     rwalld
#rusersd/1   dgram   rpc/udp wait    root    /usr/etc/rpc.rusersd    rusersd
#rquotad/1   dgram   rpc/udp wait    root    /usr/etc/rpc.rquotad    rquotad
#bootparam/1 dgram   rpc/udp wait    root    /usr/etc/rpc.bootparamd bootparam
#ypupdated/1 stream  rpc/tcp wait    root    /usr/etc/rpc.ypupdated  ypupdated
#rexd/1      stream  rpc/tcp wait    root    /usr/etc/rpc.rexd       rexd
```

If you want details on the services you are disabling, refer to their reference pages. For example, refer to rexecd(1M) for more information on the *rexecd* daemon.

2. Comment out or restrict the following entries in */etc/inetd.conf*:

```
ftp     stream  tcp     nowait  root    /usr/etc/ftpd   ftpd -la
telnet  stream  tcp     nowait  root    /usr/etc/telnetd        telnetd
shell   stream  tcp     nowait  root    /usr/etc/rshd           rshd
login   stream  tcp     nowait  root    /usr/etc/rlogind        rlogind
tftp    dgram   udp     wait    guest   /usr/etc/tftpd  tftpd -s \
/usr/local/boot /usr/etc/boot
```

If you comment them out (totally disable them), they should look like this:

```
#ftp     stream  tcp     nowait  root    /usr/etc/ftpd   ftpd -l
#telnet  stream  tcp     nowait  root    /usr/etc/telnetd        telnetd
#shell   stream  tcp     nowait  root    /usr/etc/rshd           rshd
#login   stream  tcp     nowait  root    /usr/etc/rlogind        rlogind
#tftp    dgram   udp     wait    guest   /usr/etc/tftpd  tftpd -s \
/usr/local/boot /usr/etc/boot
```

To be safe, it is best to disable all those services with the comment character as shown above. (Doing so means, however, that the host can only be accessed from the local console.) Of these services, enabling *rshd* is probably the most dangerous, and *tftpd* is almost never required on a firewall. Regarding *ftpd*, refer to "IRIX Admin: Networking and Mail." If, however, you must include any of these services, change them as indicated below so that they record a log of their use in the file */var/adm/SYSLOG*:

```
ftp     stream  tcp     nowait  root    /usr/etc/ftpd   ftpd -ll
shell   stream  tcp     nowait  root    /usr/etc/rshd           rshd -Lal
tftp    dgram   udp     wait    guest   /usr/etc/tftpd  tftpd -s -l -h /dev/null
```

Note the logging options added to each daemon invocation. (For more information, refer to the reference page for any daemon you modify.)

The *telnetd* and *rlogind* entries have not been included here because remote logins can (and should) be controlled with the use of one-time passwords. One-time passwords are just that—a password that can be used once to gain access, but any future use of that same password is disallowed. There are various ways to implement one-time passwords, and how (and if) you use them at your site depends on your need for remote login capability and the degree to which you want to authenticate such logins. Refer to the *Firewalls and Internet Security* book referenced in "Books" on page 23.

3. The *fingerd* service is also a potential security hole because it is a source of account names. You can use the **-S** option to suppress information about login status, home directory, and shell, which might be used to attack security:

```
finger  stream  tcp     nowait  guest   /usr/etc/fingerd        fingerd -S
```

   Or, to be more secure, you can configure *fingerd* with the **-f** option, to return just a message file. In the following example, a message has been placed in */etc/fingerd.message*:

```
finger  stream  tcp     nowait  guest   /usr/etc/fingerd        fingerd -f \
/etc/fingerd.message
```

   The contents of */etc/fingerd.message* might say something like:

```
Thank you for your interest in XYZ company. Please contact us at
xyz.email.address or 1-800-XYZ-PHON for more information.
```

   This message is then returned for any *finger* access.

4. When you have finished making changes to the */etc/inetd.conf* file, write the changes and exit from the editor. The changes take affect after a reboot. If you want to apply them immediately, enter:

```
# killall -HUP inetd
```

5. Test any modified services to be sure they perform as expected.

Password Protection

Limit the number of users with login accounts on the firewall system as much as possible. All accounts in */etc/passwd* should have a password (see passwd(1)).

Check to see if there are any */etc/hosts.equiv* or *$HOME/.rhosts* files. These files can be configured to allow remote access without password protection, and should not be allowed on a firewall host. Refer to hosts.equiv(4) for more information.

Refer to "Password Administration" on page 84 for details on host access password security.

Limiting rpc Services Access

You can limit access to the firewall host's RPC services by use of the *portmap* command's **-a** option. This allows you to specify the host(s) and/or network(s) that are allowed access to RPC-based services. Edit the file */etc/config/portmap.options* to add options to the *portmap* command that is executed at system startup.

For example, suppose you create a */etc/config/portmap.options* file with the following entries:

```
-a 192.0.2.0
-a 192.14.12.0
```

This restricts access to firewall host RPC services to hosts on the Class C networks 192.0.2 and 192.13.12.

The syntax for the **-a** option allows you to specify multiple network masks, network addresses, and host addresses. As usual, the fewer hosts or networks allowed access, the better the security. Refer to the reference page portmap(1M) for more information.

Disabling NIS (YP)

Because the nature of NIS (formerly called Yellow Pages) does not accommodate security needs, remove it from the firewall host:

1. Remove the NIS software from the firewall host with the *versions* command:

   ```
   # versions remove nfs.sw.nis
   ```

2. Certain databases may have been modified to add NIS information by including a + symbol in a database entry. Use an editor to remove any lines beginning with the + symbol from the files */etc/passwd*, */etc/group*, and */etc/aliases*.
3. Remove the */etc/netgroups* file if it exists.

Caution: You should not run NIS on a firewall. If you must run NIS, be sure the server is secure and have the clients run *ypbind* with the **-ypsetme** option which provides some minimal security.

Limiting NFS Access

Exporting filesystems and remote-mounting external systems on the firewall presents security problems. *You should not use NFS on a firewall*—remove it with *versions remove nfs.sw.nis*. If for some reason you cannot do this, you have a few (less secure) options:

- You can disallow NFS altogether, using this command:

  ```
  # chkconfig nfs off
  ```

- You can edit the */etc/exports* file to limit exported filesystem permissions and access. You can, for example, use the `rw=`*hostname* option to limit read-write access to a specific host, or you can use the `access=`*client* option to limit mounting to specified hosts. Refer to the reference page for exports(4) for more information.
- If you choose to mount external systems on the firewall host, use the *mount* command with the **nosuid** option to prevent running a Trojan horse. Refer to the fstab(4) reference page for details.

In general, you should mount all filesystems other than your system directories with the **nosuid**, **nodev** options (refer to mount(1M)).

Setting Up a Proper Log File

Log files provide useful information to the firewall administrator, recording specific or all attempts at firewall host login. The various options used to turn on logging for different daemons have been covered in the discussions on each daemon. Note that the log files must be reviewed periodically to be of use.

Log files are sensitive information and are best not stored on the firewall host. Refer to syslogd(1M) for information on how to forward *syslog* messages from the firewall host to a trusted host inside the firewall.

Checking Software Integrity

All software on a firewall host should be watched for modification. A record of checksums of software should be kept and compared periodically to detect unauthorized changes. For this reason too, the less software installed on the firewall host the better. You should always be on the watch for such things as device files outside of */dev*, and files with SUID and GUID permissions set.

You can use the *versions* command to display a list of system files modified since installation. For example:

```
# versions changed
Configuration Files

m     = modified since initial installation
?     = modification unknown
blank = file is as originally installed

  /etc/init.d/netsite
m /etc/init.d/netsite.O
m /etc/init.d/netsite.N
m /etc/uucp/Devices
  /etc/uucp/Devices.N
m /var/X11/xdm/Xsession.dt
  /var/X11/xdm/Xsession.dt.O
  /var/X11/xdm/xdm-config
<etc>
```

You can also use the *versions -m* command to list only modified installed files. Refer to versions(1M) for more information.

Educating Users

You can take great pains to make a secure firewall and then have security compromised by users ignorant of the consequences of their actions. If possible, do not allow user accounts on the firewall host. If you do allow user accounts, be sure to tell the account holders:

- Don't use *.rhosts* files. (As the superuser, you can add the **-l** option to the *rshd* invocation in */etc/inetd.conf* and thereby disallow these files on the firewall. See rshd(1M) for more information.)
- Use passwords with long, non-dictionary, ASCII strings, change them frequently, and don't write them down!
- Don't use the *"xhost +"* command. (As superuser, you can delete the binary, or limit its execution to the superuser as well).

Even your supposedly protected internal network can be compromised by inappropriate actions of users. If, for example, a user on an internal host attaches a modem and establishes a PPP or SLIP session with an external site, you now have a situation in which the external world has two connections to your internal network—one through the firewall, but the other directly to a non-secure, internal host.

Internal Network Configuration

While it is beyond the scope of this section to describe how to configure your internal network, this section discusses issues of DNS and Sendmail configuration that relate specifically to firewall security. Refer to "IRIX Admin: Networking and Mail" for information on basic Sendmail and DNS setup.

Domain Name System (DNS)

DNS, the name service used on the Internet, should be configured for your site to give out the addresses that other sites need to contact you. This might include the address of your router, your firewall host, and any other machines you want others to be able to communicate with. In the case of a simple firewall comprised of a dual-homed host, the dual-homed host would be a DNS server, providing the address of the Internet side of its network connection. In the case of a screened subnet, the DNS server could be any of the "public" hosts in the subnet, and it could provide addresses for all of these hosts and the router.

You should also set up the DNS Mail eXchanger (MX) record to advertise the name of the host(s) responsible for mail at your site. This may be the firewall host or another host.

Do not publish internal hostnames and addresses on the firewall host. If you have a single firewall host performing multiple services, say FTP and WWW serving, use CNAME records to "alias" the services to the hostname. This makes it easy to move these services to different hosts if you want to separate them later.

Mail Configuration

This section presents some suggestions for limiting the susceptibility of your site to an attack through the electronic mail system. Internet electronic mail is based on the Simple Mail Transfer Protocol, or SMTP. The program that implements SMTP is commonly referred to as *sendmail*. *sendmail* is a large and complicated program that is frequently the subject of attack.

Sendmail Configuration and Mail Aliases

Your mail system should be configured cooperatively with your DNS configuration. That is, whichever machine your DNS server is advertising as your Mail eXchanger (MX) host

must have its *sendmail* configured to accept mail for your network, and to do the appropriate thing with it once it is received. Usually that means to forward the mail to a master mail machine on the internal network, which knows users' internal addresses and how to deliver the mail to them.

A note about current convention: It is popular to use the domain name of your network as your electronic mail address. For example, user "harry" at company XYZ corporation, whose domain name is XYZ.com would have the electronic mail address "harry@XYZ.com," as opposed to "harry@machine1.XYZ.com." Edit the */etc/sendmail.cf* file to do this (see "IRIX Admin: Networking and Mail").

To reinforce the electronic mail address of your site, and to make it easy for others to reply to your users' mail, it is recommended that you configure your *sendmail* to rewrite all your addresses to conform to the above convention.

For details on how to configure *sendmail.cf*, refer to "IRIX Admin: Networking and Mail."

Spool Isolation

If a barrage of email is sent to your firewall host, it can fill up the disk and paralyze further operation. If you are concerned about this possibility, isolate the mail spool by putting it on a disk or disk partition of its own. While this does not prevent email from being overwhelmed, it does keep a crucial system disk partition, such as */usr*, from filling up.

Using Proxy Servers

A proxy server is an application that implements security for a particular network service. It is basically an application-level gateway that, by "understanding" the particular application protocol, is able to transparently intercept traffic and so implement protocol-specific security, logging, authentication, and so on.

Proxy servers provided on the firewall can allow, for example, internal users to use Netscape Navigator™ to access the World Wide Web, to use *ftp* to transfer files between a host on the internal network and one on the Internet, or to *telnet* to an external host for an interactive session.

The two most common proxy server solutions are server-side proxies and the SOCKS proxy server. The proxy servers available with the optional Gauntlet™ for IRIX firewall implement server-side-only applications, in which one proxy server exists for each

supported application. The SOCKS approach utilizes a *socksd* process on the server, and then requires any application that communicates with it to be "SOCKSified" that is; compiled with the SOCKS library. The Netscape Navigator™, for example, comes already "SOCKSified."

Refer to "Additional Resources" on page xxi for information on creating your own proxy support, or contact your Silicon Graphics sales representative for information on Gauntlet for IRIX and the Netscape Proxy Server.

PART THREE

Accounting

Part III, *Accounting*, contains the following chapters:

Chapter 6
Administering the System Audit Trail

Chapter 7
System Accounting

Chapter 6

Administering the System Audit Trail

The System Audit Trail features allow 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:

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*, *chlabel* and *newlabel* commands. To find out if your system is running Trusted IRIX/B, use the *uname* command with the **-a** option. Standard IRIX systems 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 Inst to install the *eoe2.sw.audit* software package from your distribution media. *inst* is described in detail in the *Software Installation Guide*. Once this package has been installed, reboot your system and use the *chkconfig* utility to enable auditing. The chkconfig(1M) 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
```

The following command enables auditing on your system:

```
chkconfig audit on
```

The system immediately begins 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 list contains 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 in “Auditable Events” on page 137.)

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* and *satconfig* 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. These utilities are discussed in detail in “Using satconfig” on page 141 and “Using sat_select” on page 142.

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* 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 and logout (sat_ae_identity)

  Any login attempt, whether successful or not, should be audited. Also, an audit record should be generated when the user logs out of the system.

- su (sat_check_priv, sat_ae_identity)

  Whenever a user invokes the *su* 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* command to change the DAC permissions on a file or the *chown* 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*.

sat_chroot
: Current root directory was changed with *chroot*.

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*. 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*.

sat_mount
: A filesystem was mounted or unmounted.

sat_file_attr_read
: The attributes of a file were read by *stat*.

sat_file_attr_write
: The attributes of a file were written by *chmod*.

sat_exec
: A new process has been introduced by *exec*.

sat_sysacct
: System accounting has been turned on or off.

sat_fchdir
: The user changed from the 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*.

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*.

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*.

sat_proc_write
: The user finalized a changes to a process's address space using *ptrace*.

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 hostname 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* 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*, *bind*, or *connect* 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 outside 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- or logout- related event occurred.

sat_ae_dbedit
: A file was modified using the *dbedit* utility. (This utility is available only with the Trusted IRIX/B optional product.)

sat_ae_mount
: An NFS filesystem 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 superuser 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 143.

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* 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 are 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 are 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 so, use this command:

```
sat_select -out > /etc/config/sat_select.options
```

Then, to restore auditing to the saved state, use this 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 pathname. 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 filesystem 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* 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* 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 the audit daemon, see the audit(1M), satd(1M), and audit_filters(5) reference pages and the comments in */etc/init.d/audit*.

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** *user_name*

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:

sat_reduce -u *user_name 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*.

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 (MAC) 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.

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 filesystem 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* 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
```

```
/usr/!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(1M) reference page.

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, are descriptions of 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, enter 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 get 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** 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*. 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 should be closely accounted by the auditor.

`sat_reduce -n` *interesting_file* **`-e sat_open -e sat_open_ro`** *sat_filename* **`| 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*, 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 record 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* utility to reduce the size of your old audit files by up to 80 percent.

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 filesystem containing your audit trail is more than 90 percent full, you should remove old audit files. If your audit files are kept in */var/adm/sat*, enter 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 percent 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 percent free space in your audit filesystem 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 percent. 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 is 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 percent of the filesystem is free, you may boot into multiuser mode without difficulty.

If your auditing system directs the audit files to the / (root) filesystem or the */usr* file system and either filesystem 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 filesystems are mounted on the Inst filesystem.

3. Once you have created free disk space on your / (root) and */usr* filesystems, 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.

Chapter 7

System Accounting

This chapter contains the following sections;

- "Process (System) Accounting" on page 157 describes how to use the accounting utilities to keep track of system usage.
- "Additional Resources" on page 174 provides information on additional accounting software.

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 159.
- 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 "IRIX Admin: Disks and Filesystems." 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 160.

Turning Off Process Accounting

To turn off process accounting, follow these steps:

1. Log in as *root*.
2. Enter this command:

   chkconfig 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, and so on. 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 starts.

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* complains 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 is charged to the login that is first in the password file for that user ID.

Fixing tacct Errors

If the installation is using the accounting system to charge users for system resources, the integrity of *sum/tacct* is quite important. Occasionally, mysterious *tacct* records appear with negative numbers, duplicate user IDs, or a user ID of 65,535. First check *sum/tacctprev* with *prtacct*. If it looks all right, the latest *sum/tacct.MMDD* should be patched up, then *sum/tacct* recreated. A simple patchup procedure would be:

1. Enter the command:

   cd /var/adm/acct/sum
2. Enter the command:

   acctmerg -v < tacct.MMDD > xtacct
3. Enter the command:

   ed xtacct
4. Remove the bad records.
5. Write duplicate UID records to another file.

6. Enter the command:

   ```
   acctmerg -i < xtacc t > tacct.MMDD
   ```

7. Enter 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. This period runs from 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* 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* 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* 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** and **write** 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?*

Additional Resources

Ask your Silicon Graphics sales representative for information on additional tools available. For example, SHARE II™ for IRIX is an optional product allowing additional administrative control of system resources including disk space, CPU entitlement, memory (real or virtual), number of processes, printer pages, terminal and modem connect-time, network packets, and more.

Index

C

D

E

F

O

P

R

S

T

U

V

W

X

Tell Us About This Manual

As a user of Silicon Graphics products, you can 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:

- 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 send the title and part number of the document with your comments. The part number for this document is 007-2862-001.

Thank you!

Three Ways to Reach Us

- To send your comments by **electronic mail**, use either of these addresses:
  - On the Internet: techpubs@sgi.com
  - For UUCP mail (through any backbone site): *[your_site]*!sgi!techpubs
- To **fax** your comments (or annotated copies of manual pages), use this fax number: 650-932-0801
- To send your comments by **traditional mail**, use this address:

  Technical Publications
  Silicon Graphics, Inc.
  2011 North Shoreline Boulevard, M/S 535
  Mountain View, California 94043-1389