System Administration Guide: Devices and File Systems

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 817–6960–11
September 2004

Adobe PostScript

040727@9495

Contents

Preface

System Administration Guide: Devices and File Systems is part of a set that includes a significant part of the Solaris™ system administration information. This guide contains information for both SPARC® based and x86 based systems.

This book assumes you have completed the following tasks:

- Installed the SunOS 5.9 operating system
- Set up all the networking software that you plan to use

The SunOS 5.9 operating system is part of the Solaris product family, which also includes many features, including the Solaris Common Desktop Environment (CDE). The SunOS 5.9 operating system is compliant with AT&T's System V, Release 4 operating system.

For the Solaris 9 release, new features interesting to system administrators are covered in sections called *What's New in ... ?* in the appropriate chapters.

Note – The Solaris operating system runs on two types of hardware, or platforms, SPARC and x86. The Solaris operating system runs on both 64–bit and 32–bit address spaces. The information in this document pertains to both platforms and address spaces unless called out in a special chapter, section, note, bullet, figure, table, example, or code example.

Note – Sun is not responsible for the availability of third-party Web sites mentioned in this document. Sun does not endorse and is not responsible or liable for any content, advertising, products, or other materials that are available on or through such sites or resources. Sun will not be responsible or liable for any actual or alleged damage or loss caused by or in connection with the use of or reliance on any such content, goods, or services that are available on or through such sites or resources.

Who Should Use This Book

This book is intended for anyone responsible for administering one or more systems running the Solaris 9 release. To use this book, you should have 1-2 years of UNIX® system administration experience. Attending UNIX system administration training courses might be helpful.

How the System Administration Volumes Are Organized

Here is a list of the topics that are covered by the volumes of the System Administration Guides.

Book Title	**Topics**
System Administration Guide: Basic Administration	User accounts and groups, server and client support, shutting down and booting a system, and managing software (packages and patches)
System Administration Guide: Advanced Administration	Printing services, terminals and modems, system resources (disk quotas, accounting, and crontabs), system processes, and troubleshooting Solaris software problems
System Administration Guide: Devices and File Systems	Removable media, disks and devices, file systems, and backing up and restoring data
System Administration Guide: IP Services	TCP/IP networks, IPv4 and IPv6, DHCP, IP Security, Mobile IP, IP Network Multipathing, and IPQoS
System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)	DNS, NIS, and LDAP naming and directory services
System Administration Guide: Naming and Directory Services (FNS and NIS+)	NIS+ naming and directory services
System Administration Guide: Security Services	Auditing, device management, file security, BART, PAM, Solaris cryptographic framework, privileges, RBAC, SASL, Solaris Secure Shell, and SEAM
System Administration Guide: Resource Management and Network Services	Resource management, remote file systems, mail, SLP, and PPP

Accessing Sun Documentation Online

The docs.sun.com^{SM} Web site enables you to access Sun technical documentation online. You can browse the docs.sun.com archive or search for a specific book title or subject. The URL is http://docs.sun.com.

What Typographic Conventions Mean

The following table describes the typographic conventions used in this book.

TABLE P–1 Typographic Conventions

Typeface or Symbol	Meaning	Example
`AaBbCc123`	The names of commands, files, and directories; on-screen computer output	Edit your `.login` file. Use `ls -a` to list all files. `machine_name% you have mail.`

`AaBbCc123`	What you type, contrasted with on-screen computer output	`machine_name%` **`su`**
		`Password:`

AaBbCc123	Command-line placeholder: replace with a real name or value	To delete a file, type **`rm`** *filename*.
AaBbCc123	Book titles, new words or terms, or words to be emphasized.	Read Chapter 6 in *User's Guide*. These are called *class* options. Do *not* save changes yet.

Shell Prompts in Command Examples

The following table shows the default system prompt and superuser prompt for the C shell, Bourne shell, and Korn shell.

TABLE P–2 Shell Prompts

Shell	Prompt
C shell prompt	`machine_name%`
C shell superuser prompt	`machine_name#`
Bourne shell and Korn shell prompt	`$`
Bourne shell and Korn shell superuser prompt	`#`

General Conventions

Be aware of the following conventions used in this book.

- When following steps or using examples, be sure to type double-quotes (`"`), left single-quotes (`‘`), and right single-quotes (`’`) exactly as shown.
- The key referred to as Return is labeled Enter on some keyboards.
- The root path usually includes the `/sbin`, `/usr/sbin`, `/usr/bin`, and `/etc` directories, so the steps in this book show the commands in these directories without absolute path names. Steps that use commands in other, less common, directories show the absolute paths in the examples.
- The examples in this book are for a basic SunOS software installation without the Binary Compatibility Package installed and without `/usr/ucb` in the path.

Caution – If `/usr/ucb` is included in a search path, it should always be at the end of the search path. Commands like `ps` or `df` are duplicated in `/usr/ucb` with different formats and options from the SunOS commands.

CHAPTER 1

Managing Removable Media (Overview)

This chapter provides general guidelines for managing removable media in the Solaris environment.

This is a list of the overview information in this chapter.

Where to Find Managing Removable Media Tasks

Use these references to find step-by-step instructions for managing removable media.

Removable Media Management Task	For More Information
Access removable media	Chapter 2
Format removable media	Chapter 3
Write data and music CDs	Chapter 4

For information on using removable media with File Manager in the Common Desktop Environment, see *Solaris Common Desktop Environment: User's Guide*.

Removable Media Features and Benefits

The Solaris environment gives users and software developers a standard interface for dealing with removable media. Referred to as volume management, this interface provides three major benefits:

- By automatically mounting removable media, it simplifies their use. (For a comparison between manual and automatic mounting, see the following section.)
- It enables you to access removable media without having to become superuser.
- It allows you to give other systems on the network automatic access to any removable media on your local system. For more information, see Chapter 2.

Comparison of Automatic and Manual Mounting

The following table compares the steps involved in manual mounting (without volume management) and automatic mounting (with volume management) of removable media.

TABLE 1–1 Comparison of Manual and Automatic Mounting

Steps	Manual Mounting	Automatic Mounting
1	Insert media.	Insert media.
2	Become superuser.	For diskettes, use the `volcheck` command.
3	Determine the location of the media device.	Volume manager (`vold`) automatically performs many of the tasks previously required to manually mount and work with removable media.
4	Create a mount point.	
5	Make sure you are not in the mount point directory.	
6	Mount the device using the proper `mount` options.	
7	Exit the superuser account.	

TABLE 1–1 Comparison of Manual and Automatic Mounting *(Continued)*

Steps	Manual Mounting	Automatic Mounting
8	Work with files on media.	Work with files on media.
9	Become superuser.	
10	Unmount the media device.	
11	Eject media.	Eject media.
12	Exit the superuser account.	

What You Can Do With Volume Management

Essentially, volume management enables you to access removable media just as manual mounting does, but more easily and without the need for superuser access. To make removable media easier to work with, you can mount removable media in easy-to-remember locations.

TABLE 1–2 How to Access Data on Removable Media Managed by Volume Manager

Access	Insert	Find the Files Here
Files on the first diskette	The diskette and enter `volcheck`	`/floppy`
Files on the first removable hard disk	The removable hard disk and enter `volcheck`	`/rmdisk/jaz0` or `/rmdisk/zip0`
Files on the first CD	The CD and wait for a few seconds	`/cdrom/`*volume-name*
Files on the first DVD	The DVD and wait for a few seconds	`/dvd/`*volume-name*
Files on the first PCMCIA	The PCMCIA and wait for a few seconds	`/pcmem/pcmem0`

If your system has more than one type of removable device, see the following table for their access points.

TABLE 1–3 Where to Access Removable Media

Media Device	Access File Systems With This Path	Access Raw Data With This Path
First diskette drive	`/floppy/floppy0`	`/vol/dev/aliases/floppy0`
Second diskette drive	`/floppy/floppy1`	`/vol/dev/aliases/floppy1`
First CD-ROM drive	`/cdrom/cdrom0`	`/vol/dev/aliases/cdrom0`
Second CD-ROM drive	`/cdrom/cdrom1`	`/vol/dev/aliases/cdrom1`
First removable hard disk	`/rmdisk/jaz0`	`/vol/dev/aliases/jaz0`
	`/rmdisk/zip0`	`/vol/dev/aliases/zip0`
First PCMCIA drive	`/pcmem/pcmem0`	`/vol/dev/aliases/pcmem0`

CHAPTER 2

Accessing Removable Media (Tasks)

This chapter describes how to access removable media from the command line in the Solaris environment.

For information on the procedures associated with accessing removable media, see the following:

- "Accessing Removable Media (Task Map)" on page 27
- "Accessing Removable Media on a Remote System (Task Map)" on page 36

For background information on removable media, see Chapter 1.

Accessing Removable Media (Task Map)

Task	Description	For Instructions
1. (Optional) Add the removable media drive	Add the removable media drive to your system, if necessary.	"How to Add a New Removable Media Drive" on page 30
2. (Optional) Decide whether you want to use removable media with or without volume management (`vold`)	Volume management (`vold`) runs by default. Decide whether you want to use removable media with or without volume management.	"Stopping and Starting Volume Management (`vold`)" on page 31
3. Access removable media	Access different kinds of removable media with or without volume management running.	"How to Access Information on Removable Media" on page 32

Task	Description	For Instructions
4. (Optional) Copy files or directories	Copy files or directories from the media as you would from any other location in the file system.	"How to Copy Information From Removable Media" on page 33
5. (Optional) Configure a system to play musical CDs or DVDs	You can configure a system to play musical CDs or DVDs, but you will need third-party software to play the media.	"How to Play a Musical CD or DVD" on page 33
6. Find out if the media still in use	Before ejecting the media, find out if it is still in use.	"How to Find Out If Removable Media Is Still in Use" on page 34
7. Eject the Media	When you finish, eject the media from the drive.	"How to Eject Removable Media" on page 35

Accessing Removable Media (Overview)

You can access information on removable media with or without using volume manager. For information on accessing information on removable media with CDE's File Manager, see "Using Removable Media with File Manager" in *Solaris Common Desktop Environment: User's Guide*.

Starting in the Solaris 8 6/00 release, volume manager (`vold`) actively manages all removable media devices. This means any attempt to access removable media with device names such as `/dev/rdsk/`*cntndnsn* or `/dev/dsk/`*cntndnsn* will be unsuccessful.

Using Removable Media Names

You can access all removable media with different names. The following table describes the different media names that can be accessed with or without volume management.

TABLE 2–1 Removable Media Names

Media	Volume Management Device Name	Volume Management Device Alias Name	Device Name

First diskette drive	`/floppy`	`/vol/dev/aliases/floppy0`	`/dev/rdiskette`
			`/vol/dev/rdiskette0/`
			volume-name

First, second, third CD-ROM or DVD-ROM drives	`/cdrom0`
	`/cdrom1`
	`/cdrom2`
		`/vol/dev/aliases/cdrom0`
		`/vol/dev/aliases/cdrom1`
		`/vol/dev/aliases/cdrom2`
			`/vol/dev/rdsk/c`*n*`t`*n*`[d`*n*`]/`
			volume-name

First, second, third Jaz drive	`/rmdisk/jaz0`
	`/rmdisk/jaz1`
	`/rmdisk/jaz2`
		`/vol/dev/aliases/jaz0`
		`/vol/dev/aliases/jaz1`
		`/vol/dev/aliases/jaz2`
			`/vol/dev/rdsk/c`*n*`t`*n*`d`*n*`/`
			volume-name

First, second, third Zip drive	`/rmdisk/zip0`
	`/rmdisk/zip1`
	`/rmdisk/zip2`
		`/vol/dev/aliases/zip0`
		`/vol/dev/aliases/zip1`
		`/vol/dev/aliases/zip2`
			`/vol/dev/rdsk/c`*n*`t`*n*`d`*n*`/`
			volume-name

First, second, third, PCMCIA drive	`/pcmem/pcmem0`
	`/pcmem/pcmem1`
	`/pcmem/pcmem2`
		`/vol/dev/aliases/pcmem0`
		`/vol/dev/aliases/pcmem1`
		`/vol/dev/aliases/pcmem2`
			`/vol/dev/rdsk/c`*n*`t`*n*`d`*n*`/`
			volume-name

Use this table to identify which removable media name to use with specific Solaris commands.

Solaris Command	Device Name	Usage Examples

`ls`, `more`, `vi`	`/floppy`
	`/cdrom`
	`/rmdisk/zip0`
	`/rmdisk/jaz0`
	`/pcmem/pcmem0`
		`ls /floppy/myfiles/`
		`more /cdrom/myfiles/filea`

`fsck`, `newfs`, `mkfs`	`/vol/dev/aliases/floppy0` `/vol/dev/rdsk/c`*n*`t`*n*`d`*n*	`newfs /vol/dev/aliases/floppy0` `mkfs -F udfs /vol/dev/rdsk/c`*n*`t`*n*`d`*n*

Guidelines for Accessing Removable Media Data

Most CDs and DVDs are formatted to the ISO 9660 standard, which is portable, so most CDs and DVDs can be mounted by volume management. However, CDs or DVDs with UFS file systems are not portable between architectures, so they must be used on the architecture for which they were designed.

For example, a CD or DVD with a UFS file system for a SPARC platform cannot be recognized by an x86 platform. Likewise, an x86 UFS CD cannot be mounted by volume management on a SPARC platform. The same limitation applies to diskettes. (Actually, some architectures share the same bit structure, so occasionally a UFS format specific to one architecture will be recognized by another architecture, but the UFS file system structure was not designed to guarantee this compatibility).

To accommodate the different formats, the CD or DVD is split into slices, which are similar in effect to partitions on hard disks. The 9660 portion is portable, but the UFS portion is architecture-specific. If you are having trouble mounting a CD or DVD, particularly if it is an installation CD or DVD, make sure its UFS file system is appropriate for your system's architecture (check the label on the CD or DVD).

Accessing Jaz Drives or Zip Drives

You can determine whether accessing your Jaz or Zip drives changes from previous Solaris releases, depending on the following:

- If you are upgrading from the Solaris 8 6/00 release to the Solaris 9 release, you can continue to access your Jaz drives and Zip drives in the same way as in previous releases.
- If you are freshly installing the Solaris 9 release, you cannot access your Jaz drives and Zip drives in the same way as in previous Solaris releases.

  Follow these steps if you want to access your Jaz and Zip drives in the same way as in previous Solaris releases:

  1. Comment the following line in the `/etc/vold.conf` file by inserting a pound (#) sign at the beginning of the text, like this:

     ```
     # use rmdisk drive /dev/rdsk/c*s2 dev_rmdisk.so rmdisk%d
     ```
  2. Reboot the system.

▼ How to Add a New Removable Media Drive

Adding a new removable media drive involves creating the `/reconfigure` file and rebooting the system so that volume management recognizes the new media drive.

Steps **1. Become superuser.**

2. **Create the `/reconfigure` file.**

   ```
   # touch /reconfigure
   ```

3. **Bring the system to run level 0.**

   ```
   # init 0
   ```

4. **Turn off power to the system.**
5. **Connect the new media drive.**

   See your hardware handbook for specific instructions.

6. **Turn on power to the system.**

   The system comes up to multiuser mode automatically.

Stopping and Starting Volume Management (`vold`)

Occasionally, you might want to manage media without the help of volume management. This section describes how to stop and restart volume management.

▼ How to Stop Volume Management (`vold`)

Steps

1. **Make sure media is not being used.**

   If you are not sure whether you have found all users of the media, use the `fuser` command, as described in "How to Find Out If Removable Media Is Still in Use" on page 34.

2. **Become superuser.**
3. **Enter the `volmgt stop` command.**

   ```
   # /etc/init.d/volmgt stop
   #
   ```

▼ How to Restart Volume Management (`vold`)

Steps

1. **Become superuser.**
2. **Enter the `volmgt start` command.**

   ```
   # /etc/init.d/volmgt start
   volume management starting.
   ```

▼ How to Access Information on Removable Media

Steps

1. **Insert the media.**

   The media is mounted after a few seconds.

2. **Check for media in the drive.**

   ```
   % volcheck
   ```

   Use the appropriate device name to access information by using the command-line interface. See Table 2–1 for an explanation of device names.

3. **List the contents of the media.**

   ```
   % ls /media
   ```

Example 2–1 Accessing Information on Removable Media

Access information on a diskette as follows:

```
$ volcheck
$ ls /floppy
myfile
```

Access information on a Jaz drive as follows:

```
$ volcheck
$ ls /rmdisk
jaz0/        jaz1/
```

Access information on a CD-ROM as follows:

```
$ volcheck
$ ls /cdrom
cdrom0@        solaris_9_sparc
```

View the symbolic links on a CD-ROM as follows:

```
$ ls -lL /cdrom/cdrom0
total 24
dr-xr-xr-x   2 root     sys         2048 Dec  3 11:54 s0/
drwxr-xr-x  18 root     root         512 Dec  3 13:09 s1/
drwxr-xr-x   2 root     root         512 Dec  3 13:10 s2/
drwxr-xr-x   2 root     root         512 Dec  3 13:10 s3/
drwxr-xr-x   2 root     root         512 Dec  3 13:10 s4/
drwxr-xr-x   2 root     root         512 Dec  3 13:10 s5/
```

Access information on a PCMCIA memory card as follows

```
$ ls /pcmem/pcmem0
pcmem0 myfiles
```

▼ How to Copy Information From Removable Media

You can access files and directories on removable media just like any other file system. The only significant restrictions are ownership and permissions.

For instance, if you copy a file from a CD into your file system, you'll be the owner, but you won't have write permissions (because the file never had them on the CD). You'll have to change the permissions yourself.

Steps

1. **Make sure the media is mounted.**

   ```
   $ ls /media
   ```

   The `ls` command displays the contents of a mounted media. If no contents are displayed, see "How to Access Information on Removable Media" on page 32.

2. **(Optional) Copy the files or directories.**

   For example, for a CD, you would do the following:

   ```
   $ cp /cdrom/sol_9_1202_sparc/s0/Solaris_9/Tools/add_install_client .
   $ ls -l
   -rwxr-xr-x   1 pmorph    gelfs    59586 Jan 16  2004 add_install_client*
   ```

   For example, for a PCMCIA memory card, you would do the following:

   ```
   $ cp /pcmem/pcmem0/readme2.doc .
   $ cp -r /pcmem/pcmem0/morefiles .
   ```

▼ How to Play a Musical CD or DVD

To play musical media from a media drive attached to a system running the Solaris release, you'll need to access public domain software, such as `xmcd`, that is available from the following locations:

- http://www.ibiblio.org/tkan/xmcd

  This site includes frequent updates to the `xmcd` software, which includes the version of `xmcd` that plays on newer Sun hardware, such as the Sun Blade™ systems.
- http://www.sun.com/software/solaris/freeware/pkgs_download.html

Keep the following in mind when using the `xmcd` software with CDDA (CD Digital Audio) support to play musical media:

- Use `xmcd`, version 3.1 (or later) on Sun Blade systems because this version has CDDA support, which must be enabled in order to listen to CDs on these systems.
- Enable CDDA by launching `xmcd`, clicking on the options button (it has a hammer and screwdriver on the button), and then by clicking on "CDDA playback".
- When CDDA is enabled, audio is directed to the audio device, so headphones and external speakers should be connected to the audio device and not to the media drive itself.

- CDDA can be enabled on other machines too. Enabling CDDA is required for playing media on the Sun Blade systems.

Consider the following issues as well:

- If you are using xmcd with standard playback on a system that *does not* have an internal connection from the CD-ROM to the audio device, you must insert headphones into the CD-ROM drive's headphone port.
- If you are using xmcd with standard playback on a system that *does* have an internal connection from the CD-ROM to the audio device, you can do either of the following:

  1. Insert headphones into the headphone port of the CD-ROM drive.
  2. Insert headphones into the headphone port on the audio device.

  If you choose #2, you must do the following:

  - Select the internal CD as the input device.
  - Make sure that Monitor Volume is non-zero.

  You can do both of these from sdtaudiocontrol's record panel.

Once you install the xmcd software, you can play a musical CD simply by inserting it into the CD-ROM drive and starting the xmcd control panel.

Steps

1. **Install the xmcd software.**
2. **Insert the media into the media drive.**
3. **Invoke the xmcd command.**

```
% ./xmcd &
```

▼ How to Find Out If Removable Media Is Still in Use

Steps

1. **Become superuser.**
2. **Identify the processes accessing the media.**

```
# fuser -u [-k] /media
```

-u	Displays the user of the media.
-k	Kills the process accessing the media.

For more information on using the fuser command, see fuser(1M).

3. **(Optional) Kill the process accessing the media.**

```
# fuser -u -k /media
```

Caution – Killing the process accessing the media should only be used in emergency situations.

4. **Verify the process is gone.**

```
# pgrep process-ID
```

Example 2–2 Finding Out If the Media Is Still in Use

The following example shows that the process `26230c`, owner `pmorph`, is accessing the `/cdrom/sol_9_1202_sparc/s0/Solaris_9/Tools/` directory.

```
# fuser -u /cdrom/sol_9_1202_sparc/s0/Solaris_9/Tools/
/cdrom/sol_9_1202_sparc/s0/Solaris_9/Tools/:     7258c(pmorph)
```

▼ How to Eject Removable Media

Steps

1. **Make sure the media is not being used.**

   Remember, media is "being used" if a shell or an application is accessing any of its files or directories. If you are not sure whether you have found all users of a CD (a shell hidden behind a desktop tool might be accessing it), use the `fuser` command, as described in "How to Find Out If Removable Media Is Still in Use" on page 34.

2. **Eject the media.**

   ```
   # eject media
   ```

   For example, for a CD, you would do the following

   ```
   # eject cdrom
   ```

   For example, for a PCMCIA memory card, you would do the following:

   ```
   # eject pcmem0
   ```

Accessing Removable Media on a Remote System (Task Map)

The following table describes the tasks need to access removable media on a remote system.

Task	Description	For Instructions
1. Make local media available to remote systems	Add the removable media drive to your system, if necessary.	"How to Make Local Media Available to Other Systems" on page 36
2. Access removable media on remote systems	Insert the media into the drive.	"How to Access Information on Removable Media" on page 32

▼ How to Make Local Media Available to Other Systems

You can configure your system to share its media drives to make any media in those drives available to other systems. (This does not apply to musical CDs.) Once your media drives are shared, other systems can access the media they contain simply by mounting them, as described in "How to Access Removable Media on Remote Systems" on page 39.

Steps

1. **Become superuser.**

2. **Find out whether the NFS daemon (`nfsd`) is running.**

```
# ps -ef | grep nfsd
root 14533    1 17 10:46:55 ?      0:00 /usr/lib/nfs/nfsd -a 16
root 14656  289  7 14:06:02 pts/3 0:00 grep nfsd
```

   If the daemon is running, a line for `/usr/lib/nfs/nfsd` will appear, as shown above. If the daemon is not running, only the `grep nfsd` line will appear.

3. **Identify the `nfsd` status and select one of the following:**

   a. **If `nfsd` is running, go to Step 8.**

   b. **If `nfsd` is *not* running, continue with Step 4.**

4. **Create a dummy directory for `nfsd` to share.**

```
# mkdir / dummy-dir
```

The *dummy-dir* mount point can be any directory name. For example, *dummy*. This directory will not contain any files. Its only purpose is to "wake up" the NFS daemon so that it notices your shared media drive.

5. **Add the following entry into the `/etc/dfs/dfstab` file.**

   ```
   share -F nfs -o ro [-d comment] /dummy-dir
   ```

   When you start the NFS daemon, it will see this entry, "wake up," and notice the shared media drive. Note that the comment (preceded by `-d`) is optional.

6. **Start the NFS daemon.**

   ```
   # /etc/init.d/nfs.server start
   ```

7. **Verify that the NFS daemon is indeed running.**

   ```
   # ps -ef | grep nfsd
   root 14533    1 17 10:46:55 ?     0:00 /usr/lib/nfs/nfsd -a 16
   root 14656  289  7 14:06:02 pts/3 0:00 /grep nfsd
   ```

8. **Eject any media currently in the drive.**

   ```
   # eject media
   ```

9. **Assign root write permissions to the `/etc/rmmount.conf` file.**

   ```
   # chmod 644 /etc/rmmount.conf
   ```

10. **Add the following lines to the `/etc/rmmount.conf` file.**

    ```
    # File System Sharing
    share media*
    ```

    These lines share any media loaded into your system's CD-ROM drive. You can, however, limit sharing to a particular CD or series of CDs, as described in `share`(1M).

11. **Remove write permissions from the `/etc/rmmount.conf` file.**

    ```
    # chmod 444 /etc/rmmount.conf
    ```

    This step returns the file to its default permissions.

12. **Load the media.**

    The media you now load, and all subsequent media, will be available to other systems. Remember to wait until the light on the drive stops blinking before you verify this task.

    To access the media, the remote user must mount it by name, according to the instructions in "How to Access Removable Media on Remote Systems" on page 39.

13. **Verify that the media is indeed available to other systems by using the `share` command.**

    If the media is available, its share configuration will be displayed. (The shared `dummy` directory will also be displayed.)

```
# share
-    /dummy  ro "dummy dir to wake up NFS daemon"
-    /cdrom/sol_9_1202_sparc/s5   ro   ""
-    /cdrom/sol_9_1202_sparc/s4   ro   ""
-    /cdrom/sol_9_1202_sparc/s3   ro   ""
-    /cdrom/sol_9_1202_sparc/s2   ro   ""
-    /cdrom/sol_9_1202_sparc/s1   ro   ""
-    /cdrom/sol_9_1202_sparc/s0   ro   ""
```

Example 2–3 Making Local CDs Available to Other Systems

The following example shows how to make any local CD available to other systems on the network.

```
# ps -ef | grep nfsd
    root 10127  9986  0 08:25:01 pts/2    0:00 grep nfsd
    root 10118     1  0 08:24:39 ?        0:00 /usr/lib/nfs/nfsd -a
# mkdir /dummy
# vi /etc/dfs/dfstab
(Add the following line:)
 share -F nfs -o ro  /dummy
# eject cdrom0
# chmod 644 /etc/rmmount.conf
# vi /etc/rmmount.conf
(Add the following line to the File System Sharing section:)
 share cdrom*
# chmod 444 /etc/rmmount.conf
(Load a CD.)
# share
-               /dummy   ro   ""
-               /cdrom/sol_9_1202_sparc/s5   ro   ""
-               /cdrom/sol_9_1202_sparc/s4   ro   ""
-               /cdrom/sol_9_1202_sparc/s3   ro   ""
-               /cdrom/sol_9_1202_sparc/s2   ro   ""
-               /cdrom/sol_9_1202_sparc/s1   ro   ""
-               /cdrom/sol_9_1202_sparc/s0   ro   ""
```

Example 2–4 Making Local Diskettes Available to Other Systems

The following example shows how to make any local diskette available to other systems on the network.

```
# ps -ef | grep nfsd
    root 10127  9986  0 08:25:01 pts/2    0:00 grep nfsd
    root 10118     1  0 08:24:39 ?        0:00 /usr/lib/nfs/nfsd -a
# mkdir /dummy
# vi /etc/dfs/dfstab
(Add the following line:)
share -F nfs -o ro  /dummy
# eject floppy0
# chmod 644 /etc/rmmount.conf
# vi /etc/rmmount.conf
(Add the following line to the File System Sharing section.)
```

```
share floppy*
# chmod 444 /etc/rmmount.conf
(Load a diskette.)
# volcheck -v
media was found
# share
-               /dummy   ro   ""
-               /floppy/myfiles   rw   ""
```

Example 2–5 Making Local PCMCIA Memory Cards Available to Other Systems

The following example shows how to make any local PCMCIA memory card available to other systems on the network.

```
# ps -ef | grep nfsd
    root 10127  9986  0 08:25:01 pts/2    0:00 grep nfsd
    root 10118     1  0 08:24:39 ?        0:00 /usr/lib/nfs/nfsd -a
# mkdir /dummy
# vi /etc/dfs/dfstab
(Add the following line:)
share -F nfs -o ro  /dummy
# eject pcmem0
# chmod 644 /etc/rmmount.conf
# vi /etc/rmmount.conf
(Add the following line to the File System Sharing section:)
share floppy*
# chmod 444 /etc/rmmount.conf
(Load a PCMCIA memory card.)
# volcheck -v
media was found
# share
-               /dummy   ro   ""
-               /pcmem/myfiles   rw   ""
```

▼ How to Access Removable Media on Remote Systems

You can access media on a remote system by mounting it manually into your file system, provided the other system has shared its media according to the instructions in "How to Make Local Media Available to Other Systems" on page 36.

Steps

1. **Select an existing directory to serve as the mount point or create one.**

   ```
   $ mkdir directory
   ```

   directory is the name of the directory that you create to serve as a mount point for the other system's CD.

2. **Find the name of the media you want to mount.**

   ```
   $ showmount -e system-name
   ```

For example:

```
export list for starbug:
/dummy                      (everyone)
/cdrom/sol_9_1202_sparc/s5 (everyone)
/cdrom/sol_9_1202_sparc/s4 (everyone)
/cdrom/sol_9_1202_sparc/s3 (everyone)
/cdrom/sol_9_1202_sparc/s2 (everyone)
/cdrom/sol_9_1202_sparc/s1 (everyone)
/cdrom/sol_9_1202_sparc/s0 (everyone)
```

As superuser, mount the media.

```
# mount -F nfs -o ro system-name:/media/media-name local-mount-point
```

system-name	The name of the system whose media you will mount.
media-name	The name of the media you want to mount.
local-mount-point	The local directory onto which you will mount the remote media.

3. **Log out as superuser.**

4. **Verify that the media is mounted.**

```
$ ls /media
```

Example 2–6 Accessing CDs on Other Systems

The following example shows how to mount the CD named `sol_9_1202_sparc` from the remote system `starbug` onto the `/mnt` directory of the local system.

```
$ showmount -e starbug
export list for starbug:
/dummy                      (everyone)
/cdrom/sol_9_1202_sparc/s5 (everyone)
/cdrom/sol_9_1202_sparc/s4 (everyone)
/cdrom/sol_9_1202_sparc/s3 (everyone)
/cdrom/sol_9_1202_sparc/s2 (everyone)
/cdrom/sol_9_1202_sparc/s1 (everyone)
/cdrom/sol_9_1202_sparc/s0 (everyone)
$ su
Password: password
# mount -F nfs -o ro starbug:/cdrom/sol_9_1202_sparc/s0 /mnt
# exit
$ ls /mnt
Copyright  Solaris_9
```

Example 2–7 Accessing Diskettes on Other Systems

The following example shows how to mount the diskette named `myfiles` from the remote system `mars` onto the `/floppy` directory of the local system.

```
$ cd /net/mars
$ ls /floppy
floppy0     myfiles
$ su
Password: password
# mount -F nfs mars:/floppy/myfiles /floppy
# exit
$ ls /floppy
myfiles
```

Example 2–8 Accessing PCMCIA Memory Cards on Other Systems

The following example shows how to mount the PCMCIA memory card named `myfiles` from the remote system `mars` onto the `/pcmem` directory of the local system.

```
$ cd /net/mars
$ ls /pcmem
pcmem0     myfiles
$ su
Password: password
# mount -F nfs mars:/pcmem/myfiles /pcmem
# exit
$ ls /pcmem
myfiles
```

CHAPTER 3

Formatting Removable Media (Tasks)

This chapter describes how to format removable media from the command line in the Solaris environment.

For information on the procedures associated with formatting removable media, see "Formatting Removable Media (Task Map)" on page 43.

For background information on removable media, see Chapter 1.

Formatting Removable Media (Task Map)

Task	Description	For Instructions
1. Load unformatted media	Insert the media into the drive and enter the `volcheck` command.	"How to Load a Removable Media" on page 46
2. Format the media	Format removable media.	"How to Format Removable Media (`rmformat`)" on page 48
3. (Optional) Add a UFS file system	Add a UFS file system to use the diskette for transferring files.	"How to Format Removable Media for Adding a File System" on page 49

Task	Description	For Instructions
4. (Optional) Check the media	Verify the integrity of the file system on the media.	"How to Check a File System on Removable Media" on page 50
5. (Optional) Repair bad blocks on the media	Repair any bad blocks on the media, if necessary.	"How to Repair Bad Blocks on Removable Media" on page 51
6. (Optional) Apply Read or Write and Password Protection	Apply read or write protection or password protection on the media, if necessary.	"How to Enable or Disable Write Protection on Removable Media" on page 52

Formatting Removable Media Overview

The `rmformat` command is a non-superuser utility that you can use to format and protect rewritable removable media. The `rmformat` command has three formatting options:

- `quick` – This option formats removable media without certification or with limited certification of certain tracks on the media.
- `long` – This option formats removable media completely. For some devices, the use of this option might include the certification of the whole media by the drive itself.
- `force` – This option formats completely without user confirmation. For media with a password-protection mechanism, this option clears the password before formatting. This feature is useful when a password is forgotten. On media without password protection, this option forces a long format.

Formatting Removable Media Guidelines

Keep the following in mind when formatting removable media:

- Close and quit the file manager window.

  File Manager automatically displays a formatting window when you insert an unformatted media. To avoid the window, quit from File Manager. If you prefer to keep File Manager open, quit the formatting window when it appears.
- Volume manager (`vold`) mounts file systems automatically so you might have to unmount media before you can format it, if it contains an existing file system.

Removable Media Hardware Considerations

This section describes removable media hardware considerations.

Diskette Hardware Considerations

Keep the following in mind when formatting diskettes:

- For information on diskette names, see Table 2–1.
- Diskettes that are not named (that is, they have no “label”) are assigned the default name of `noname`.

A Solaris system can format diskettes for use on both Solaris and DOS systems. However, the hardware platform imposes some limitations. These limitations are summarized in the following table.

Platform Type	Diskettes Format Type
SPARC based systems	UFS
	MS-DOS or NEC-DOS (PCFS)
	UDFS
x86 based systems	UFS
	MS-DOS or NEC-DOS (PCFS)
	UDFS

Diskettes formatted for UFS are restricted to the hardware platform on which they were formatted. In other words, a UFS diskette formatted on a SPARC based platform cannot be used for UFS on an x86 platform, nor can a diskette formatted on an x86 platform be used on a SPARC based platform. This is because the SPARC and x86 UFS formats are different. SPARC uses little-endian bit coding, x86 uses big-endian.

A complete format for SunOS file systems consists of the basic “bit” formatting plus the structure to support a SunOS file system. A complete format for a DOS file system consists of the basic “bit” formatting plus the structure to support either an MS-DOS or an NEC-DOS file system. The procedures required to prepare a diskette for each type of file system are different. Therefore, before you format a diskette, consider which procedure to follow. For more information, see “Formatting Removable Media (Task Map)” on page 43.

On a Solaris system (either SPARC or x86), you can format diskettes with the following densities.

Diskette Size	Diskette Density	Capacity
3.5″	High Density (HD)	1.44 Mbytes
3.5″	Double Density (DD)	720 Kbytes

By default, the diskette drive formats a diskette to a like density. This default means that a 1.44 Mbyte drive attempts to format a diskette for 1.44 Mbytes, whether the diskette is in fact a 1.44 Mbyte diskette or not, unless you instruct it otherwise. In other words, a diskette can be formatted to its capacity or lower, and a drive can format to its capacity or lower.

PCMCIA Memory Card Hardware Considerations

A Solaris platform can format PCMCIA memory cards for use on both Solaris and DOS platforms. However, the hardware platform imposes some limitations. These limitations are summarized in the following table.

Platform Type	PCMCIA Memory Cards Format Type
SPARC based systems	UFS
	MS-DOS or NEC-DOS (PCFS)
x86 based systems	UFS
	MS-DOS or NEC-DOS (PCFS)

PCMCIA memory cards formatted for UFS are restricted to the hardware platform on which they were formatted. In other words, a UFS PCMCIA memory card formatted on a SPARC platform cannot be used for UFS on an x86 platform. Likewise, PCMCIA memory cards formatted on an x86 platform cannot be used on a SPARC platform. This is because the SPARC and x86 UFS formats are different.

A complete format for UFS file systems consists of the basic “bit” formatting plus the structure to support a UFS file system. A complete format for a DOS file system consists of the basic “bit” formatting plus the structure to support either an MS-DOS or an NEC-DOS file system. The procedures required to prepare a PCMCIA memory card for each type of file system are different. Therefore, before you format a PCMCIA memory card, consider which file system you are using.

▼ How to Load a Removable Media

Steps

1. **Insert the media.**
2. **Make sure the media is formatted.**

If you aren't sure, insert it and check the status messages in the console, as described in Step 3. If you need to format the diskette, go to "How to Format Removable Media (`rmformat`)" on page 48.

3. **Notify volume management.**

   ```
   $ volcheck -v media was found
   ```

   Two status messages are possible:

`media was found`	Volume management detected the media and will attempt to mount it in the directory described in Table 2–1.
	
	If the media is formatted properly, no error messages appear in the console.
	
	If the media is not formatted, the "`media was found`" message is still displayed, but the error messages similar to the following appear in the Console:
	
	`fd0: unformatted diskette or no diskette in the drive`
	
	`fd0: read failed (40 1 0)`
	
	`fd0: bad format`
	
	You must format the media before volume management can mount it. For more information, see Chapter 3.

`no media was found`	Volume management did not detect the media. Make sure the media is inserted properly and run volcheck again. If unsuccessful, check the media, it could be damaged. You can also try to mount the media manually.

4. **Verify that the media was mounted by listing its contents.**

   For example, do the following for a diskette:

   ```
   $ ls /floppy
   floppy0 myfiles
   ```

   As described earlier, `floppy0` is a symbolic link to the actual name of the diskette, In this case, `myfiles`. If the diskette has no name but is formatted correctly, the system will refer to it as `unnamed_floppy`.

   If nothing appears under the `/floppy` directory, the diskette was either not mounted or is not formatted properly. To find out, run the `mount` command and look for the line that begins with `/floppy` (usually at the end of the listing):

   `/floppy/`*name* on `/vol/dev/diskette0/`*name*

If the line does not appear, the diskette was not mounted. Check the console window for error messages.

▼ How to Format Removable Media (`rmformat`)

You can use the `rmformat` command to format the media. By default, this command creates two partitions on the media: partition 0 and partition 2 (the whole media).

Steps

1. **Verify that the volume manager is running, which means you can use the shorter nickname for the device name.**

   ```
   $ ps -ef | grep vold
   root    212     1  0   Nov 03 ?        0:01 /usr/sbin/vold
   ```

   For information on starting `vold`, see "How to Restart Volume Management (`vold`)" on page 31. For information on identifying media device names, see "Using Removable Media Names" on page 28.

2. **Format the removable media.**

   ```
   $ rmformat -F [ quick | long | force ] device-name
   ```

   See the previous section for more information on `rmformat` formatting options.

   If the `rmformat` output indicates bad blocks, see "How to Repair Bad Blocks on Removable Media" on page 51 for information on repairing bad blocks.

3. **(Optional) Label the removable media with an 8-character label to be used in the Solaris environment.**

   ```
   $ rmformat -b label device-name
   ```

   For information on creating a DOS label, see `mkfs_pcfs`(1M).

Example 3–1 Formatting Removable Media

This example shows how to format a diskette.

```
$ rmformat -F quick /dev/rdiskette
Formatting will erase all the data on disk.
Do you want to continue? (y/n) y
.........................................................................
```

This example shows how to format a Zip drive.

```
$ rmformat -F quick /vol/dev/aliases/zip0
Formatting will erase all the data on disk.
Do you want to continue? (y/n) y
.........................................................................
```

▼ How to Format Removable Media for Adding a File System

Steps

1. **Format the media.**

   ```
   $ rmformat -F quick device-name
   ```

2. **(Optional) Create an alternate Solaris partition table.**

   ```
   $ rmformat -s slice-file device-name
   ```

   A sample slice file looks like the following:

   ```
   slices: 0 = 0, 30MB, "wm", "home" :
                 1 = 30MB, 51MB :
                 2 = 0, 94MB, "wm", "backup" :
                 6 = 81MB, 13MB
   ```

3. **Become superuser.**

4. **Determine the appropriate file system type and select one of the following:**

   a. **Create a UFS file system.**

      ```
      # newfs device-name
      ```

   b. **Create a UDFS file system.**

      ```
      # mkfs -F udfs device-name
      ```

Example 3–2 Formatting a Diskette for a UFS File System

The following example shows how to format a diskette and create a UFS file system on the diskette.

```
$ rmformat -F quick /vol/dev/aliases/floppy0
Formatting will erase all the data on disk.
Do you want to continue? (y/n) y
$ su
# /usr/sbin/newfs /vol/dev/aliases/floppy0
newfs: construct a new file system /dev/rdiskette: (y/n)? y
/dev/rdiskette: 2880 sectors in 80 cylinders of 2 tracks, 18 sectors
        1.4MB in 5 cyl groups (16 c/g, 0.28MB/g, 128 i/g)
super-block backups (for fsck -F ufs -o b=#) at:
 32, 640, 1184, 1792, 2336,
#
```

Example 3–3 Formatting a PCMCIA Memory Card for a UFS File System

The following example shows how to format a PCMCIA memory card and create a UFS file system on the card.

```
$ rmformat -F quick /vol/dev/aliases/pcmem0
$ su
# /usr/sbin/newfs -v /vol/dev/aliases/pcmem0
newfs: construct a new file system /vol/dev/aliases/pcmem0:(y/n)? y
.
.
.
#
```

Example 3–4 Formatting Removable Media for a PCFS File System

This example shows how to create an alternate `fdisk` partition.

```
$ rmformat -F quick /dev/rdsk/c0t4d0s2:c
Formatting will erase all the data on disk.
Do you want to continue? (y/n) y
$ su
# fdisk /dev/rdsk/c0t4d0s2:c
# mkfs -F pcfs /dev/rdsk/c0t4d0s2:c
Construct a new FAT file system on /dev/rdsk/c0t4d0s2:c: (y/n)? y
#
```

This example shows how to create a PCFS file system without an `fdisk` partition.

```
$ rmformat -F quick /dev/rdiskette
Formatting will erase all the data on disk.
Do you want to continue? (y/n) y
$ su
# mkfs -F pcfs -o nofdisk,size=2 /dev/rdiskette
Construct a new FAT file system on /dev/rdiskette: (y/n)? y
#
```

▼ How to Check a File System on Removable Media

Steps

1. **Become superuser.**
2. **Identify the name service and select one of the following:**
   a. **Check a UFS file system.**

      `# fsck -F ufs` *device-name*

   b. **Check a UDFS file system.**

      `# fsck -F udfs` *device-name*

   c. **Check a PCFS file system.**

      `# fsck -F pcfs` *device-name*

Example 3–5 Checking a PCFS File System on Removable Media

The following example shows how check the consistency of a PCFS file system on media.

```
# fsck -F pcfs /dev/rdsk/c0t4d0s2
** /dev/rdsk/c0t4d0s2
** Scanning file system meta-data
** Correcting any meta-data discrepancies
1457664 bytes.
0 bytes in bad sectors.
0 bytes in 0 directories.
0 bytes in 0 files.
1457664 bytes free.
512 bytes per allocation unit.
2847 total allocation units.
2847 available allocation units.
#
```

▼ How to Repair Bad Blocks on Removable Media

You can only use the `rmformat` command to verify, analyze, and repair bad sectors that are found during verification if the drive supports bad block management. Most diskettes and PCMCIA memory cards do not support bad block management.

If the drive supports bad block management, a best effort is made to rectify the bad block. If the bad block cannot be rectified despite the best effort mechanism, a message indicates a failure to repair.

Steps

1. **Repair bad blocks on removable media.**

   ```
   $ rmformat -c block-numbers device-name
   ```

   Supply the block number in decimal, octal, or hexadecimal format from a previous `rmformat` session.

2. **Verify the media.**

   ```
   $ rmformat -V read device-name
   ```

Applying Read or Write and Password Protection to Removable Media

You can apply read protection or write protection and set a password on Iomega media such as Zip drives and Jaz drives.

▼ How to Enable or Disable Write Protection on Removable Media

Steps

1. **Determine whether you want to enable or disable write protection and select one of the following:**

   a. **Enable write protection.**

   ```
   $ rmformat -w enable device-name
   ```

   b. **Disable write protection.**

   ```
   $ rmformat -w disable device-name
   ```

2. **Verify whether the media's write protection is enabled or disabled.**

   ```
   $ rmformat -p device-name
   ```

▼ How to Enable or Disable Read or Write Protection and a Password on Iomega Media

You can apply a password with a maximum of 32 characters for Iomega media that support this feature. You cannot set read protection or write protection without a password on Iomega media. In this situation, you are prompted to provide a password.

You receive a warning message if you attempt to apply a password on media that does not support this feature.

Steps

1. **Determine whether you want to enable or disable read protection or write protection and a password.**

   a. **Enable read protection or write protection.**

   ```
   $ rmformat -W enable device-name
   Please enter password (32 chars maximum): xxx
   Please reenter password:
   ```

   ```
   $ rmformat -R enable device-name
   Please enter password (32 chars maximum): xxx
   Please reenter password:
   ```

   b. **Disable read protection or write protection and remove the password.**

   ```
   $ rmformat -W disable device-name
   Please enter password (32 chars maximum): xxx
   ```

```
$ rmformat -R disable device-name
Please enter password (32 chars maximum): xxx
```

2. **Verify whether the media's read protection or write protection is enabled or disabled.**

   ```
   $ rmformat -p device-name
   ```

Example 3–6 Enabling or Disabling Read or Write Protection

This example shows how to enable write protection and set a password on a Zip drive.

```
$ rmformat -W enable /vol/dev/aliases/zip0
Please enter password (32 chars maximum): xxx
Please reenter password: xxx
```

This example shows how to disable write protection and remove the password on a Zip drive.

```
$ rmformat -W disable /vol/dev/aliases/zip0
Please enter password (32 chars maximum): xxx
```

This example shows how to enable read protection and set a password on a Zip drive.

```
rmformat -R enable /vol/dev/aliases/zip0
Please enter password (32 chars maximum): xxx
Please reenter password: xxx
```

This example shows to disable read protection and remove the password on a Zip drive.

```
$ rmformat -R disable /vol/dev/aliases/zip0
Please enter password (32 chars maximum): xxx
```

CHAPTER 4

Writing CDs (Tasks)

This chapter provides step-by-step instructions for writing and copying data and audio CDs with the `cdrw` command.

Working with Audio and Data CDs

You can use the `cdrw` command to write CD file systems in ISO 9660 format with Rock Ridge or Joliet extensions on CD-R or CD-RW media devices.

You can use the `cdrw` command to:

- Create data CDs
- Create audio CDs
- Extract audio data from an audio CD
- Copy CDs
- Erase CD-RW media

The `cdrw` command is available in the following releases:

- Software Supplement for the Solaris 8 Operating Environment 1/01 CD
- Part of the Solaris environment starting in the Solaris 9 release

For information on recommended CD-R or CD-RW devices, go to `http://www.sun.com/io_technologies/pci/removable.html`.

CD Media Commonly Used Terms

Commonly used terms when referring to CD media are:

Term	Description
CD-R	CD read media that can be written once and after that, can only be read from.
CD-RW	CD rewritable media that can be written to and erased. CD-RW media can only be read by CD-RW devices.
ISO 9660	ISO, an acronym for Industry Standards Organization, is an organization that sets standards computer storage formats. An ISO 9660 file system is a standard CD-ROM file system that enables you to read the same CD-ROM on any major computer platform. The standard, issued in 1988, was written by an industry group named High Sierra, named after the High Sierra Hotel in Nevada. Almost all computers with CD-ROM drives can read files from an ISO 9660 file system.
Joliet extensions	Adds Windows™ file system information.
Rock Ridge extensions	Adds UNIX™ file system information. (Rock Ridge is named after the town in Blazing Saddles.) **Note –** These extensions are not exclusive. You can specify both `mkisofs -R` and `-j` options for compatibility with both systems. (See mkisofs(1M) for details.)
MMC-compliant record	Acronym for Multi Media Command, which means these recorder comply with a common command set. Programs that can write to one MMC-compliant recorder should be able to write to all others.
Red Book CDDA	Acronym for Compact Disc Digital Audio, which is an industry standard method for storing digital audio on compact discs. It is also known by the term “Red Book” format. The official industry specification calls for one or more audio files sampled in 16-bit stereo sound at a sampling rate of 44.1 kilohertz (kHz).

Commonly used terms when working with the CD media are:

Term	Description
blanking	The process of erasing data from the CD-RW media.
`mkisofs`	Command for making a ISO file system to write onto a CD.
session	A complete track with lead-in and lead-out information.
track	A complete data or audio unit.

Writing Data and Audio CDs

The process of writing to a CD cannot be interrupted and needs a constant stream of data. Consider using the `cdrw -S` option to simulate writing to the media to verify if the system can provide data at a rate good enough for writing to the CD.

Write errors can be caused by one of the following:

- The media cannot handle the drive speed. For example, some media are only certified for 2x or 4x speeds.
- The system is running too many heavy processes that can starve the writing process.
- Network congestion can cause delays in reading the image if the image is on a remote system.
- The source drive might be slower than the destination drive when copying from CD-to-CD.

If any of these problems occur, you can lower the writing speed of the device with the `cdrw -p` option.

For example, simulate writing at 4x speed.

```
$ cdrw -iS -p 4 image.iso
```

You can also use the `cdrw -C` option to use the stated media capacity for copying an 80–minute CD. Otherwise, the `cdrw` command uses a default value of 74 minutes for copying an audio CD.

For more information, see `cdrw`(1).

Restricting User Access to Removable Media with RBAC

By default, all users can access removable media starting in the Solaris 9 release. However, you can restrict user access to removable media by setting up a role through role based access control (RBAC). Access to removable media is restricted by assigning the role to a limited set of users.

For a discussion of using roles, see Chapter 5, “Role-Based Access Control (Overview),” in *System Administration Guide: Security Services*.

▼ How to Restrict User Access to Removable Media with RBAC

Steps

1. **Become superuser or assume an equivalent role.**
2. **Start the Solaris Management Console.**

   ```
   $ /usr/sadm/bin/smc &
   ```

   For more information on starting the console, see “Starting the Solaris Management Console” in *System Administration Guide: Basic Administration*.
3. **Set up a role that includes the Device Management rights.**

   For more information, see Chapter 6, “Role-Based Access Control (Tasks),” in *System Administration Guide: Security Services*.
4. **Add users who need to use the `cdrw` command to the newly created role.**
5. **Comment the following line in the `/etc/security/policy.conf` file.**

   ```
   AUTHS_GRANTED=solaris.device.cdrw
   ```

   If you do not do this step, all users still have access to the `cdrw` command, not just the members of the device management role.

   After this file is modified, the device management role members are the only users who can use the `cdrw` command. Everyone else is denied access with the following message:

   ```
   Authorization failed, Cannot access disks.
   ```

How to Identify a CD Writer

Use the `cdrw -l` command to identify the CD writers on the system.

```
$ cdrw -l
Looking for CD devices...
    Node              |    Connected Device            |  Device type
```

```
----------------------+---------------------------------+-----------------
 cdrom0               | YAMAHA   CRW8424S         1.0d | CD Reader/Writer
```

If you want to use a specific CD writer, use the `-d` option. For example:

```
$ cdrw -a filename.wav -d cdrom2
```

Use the `cdrw -M` command to to identify whether the media is blank or whether there is an existing table of contents.

```
$ cdrw -M

Device : YAMAHA   CRW8424S
Firmware : Rev. 1.0d (06/10/99)
Media is blank
%
```

▼ How to Check the CD Media

The `cdrw` command works with or without `vold` running. However, you must have superuser or role access to stop and start the `vold` daemon.

Steps

1. **Insert a CD into the CD-RW device.**

   The CD can be any CD that the device can read.

2. **Check that the CD-RW drive is connected properly by listing the device.**

   ```
   $ cdrw -l
    Looking for CD devices...
       Node                   Connected Device                Device type
   ----------------------+---------------------------------+-----------------
    cdrom1               | YAMAHA   CRW8424S         1.0d | CD Reader/Writer
   ```

3. **(Optional) If you do not see the drive in the list, you might have to do a reconfiguration boot so that the system recognizes the device.**

   ```
   # touch /reconfigure
   # init 6
   ```

   Or, use the following commands to add the CD-RW device without rebooting the system.

   ```
   # drvconfig
   # disks
   ```

   Then restart `vold`.

   ```
   # /etc/init.d/vold stop
   # /etc/init.d/vold start
   ```

Creating a Data CD

Prepare the data first by using the `mkisofs` command to convert the file and file information into the High Sierra format used on CDs.

▼ How to Create an ISO 9660 File System for a Data CD

Steps

1. **Insert a blank CD into the CD-RW device.**
2. **Create the ISO 9660 file system on the new CD.**

   ```
   $ mkisofs -r /pathname > cd-file-system
   ```

`-r`	Creates Rock Ridge information and resets file ownerships to zero.
/pathname	Identifies the pathname used to create the ISO 9660 file system.
> *cd-file-system*	Identifies the name of the file system to be put on the CD.

3. **Copy the CD file system onto the CD.**

   ```
   $ cdrw -i cd-file-system
   ```

`-i` *cd-file-system*	Specifies the image file for creating a data CD.

Example 4–1 Creating an ISO 9660 File System for a Data CD

The following example shows how to create a ISO 9660 file system for a data CD.

```
$ mkisofs -r /home/dubs/ufs_dir > ufs_cd
Total extents actually written = 56
Total translation table size: 0
Total rockridge attributes bytes: 329
Total directory bytes: 0
Path table size(bytes): 10
Max brk space used 8000
56 extents written (0 Mb)
```

Then copy the CD file system onto the CD. For example:

```
$ cdrw -i ufs_cd
Initializing device...done.
Writing track 1...done.
Finalizing (Can take several minutes)...done.
```

▼ How to Create a Multi-Session Data CD

This procedure describes how to put more than one session on the CD. This procedure includes an example of copying the `infoA` and `infoB` directories onto the CD.

Steps

1. **Create the file system for the first CD session.**

```
$ mkisofs -o infoA -r -V my_infoA /data/infoA
Total translation table size: 0
Total rockridge attributes bytes: 24507
Total directory bytes: 34816
Path table size(bytes): 98
Max brk space used 2e000
8929 extents written (17 Mb)
```

`-o infoA`	Identifies the name of the ISO file system.
`-r`	Creates Rock Ridge information and resets file ownerships to zero.
`-V my_infoA`	Identifies a volume label to be used as the mount point by `vold`.
`/data/infoA`	Identifies the ISO image directory to create.

2. **Copy the ISO file system for the first session onto the CD.**

```
$ cdrw -iO infoA
Initializing device...done.
Writing track 1...done.
done.
Finalizing (Can take several minutes)...done.
```

`-i` *infoA*	Identifies the name of the image file to write to the CD.
`-O`	Keeps the CD open for writing.

3. **Re-insert the CD after it is ejected.**

4. **Identify the pathname of the CD media to include in the next write session.**

```
$ eject -n
.
.
.
cdrom0 -> /vol/dev/rdsk/c2t4d0/my_infoA
```

Note the `/vol/dev/...` pathname.

5. **Identify the next writeable address on the CD to write the next session.**

```
% cdrw -M /cdrom
Device : YAMAHA   CRW8424S
Firmware : Rev. 1.0d (06/10/99)

Track No. |Type    |Start address
```

```
----------+--------+------------
 1        |Audio   |0
 2        |Audio   |33057
 3        |Data    |60887
 4        |Data    |68087
 5        |Data    |75287
Leadout   |Data    |84218

Last session start address: 75287
Next writable address: 91118
```

Note the address in the `Next writable address:` output so you can provide this when you write the next session.

6. **Create the next ISO file system for the next CD session and write it onto the CD.**

```
$ mkisofs -o infoB -r -C 0,91118 -M /vol/dev/rdsk/c2t4d0/my_infoA
/data/infoB
Total translation table size: 0
Total rockridge attributes bytes: 16602
Total directory bytes: 22528
Path table size(bytes): 86
Max brk space used 20000
97196 extents written (189 Mb)
```

`-o` *infoB*	Identifies the name of the ISO file system.
`-r`	Creates Rock Ridge information and resets file ownerships to zero.
`-C` *0,91118*	Identifies the starting address of the first session and the next writable address.
`-M /vol/dev/rdsk/c2t4d0/my_infoA`	Specifies the path of the existing ISO image to be merged.
`/data/infoB`	Identifies the ISO image directory to create.

Creating an Audio CD

You can use the `cdrw` command to create audio CDs from individual audio tracks or from `.au` and `.wav` files.

The supported audio formats are:

Format	Description
sun	Sun .au files with data in Red Book CDDA format
wav	RIFF (.wav) files with data in Red Book CDDA format
cda	.cda files with raw CD audio data, which is 16–bit PCM stereo at 44.1 kHz sample rate in little-endian byte order)
aur	.aur files with raw CD data in big-endian byte order

If no audio format is specified, the `cdrw` command tries to determine the audio file format based on the file extension. The case of the characters in the extension is ignored.

▼ How to Create an Audio CD

This procedure describes how to copy audio files onto a CD.

Steps

1. **Insert a blank CD into the CD-RW device.**
2. **Change to the directory that contains the audio files.**
   ```
   $ cd /myaudiodir
   ```
3. **Copy the audio files onto the CD.**
   ```
   $ cdrw -a track1.wav track2.wav track3.wav
   ```
   The `-a` option creates an audio CD.

Example 4–2 Creating an Audio CD

The following example shows how to create an audio CD.

```
$ cdrw -a bark.wav chirp.au meow.wav
Initializing device...done.
Writing track 1...done.
done.
Writing track 2...done.
Writing track 3...done.
done.
Finalizing (Can take several minutes)...done.
```

The following example shows how to create a multisession audio CD. The CD is ejected after the first session is written. Re-insert the CD before the next writing session.

```
$ cdrw -aO groucho.wav chico.au harpo.wav
Initializing device...done.
Writing track 1...done.
```

```
done.
Writing track 2...done.
Writing track 3...done.
done.
Finalizing (Can take several minutes)...done.
<Re-insert CD>
$ cdrw -a zeppo.au
Initializing device...done.
Writing track 1...done.
done.
Finalizing (Can take several minutes)...done.
```

▼ How to Extract an Audio Track on a CD

Use the following procedure to extract an audio track from a CD and copy it to a new CD.

If you don't use the `cdrw -T` option to specify the audio file type, `cdrw` uses the filename extension to determine the audio file type. For example, the `cdrw` command detects that this file is a `.wav` file.

```
$ cdrw -x 1 testme.wav
```

Steps

1. **Insert a audio CD into the CD-RW device.**
2. **Extract an audio track.**

   ```
   $ cdrw -x -T audio-type 1 audio-file
   ```

`-x`	Extracts audio data from an audio CD.
T *audio-type*	Identifies the type of audio file to be extracted. Supported audio types are `sun`, `wav`, `cda`, or `aur`.

3. **Copy the track to a new CD.**

   ```
   $ cdrw -a audio-file
   ```

Example 4–3 Extracting and Creating Audio CDs

The following example shows how to extract the first track from an audio CD and names the file `song1.wav`.

```
$ cdrw -x -T wav 1 song1.wav
Extracting audio from track 1...done.
```

This example describes how to copy a track to an audio CD.

```
$ cdrw -a song1.wav
Initializing device...done.
Writing track 1...done.
```

```
Finalizing (Can take several minutes)...done.
```

▼ How to Copy a CD

This procedure describes how to extract all the tracks from an audio CD into a directory and then copy all them onto a blank CD.

Note – By default, the `cdrw` command copies the CD into the `/tmp` directory. The copying might require up to 700 Mbytes of free space. If there is insufficient space in the `/tmp` directory for copying the CD, use the `-m` option to specify an alternate directory.

Steps

1. **Insert an audio CD into a CD-RW device.**
2. **Extract the tracks from the audio CD.**

   ```
   $ mkdir music_dir
   $ cdrw -c -m music_dir
   ```

   An `Extracting audio ...` message is display for each track.

   The CD is ejected when all the tracks are extracted.
3. **Insert a blank CD and press Return.**

   After the tracks are extracted, the audio CD is ejected, and you are prompted to insert a blank CD.

Example 4–4 Copying a CD

This example describes how to copy one CD to another CD. You must have two CD-RW devices to do this task.

```
$ cdrw -c -s cdrom0 -d cdrom1
```

▼ How to Erase CD-RW Media

You have to erase existing CD-RW data before the CD can be rewritten.

Step

● **Erase the entire media or just the last session on the CD by selecting one of the following:**

a. **Erase the last session only.**

   ```
   $ cdrw -d cdrom0 -b session
   ```

Erasing just the last session with the `-b session` option is faster than erasing the entire media with the `-b all` option. You can use the `-b session` option even if you used the `cdrw` command to create a data or audio CD in just one session.

b. **Erase the entire media.**

```
$ cdrw -d cdrom0 -b all
```

CHAPTER 5

Managing Devices (Tasks)

This chapter provides overview information and step-by-step instructions for managing peripheral devices, such as disks, CD-ROMs, and tape devices, in the Solaris environment.

This is a list of the overview information in this chapter.

This is a list of the step-by-step instructions in this chapter.

For information about accessing peripheral devices, see Chapter 9.

Device management in the Solaris environment usually involves adding and removing peripheral devices from systems, possibly adding a third-party device driver to support a device, and displaying system configuration information.

What's New in Device Management?

This section provides information about new device management features.

USB Device Enhancements

For information on new USB device enhancements, see "What's New in USB Devices?" on page 107.

Where to Find Device Management Tasks

The following table describes where to find step-by-step instructions for hot-plugging devices and adding serial devices, such as printers and modems, and peripheral devices, such as a disk, CD-ROM, or tape devices.

TABLE 5–1 Where to Find Instructions for Adding a Device

Device Management Task	For More Information
Adding a disk that is not hot-pluggable	Chapter 12 or Chapter 13
Hot-plugging a SCSI or PCI device	"SCSI Hot-Plugging With the `cfgadm` Command" on page 84 or "x86: PCI Hot-Plugging With the `cfgadm` Command" on page 94
Hot-plugging a USB device	"Hot-Plugging USB Devices (Task Map)" on page 134
Adding a CD-ROM or tape device	"How to Add a Peripheral Device" on page 75
Adding a modem	Chapter 10, "Managing Terminals and Modems (Overview)," in *System Administration Guide: Advanced Administration*
Adding a printer	Chapter 2, "Managing Printing Services (Overview)," in *System Administration Guide: Advanced Administration*

About Device Drivers

A computer typically uses a wide range of peripheral and mass-storage devices. Your system, for example, probably has a disk drive, a keyboard and a mouse, and some kind of magnetic backup medium. Other commonly used devices include CD-ROM drives, printers and plotters, light pens, touch-sensitive screens, digitizers, and tablet-and-stylus pairs.

The Solaris software does not directly communicate with all these devices. Each type of device requires different data formats, protocols, and transmission rates.

A *device driver* is a low-level program that allows the operating system to communicate with a specific piece of hardware. The driver serves as the operating system's "interpreter" for that piece of hardware.

Automatic Configuration of Devices

The kernel, consisting of a small generic core with a platform-specific component and a set of modules, is configured automatically in the Solaris environment.

A kernel module is a hardware or software component that is used to perform a specific task on the system. An example of a *loadable* kernel module is a device driver that is loaded when the device is accessed.

The platform-independent kernel is `/kernel/genunix`. The platform-specific component is `/platform/‘uname -m‘/kernel/unix`.

The kernel modules are described in the following table.

TABLE 5–2 Description of Kernel Modules

Location	Directory Contents
`/platform/‘uname -m‘/kernel`	Platform-specific kernel components
`/kernel`	Kernel components common to all platforms that are needed for booting the system
`/usr/kernel`	Kernel components common to all platforms within a particular instruction set

The system determines what devices are attached to it at boot time. Then, the kernel configures itself dynamically, loading needed modules into memory. At this time, device drivers are loaded when devices, such as disk and tape devices, are accessed. This process is called *autoconfiguration* because all kernel modules are loaded automatically when they are needed.

You can customize the way in which kernel modules are loaded by modifying the `/etc/system` file. For instructions on modifying this file, see `system`(4).

Features and Benefits of Autoconfiguration

The benefits of autoconfiguration are as follows:

- Main memory is used more efficiently because modules are loaded when needed.
- There is no need to reconfigure the kernel when new devices are added to the system.
- Drivers can be loaded and tested without having to rebuild the kernel and reboot the system.

You will use autoconfiguration is used by a system administrator when you add a new device (and driver) to the system. At this time, you will perform a reconfiguration boot so that the system will recognize the new device.

What You Need for Unsupported Devices

Device drivers needed to support a wide range of standard devices are included in the Solaris environment. These drivers can be found in the `/kernel/drv` and `/platform/'uname -m'/kernel/drv` directories.

However, if you've purchased an unsupported device, the manufacturer should provide the software that is needed for the device to be properly installed, maintained, and administered.

At a minimum, this software includes a device driver and its associated configuration (`.conf`) file. The `.conf` files reside in the `drv` directories. This software might also include custom maintenance and administrative utilities since the device might be incompatible with Solaris utilities.

Contact your device manufacturer for more information.

Displaying Device Configuration Information

Three commands are used to display system and device configuration information.

Command	Man Page	Description
`prtconf`	`prtconf`(1M)	Displays system configuration information, including total amount of memory and the device configuration as described by the system's device hierarchy. The output displayed by this command depends upon the type of system.
`sysdef`	`sysdef`(1M)	Displays device configuration information including system hardware, pseudo devices, loadable modules, and selected kernel parameters.
`dmesg`	`dmesg`(1M)	Displays system diagnostic messages as well as a list of devices attached to the system since the last reboot.

For information on the device names that are used to identify devices on the system, see "Device Naming Conventions" on page 152.

`driver not attached` Message

The following driver-related message might be displayed by the `prtconf` and `sysdef` commands:

```
device, instance #number (driver not attached)
```

This message does not always mean that a driver is unavailable for this device. This message means that no driver is *currently* attached to the device instance because there is no device at this node or the device is not in use. Drivers are loaded automatically when the device is accessed and unloaded when the device is not in use.

Identifying a System's Devices

Use the output of the prtconf and sysdef commands to identify which disk, tape, and CD-ROM devices are connected to the system. The output of these commands display the driver not attached messages next to the device instances. Since these devices are always being monitored by some system process, the driver not attached message is usually a good indication that there is no device at that device instance.

For example, the following prtconf output identifies a device at instance #3 and instance #6, which is probably a disk device at target 3 and a CD-ROM device at target 6 of the first SCSI host adapter (esp, instance #0).

```
$ /usr/sbin/prtconf
.
.
.

esp, instance #0
            sd (driver not attached)
            st (driver not attached)
            sd, instance #0 (driver not attached)
            sd, instance #1 (driver not attached)
            sd, instance #2 (driver not attached)
            sd, instance #3
            sd, instance #4 (driver not attached)
            sd, instance #5 (driver not attached)
            sd, instance #6
.
.
.
```

You can use the following command to display only the devices that are attached to the system.

```
$ prtconf | grep -v not
```

You can also glean device information from the sysdef output.

How to Display System Configuration Information

Use the prtconf command to display system configuration information.

```
# /usr/sbin/prtconf
```

Use the sysdef command to display system configuration information that include pseudo devices, loadable modules, and selected kernel parameters.

```
# /usr/sbin/sysdef
```

Examples—Displaying System Configuration Information

The following `prtconf` output is displayed on a SPARC based system.

```
# prtconf
System Configuration:  Sun Microsystems  sun4u
Memory size: 128 Megabytes
System Peripherals (Software Nodes):
SUNW,Ultra-5_10
    packages (driver not attached)
        terminal-emulator (driver not attached)
        deblocker (driver not attached)
        obp-tftp (driver not attached)
        disk-label (driver not attached)
        SUNW,builtin-drivers (driver not attached)
        sun-keyboard (driver not attached)
        ufs-file-system (driver not attached)
    chosen (driver not attached)
    openprom (driver not attached)
        client-services (driver not attached)
    options, instance #0
    aliases (driver not attached)
    memory (driver not attached)
    virtual-memory (driver not attached)
    pci, instance #0
        pci, instance #0
            ebus, instance #0
                auxio (driver not attached)
                power, instance #0
                SUNW,pll (driver not attached)
                se, instance #0
                su, instance #0
                su, instance #1
                ecpp (driver not attached)
                fdthree, instance #0
.
.
.
```

The following `sysdef` output is displayed from an x86 based system.

```
# sysdef
* Hostid
*
  29f10b4d
*
* i86pc Configuration
*
*
* Devices
*
+boot (driver not attached)
memory (driver not attached)
aliases (driver not attached)
chosen (driver not attached)
```

```
i86pc-memory (driver not attached)
i86pc-mmu (driver not attached)
openprom (driver not attached)
options, instance #0
packages (driver not attached)
delayed-writes (driver not attached)
itu-props (driver not attached)
isa, instance #0
    motherboard (driver not attached)
    pnpADP,1542, instance #0
    asy, instance #0
    asy, instance #1
    lp, instance #0 (driver not attached)
    fdc, instance #0
        fd, instance #0
        fd, instance #1 (driver not attached)
    kd (driver not attached)
    kdmouse (driver not attached)
.
.
.
```

How to Display Device Information

Display device information with the `dmesg` command.

```
# /usr/sbin/dmesg
```

The `dmesg` output is displayed as messages on the system console and identifies which devices are connected to the system since the last reboot.

Examples—Displaying Device Information

The following `dmesg` output is displayed from a SPARC based system.

```
Apr  2 13:26:19 venus genunix: [ID 540533 kern.notice] SunOS Release 5.9 Version Generic ...
Apr  2 13:26:19 venus genunix: [ID 943905 kern.notice] Copyright 1983-2003...
Apr  2 13:26:19 venus genunix: [ID 678236 kern.info] Ethernet address ...
Apr  2 13:26:19 venus unix: [ID 389951 kern.info] mem = 65536K (0x4000000)
Apr  2 13:26:19 venus unix: [ID 930857 kern.info] avail mem = 57688064
Apr  2 13:26:19 venus rootnex: [ID 466748 kern.info] root nexus =
Sun Ultra 1 SBus (UltraSPARC 167MHz)
```

The following `dmesg` output is displayed from an x86 based system.

```
# dmesg
Dec 17 16:32:10 naboo unix: [ID 930857 kern.info] avail mem = 1037565952
Dec 17 16:32:10 naboo rootnex: [ID 466748 kern.info] root nexus = i86pc
Dec 17 16:32:10 naboo unix: [ID 406534 kern.info] ACPI detected: 2 13 0 0
Dec 17 16:32:10 naboo rootnex: [ID 349649 kern.info] pci1 at root: isa 0x0
Dec 17 16:32:10 naboo genunix: [ID 936769 kern.info] pci1 is /pci@1,0
Dec 17 16:32:10 naboo pcplusmp: [ID 637496 kern.info] pcplusmp: ...
Dec 17 16:32:10 naboo pci: [ID 370704 kern.info] PCI-device: ...
Dec 17 16:32:10 naboo genunix: [ID 936769 kern.info] cadp1601 ...
Dec 17 16:32:13 naboo scsi: [ID 193665 kern.info] sd5 at cadp1601: ...
.
.
.
```

Adding a Peripheral Device to a System

Adding a new (non-hot-pluggable) peripheral device usually involves the following:

- Shutting down the system
- Connecting the device to the system
- Rebooting the system

Use the "How to Add a Peripheral Device" on page 75 procedure to add the following devices that are not hot-pluggable to a system:

- CD-ROM
- Secondary disk drive
- Tape drive
- SBUS card

In some cases, you might have to add a third-party device driver to support the new device.

For information on hot-plugging devices, see Chapter 6.

▼ How to Add a Peripheral Device

Steps

1. **Become superuser.**

2. **Follow steps 2 and 3 of "How to Add a Device Driver" on page 76 if you need to add a device driver to support the device.**

3. **Create the `/reconfigure` file.**

   ```
   # touch /reconfigure
   ```

   The `/reconfigure` file will cause the Solaris software to check for the presence of any newly installed devices the next time you turn on or boot your system.

4. **Shut down the system.**

   ```
   # shutdown -i0 -g30 -y
   ```

`-i0`	Brings the system to the 0 init state, which is the appropriate state for turning the system power off for adding and removing devices.
`-g30`	Shuts the system down in 30 seconds. The default is 60 seconds.
`-y`	Continues the system shutdown without user intervention. Otherwise, you are prompted to continue the shutdown process.

5. **Select one of the following to turn off power to the system after it is shut down.**

   a. **For SPARC platforms, it is safe to turn off power if the `ok` prompt is displayed.**

   b. **For x86 platforms, it is safe to turn off power if the `type any key to continue` prompt is displayed.**

   Refer to the hardware installation guide that accompanies your system for the location of the power switch.

6. **Turn off power to all external devices.**

   For the location of power switches on any peripheral devices, refer to the hardware installation guides that accompany your peripheral devices.

7. **Install the peripheral device, making sure that the device you are adding has a different target number than the other devices on the system.**

   You often will find a small switch located at the back of the disk for selecting the target number.

   Refer to the hardware installation guide that accompanies the peripheral device for information on installing and connecting the device.

8. **Turn on the power to the system.**

   The system boots to multiuser mode and the login prompt is displayed.

9. **Verify that the peripheral device has been added by attempting to access the device.**

   For information on accessing the device, see Chapter 9.

▼ How to Add a Device Driver

This procedure assumes that the device has already been added to the system. If not, see "What You Need for Unsupported Devices" on page 70.

Steps

1. **Become superuser.**

2. **Place the tape, diskette, or CD-ROM into the drive.**

3. **Install the driver.**

```
# pkgadd -d device package-name
```

`-d` *device*	Identifies the device path name that contains the package.
package-name	Identifies the package name that contains the device driver.

4. **Verify that the package has been added correctly.**

```
# pkgchk package-name
#
```

The system prompt returns with no response if the package is installed correctly.

Example 5–1 Adding a Device Driver

The following example shows how to install and verify a package called `XYZdrv`.

```
# pkgadd XYZdrv
(licensing messages displayed)
.
.
.
Installing XYZ Company driver as <XYZdrv>
.
.
.
Installation of <XYZdrv> was successful.
# pkgchk XYZdrv
#
```

CHAPTER 6

Dynamically Configuring Devices (Tasks)

This chapter provides instructions for dynamically configuring devices in the Solaris environment. You can add, remove, or replace devices in the Solaris environment while the system is still running, if the system components support hot-plugging. If the system components do not support hot-plugging, you can reboot the system to reconfigure the devices.

For information on the procedures associated with dynamically configuring devices, see the following:

- “SCSI Hot-Plugging With the `cfgadm` Command (Task Map)” on page 83
- “PCI Hot-Plugging With the `cfgadm` Command (Task Map)” on page 93
- “Application Developer RCM Script (Task Map)” on page 100
- “System Administrator RCM Script (Task Map)” on page 100

For information on hot-plugging USB devices with the `cfgadm` command, see “Hot-Plugging USB Devices With the `cfgadm` Command” on page 144.

For information about accessing devices, see Chapter 9.

Dynamic Reconfiguration and Hot-Plugging

Hot-plugging is the ability to physically add, remove, or replace system components while the system is running. *Dynamic reconfiguration* refers to the ability to hot-plug system components. This term also refers to the general ability to move system resources (both hardware and software) around in the system or to disable them in some way without physically removing them from the system.

You can hot-plug the following devices with the `cfgadm` command:

- USB devices on SPARC and x86 platforms
- SCSI devices on SPARC and x86 platforms
- PCI devices on x86 platforms

Features of the `cfgadm` command include the following:

- Displaying system component status
- Testing system components
- Changing component configurations
- Displaying configuration help messages

The benefit of using the `cfgadm` command to reconfigure systems components is that you can add, remove, or replace components while the system is running. An added benefit is that the `cfgadm` command guides you through the steps needed to add, remove, or replace system components.

For step-by-step instructions on hot-plugging SCSI components, see `cfgadm`(1M) and "SCSI Hot-Plugging With the `cfgadm` Command" on page 84. For step-by-step instructions on hot-plugging PCI adapter cards on x86 based systems, see "x86: PCI Hot-Plugging With the `cfgadm` Command" on page 94.

Note – Not all SCSI and PCI controllers support hot-plugging with the `cfgadm` command.

As part of Sun's high availability strategy, dynamic reconfiguration is expected to be used in conjunction with additional layered products, such as alternate pathing or fail-over software. Both products provide fault tolerance in the event of a device failure.

Without any high availability software, you can replace a failed device by manually stopping the appropriate applications, unmounting noncritical file systems, and then proceeding with the add or remove operations.

Note – For information about hot-plugging devices on your specific hardware configuration, such as enterprise-level systems, please refer to your hardware configuration documentation.

Attachment Points

The `cfgadm` command displays information about *attachment points*, which are locations in the system where dynamic reconfiguration operations can occur.

An attachment point consists of the following:

- An *occupant*, which represents a hardware component that can be configured into the system

- A *receptacle*, which is the location that accepts the occupant

Attachment points are represented by logical and physical attachment point IDs (`Ap_Ids`). The physical `Ap_Id` is the physical pathname of the attachment point. The logical `Ap_Id` is a user-friendly alternative for the physical `Ap_Id`. For more information on `Ap_Ids`, refer to `cfgadm`(1M).

The logical `Ap_Id` for a SCSI Host Bus Adapter (HBA), or SCSI controller, is usually represented by the controller number, such as `c0`.

In cases where no controller number has been assigned to a SCSI HBA, then an internally-generated unique identifier is provided. An example of a unique identifier for a SCSI controller is the following:

```
fas1:scsi
```

The logical `Ap_Id` for a SCSI device usually looks like this:

HBA-logical-apid::device-identifier

In the following example, `c0` is the logical `Ap_Id` for the SCSI HBA:

```
c0::dsk/c0t3d0
```

The device identifier is typically derived from the logical device name for the device in the `/dev` directory. For example, a tape device with logical device name, `/dev/rmt/1`, has the following logical `Ap_Id`:

```
c0::rmt/1
```

If a logical `Ap_Id` of a SCSI device cannot be derived from the logical name in the `/dev` directory, then an internally-generated unique identifier is provided. An example of an identifier for the `/dev/rmt/1` tape device is the following:

```
c0::st4
```

For more information on SCSI `Ap_Ids`, refer to `cfgadm_scsi`(1M).

The `cfgadm` command represents all resources and dynamic reconfiguration operations in terms of a common set of states (such as configured, unconfigured) and operations (connect, configure, unconfigure, and so on). For more information on these common states and operations, see `cfgadm`(1M).

The receptacle and occupant states for the SCSI HBA attachment points are as follows:

Receptacle State	Description	Occupant State	Description
`empty`	N/A for SCSI HBA	`configured`	One or more devices configured on the bus

Receptacle State	Description	Occupant State	Description
`disconnected`	Bus quiesced	`unconfigured`	No devices configured
`connected`	Bus active		

Receptacle and occupant states for SCSI device attachment points are as follows:

Receptacle State	Description	Occupant State	Description
`empty`	N/A for SCSI devices	`configured`	Device is configured
`disconnected`	Bus quiesced	`unconfigured`	Device is not configured
`connected`	Bus active		

The state of SCSI attachment points is unknown unless there is special hardware to indicate otherwise. For instructions on displaying SCSI component information, see "How to Display Information About SCSI Devices" on page 84.

x86: Detaching PCI Adapter Cards

A PCI adapter card that is hosting nonvital system resources can be removed if the device driver supports hot-plugging. A PCI adapter card is not detachable if it is a vital system resource. For a PCI adapter card to be detachable the following conditions must be met:

- The device driver must support hot-plugging.
- Critical resources must be accessible through an alternate pathway.

For example, if a system has only one Ethernet card installed in it, the Ethernet card cannot be detached without losing the network connection. This detachment requires additional layered software support to keep the network connection active.

x86: Attaching PCI Adapter Cards

A PCI adapter card can be added to the system as long as the following conditions are met:

- There are slots available.
- The device driver supports hot-plugging for this adapter card.

For step-by-step instructions on adding or removing a PCI adapter card, see "x86: PCI Hot-Plugging With the `cfgadm` Command" on page 94.

SCSI Hot-Plugging With the `cfgadm` Command (Task Map)

Task	Description	For Instructions
1. Display information about SCSI devices	Display information about SCSI controllers and devices.	"How to Display Information About SCSI Devices" on page 84
2. Unconfigure a SCSI controller	Unconfigure a SCSI controller.	"How to Unconfigure a SCSI Controller" on page 85
3. Configure a SCSI controller	Configure a SCSI controller that was previously unconfigured.	"How to Configure a SCSI Controller" on page 85
4. Configure a SCSI device	Configure a specific SCSI device.	"How to Configure a SCSI Device" on page 86
5. Disconnect a SCSI controller	Disconnect a specific SCSI controller.	"How to Disconnect a SCSI Controller" on page 87
6. Connect a SCSI controller	Connect a specific SCSI controller that was previously disconnected.	"SPARC: How to Connect a SCSI Controller" on page 88
7. Add a SCSI device to a SCSI bus	Add a specific SCSI device to a SCSI bus.	"SPARC: How to Add a SCSI Device to a SCSI Bus" on page 88
8. Replace an identical device on a SCSI controller	Replace a device on the SCSI bus with another device of the same type.	"SPARC: How to Replace an Identical Device on a SCSI Controller" on page 89
9. Remove a SCSI device	Remove a SCSI device from the system.	"SPARC: How to Remove a SCSI Device" on page 90
10. Troubleshooting SCSI configuration problems	Resolve a failed SCSI unconfigure operation.	"How to Resolve a Failed SCSI Unconfigure Operation" on page 93

SCSI Hot-Plugging With the `cfgadm` Command

This section describes various SCSI hot-plugging procedures that you can perform with the `cfgadm` command.

These procedures use specific devices as examples to illustrate how to use the `cfgadm` command to hot-plug SCSI components. The device information that you supply, and that the `cfgadm` command displays, depends on your system configuration.

▼ How to Display Information About SCSI Devices

The following procedure uses SCSI controllers `c0` and `c1` and the devices that are attached to them as examples of the type of device configuration information that you can display with the `cfgadm` command.

Note – If the SCSI device is not supported by the `cfgadm` command, it does not display in the `cfgadm` command output.

Steps

1. **Become superuser.**

2. **Display information about attachment points on the system.**

```
# cfgadm -l
Ap_Id                Type         Receptacle   Occupant     Condition
c0                   scsi-bus     connected    configured   unknown
c1                   scsi-bus     connected    configured   unknown
```

   In this example, `c0` and `c1` represent two SCSI controllers.

3. **Display information about a system's SCSI controllers and their attached devices.**

```
# cfgadm -al
Ap_Id                Type         Receptacle   Occupant     Condition
c0                   scsi-bus     connected    configured   unknown
c0::dsk/c0t0d0       disk         connected    configured   unknown
c0::rmt/0            tape         connected    configured   unknown
c1                   scsi-bus     connected    configured   unknown
c1::dsk/c1t3d0       disk         connected    configured   unknown
c1::dsk/c1t4d0       unavailable  connected    unconfigured unknown
```

Note – The `cfgadm -l` commands displays information about SCSI HBAs but not SCSI devices. Use the `cfgadm -al` command to display information about SCSI devices such as disk and tapes.

▼ How to Unconfigure a SCSI Controller

The following procedure uses SCSI controller `c1` as an example of unconfiguring a SCSI controller.

Steps

1. **Become superuser.**
2. **Unconfigure a SCSI controller.**

   ```
   # cfgadm -c unconfigure c1
   ```

3. **Verify that the SCSI controller is unconfigured.**

   ```
   # cfgadm -al
   Ap_Id                 Type         Receptacle   Occupant     Condition
   c0                    scsi-bus     connected    configured   unknown
   c0::dsk/c0t0d0        disk         connected    configured   unknown
   c0::rmt/0             tape         connected    configured   unknown
   c1                    scsi-bus     connected    unconfigured unknown
   ```

   Notice that the `Occupant` column for `c1` specifies `unconfigured`, indicating that the SCSI bus has no configured occupants.

   If the unconfigure operation fails, see "How to Resolve a Failed SCSI Unconfigure Operation" on page 93.

▼ How to Configure a SCSI Controller

The following procedure uses SCSI controller `c1` as an example of configuring a SCSI controller.

Steps

1. **Become superuser.**
2. **Configure a SCSI controller.**

   ```
   # cfgadm -c configure c1
   ```

3. **Verify that the SCSI controller is configured.**

   ```
   # cfgadm -al
   Ap_Id                Type         Receptacle   Occupant     Condition
   c0                   scsi-bus     connected    configured   unknown
   ```

```
c0::dsk/c0t0d0       disk         connected    configured   unknown
c0::rmt/0            tape         connected    configured   unknown
c1                   scsi-bus     connected    configured   unknown
c1::dsk/c1t3d0       disk         connected    configured   unknown
c1::dsk/c1t4d0       unavailable  connected    unconfigured unknown
```

The previous unconfigure procedure removed all devices on the SCSI bus. Now all the devices are configured back into the system.

▼ How to Configure a SCSI Device

The following procedure uses SCSI disk `c1t4d0` as an example of configuring a SCSI device.

Steps

1. **Become superuser.**
2. **Identify the device to be configured.**

```
# cfgadm -al
Ap_Id                Type          Receptacle   Occupant     Condition
c0                   scsi-bus      connected    configured   unknown
c0::dsk/c0t0d0       disk          connected    configured   unknown
c0::rmt/0            tape          connected    configured   unknown
c1                   scsi-bus      connected    configured   unknown
c1::dsk/c1t3d0       disk          connected    configured   unknown
c1::dsk/c1t4d0       unavailable   connected    unconfigured unknown
```

3. **Configure the SCSI device.**

```
# cfgadm -c configure c1::dsk/c1t4d0
```

4. **Verify that the SCSI device is configured.**

```
# cfgadm -al
Ap_Id                Type          Receptacle   Occupant     Condition
c0                   scsi-bus      connected    configured   unknown
c0::dsk/c0t0d0       disk          connected    configured   unknown
c0::rmt/0            tape          connected    configured   unknown
c1                   scsi-bus      connected    configured   unknown
c1::dsk/c1t3d0       disk          connected    configured   unknown
c1::dsk/c1t4d0       disk          connected    configured   unknown
```

▼ How to Disconnect a SCSI Controller

Caution – Disconnecting a SCSI device must be done with caution, particularly when you are dealing with controllers for disks that contain critical file systems such as root (/), usr, var, and the swap partition. The dynamic reconfiguration software cannot detect all cases where a system hang might result. Use this procedure with caution.

The following procedure uses SCSI controller c1 as an example of disconnecting a SCSI device.

Steps

1. **Become superuser.**
2. **Verify that the device is connected before you disconnect it.**

```
# cfgadm -al
Ap_Id                Type          Receptacle   Occupant     Condition
c0                   scsi-bus      connected    configured   unknown
c0::dsk/c0t0d0       disk          connected    configured   unknown
c0::rmt/0            tape          connected    configured   unknown
c1                   scsi-bus      connected    configured   unknown
c1::dsk/c1t3d0       disk          connected    configured   unknown
c1::dsk/c1t4d0       disk          connected    configured   unknown
```

3. **Disconnect the SCSI controller.**

```
# cfgadm -c disconnect c1
WARNING: Disconnecting critical partitions may cause system hang.
Continue (yes/no)? y
```

Caution – This command suspends all I/O activity on the SCSI bus until the cfgadm -c connect command is used. The cfgadm command does some basic checking to prevent critical partitions from being disconnected, but it cannot detect all cases. Inappropriate use of this command can result in a system hang and could require a system reboot.

4. **Verify that the SCSI bus is disconnected.**

```
# cfgadm -al
Ap_Id                Type          Receptacle   Occupant     Condition
c0                   scsi-bus      connected    configured   unknown
c0::dsk/c0t0d0       disk          connected    configured   unknown
c0::rmt/0            tape          connected    configured   unknown
c1                   unavailable   disconnected configured   unknown
c1::dsk/c1t10d0      unavailable   disconnected configured   unknown
c1::dsk/c1t4d0       unavailable   disconnected configured   unknown
```

The controller and all the devices that are attached to it are disconnected from the system.

▼ SPARC: How to Connect a SCSI Controller

The following procedure uses SCSI controller `c1` as an example of connecting a SCSI controller.

Steps

1. **Become superuser.**

2. **Verify that the device is disconnected before you connect it.**

```
# cfgadm -al
Ap_Id                Type          Receptacle   Occupant     Condition
c0                   scsi-bus      connected    configured   unknown
c0::dsk/c0t0d0       disk          connected    configured   unknown
c0::rmt/0            tape          connected    configured   unknown
c1                   unavailable   disconnected configured   unknown
c1::dsk/c1t10d0      unavailable   disconnected configured   unknown
c1::dsk/c1t4d0       unavailable   disconnected configured   unknown
```

3. **Connect the SCSI controller.**

```
# cfgadm -c connect c1
```

4. **Verify that the SCSI controller is connected.**

```
# cfgadm -al
Ap_Id                Type          Receptacle   Occupant     Condition
c0                   scsi-bus      connected    configured   unknown
c0::dsk/c0t0d0       disk          connected    configured   unknown
c0::rmt/0            tape          connected    configured   unknown
c1                   scsi-bus      connected    configured   unknown
c1::dsk/c1t3d0       disk          connected    configured   unknown
c1::dsk/c1t4d0       disk          connected    configured   unknown
```

▼ SPARC: How to Add a SCSI Device to a SCSI Bus

SCSI controller `c1` provides an example of how to add a SCSI device to a SCSI bus.

Note – When you add devices, you specify the `Ap_Id` of the SCSI HBA (controller) to which the device is attached, not the `Ap_Id` of the device itself.

Steps

1. **Become superuser.**

2. **Identify the current SCSI configuration.**

```
# cfgadm -al
Ap_Id                Type          Receptacle   Occupant     Condition
c0                   scsi-bus      connected    configured   unknown
c0::dsk/c0t0d0       disk          connected    configured   unknown
```

```
c0::rmt/0             tape          connected    configured   unknown
c1                    scsi-bus      connected    configured   unknown
c1::dsk/c1t3d0        disk          connected    configured   unknown
```

3. **Add the SCSI device to the SCSI bus.**

   ```
   # cfgadm -x insert_device c1
   Adding device to SCSI HBA: /devices/sbus@1f,0/SUNW,fas@1,8800000
   This operation will suspend activity on SCSI bus: c1
   ```

   a. **Type `y` at the `Continue (yes/no)?` prompt to proceed.**

      ```
      Continue (yes/no)? y
      SCSI bus quiesced successfully.
      It is now safe to proceed with hotplug operation.
      ```

      I/O activity on the SCSI bus is suspended while the hot-plug operation is in progress.

   b. **Connect the device and then power it on.**

   c. **Type `y` at the `Enter y if operation is complete or n to abort (yes/no)?` prompt.**

      ```
      Enter y if operation is complete or n to abort (yes/no)? y
      ```

4. **Verify that the device has been added.**

   ```
   # cfgadm -al
   Ap_Id                 Type          Receptacle   Occupant     Condition
   c0                    scsi-bus      connected    configured   unknown
   c0::dsk/c0t0d0        disk          connected    configured   unknown
   c0::rmt/0             tape          connected    configured   unknown
   c1                    scsi-bus      connected    configured   unknown
   c1::dsk/c1t3d0        disk          connected    configured   unknown
   c1::dsk/c1t4d0        disk          connected    configured   unknown
   ```

   A new disk has been added to controller `c1`.

▼ SPARC: How to Replace an Identical Device on a SCSI Controller

The following procedure uses SCSI disk `c1t4d0` as an example of replacing an identical device on a SCSI controller.

Steps

1. **Become superuser.**

2. **Identify the current SCSI configuration.**

   ```
   # cfgadm -al
   Ap_Id                 Type          Receptacle   Occupant     Condition
   c0                    scsi-bus      connected    configured   unknown
   ```

```
c0::dsk/c0t0d0       disk         connected    configured   unknown
c0::rmt/0            tape         connected    configured   unknown
c1                   scsi-bus     connected    configured   unknown
c1::dsk/c1t3d0       disk         connected    configured   unknown
c1::dsk/c1t4d0       disk         connected    configured   unknown
```

3. **Replace a device on the SCSI bus with another device of the same type.**

```
# cfgadm -x replace_device c1::dsk/c1t4d0
Replacing SCSI device: /devices/sbus@1f,0/SUNW,fas@1,8800000/sd@4,0
This operation will suspend activity on SCSI bus: c1
```

   a. **Type `y` at the `Continue (yes/no)?` prompt to proceed.**

   I/O activity on the SCSI bus is suspended while the hot-plug operation is in progress.

```
Continue (yes/no)? y
SCSI bus quiesced successfully.
It is now safe to proceed with hotplug operation.
```

   b. **Power off the device to be removed and remove it.**

   c. **Add the replacement device. Then, power it on.**

   The replacement device should be of the same type and at the same address (target and lun) as the device to be removed

   d. **Type `y` at the `Enter y if operation is complete or n to abort (yes/no)?` prompt.**

```
Enter y if operation is complete or n to abort (yes/no)? y
```

4. **Verify that the device has been replaced.**

```
# cfgadm -al
Ap_Id                Type         Receptacle   Occupant     Condition
c0                   scsi-bus     connected    configured   unknown
c0::dsk/c0t0d0       disk         connected    configured   unknown
c0::rmt/0            tape         connected    configured   unknown
c1                   scsi-bus     connected    configured   unknown
c1::dsk/c1t3d0       disk         connected    configured   unknown
c1::dsk/c1t4d0       disk         connected    configured   unknown
```

▼ SPARC: How to Remove a SCSI Device

The following procedure uses SCSI disk `c1t4d0` as an example of removing a device on a SCSI controller.

Steps

1. **Become superuser.**

2. **Identify the current SCSI configuration.**

```
# cfgadm -al
Ap_Id                 Type         Receptacle   Occupant     Condition
c0                    scsi-bus     connected    configured   unknown
c0::dsk/c0t0d0        disk         connected    configured   unknown
c0::rmt/0             tape         connected    configured   unknown
c1                    scsi-bus     connected    configured   unknown
c1::dsk/c1t3d0        disk         connected    configured   unknown
c1::dsk/c1t4d0        disk         connected    configured   unknown
```

3. **Remove the SCSI device from the system.**

```
# cfgadm -x remove_device c1::dsk/c1t4d0
Removing SCSI device: /devices/sbus@1f,0/SUNW,fas@1,8800000/sd@4,0
This operation will suspend activity on SCSI bus: c1
```

   a. **Type `y` at the `Continue (yes/no)?` prompt to proceed.**

   ```
   Continue (yes/no)? y
   SCSI bus quiesced successfully.
   It is now safe to proceed with hotplug operation.
   ```

   I/O activity on the SCSI bus is suspended while the hot-plug operation is in progress.

   b. **Power off the device to be removed and remove it.**

   c. **Type `y` at the `Enter y if operation is complete or n to abort (yes/no)?` prompt.**

   ```
   Enter y if operation is complete or n to abort (yes/no)? y
   ```

4. **Verify that the device has been removed from the system.**

```
# cfgadm -al
Ap_Id                 Type         Receptacle   Occupant     Condition
c0                    scsi-bus     connected    configured   unknown
c0::dsk/c0t0d0        disk         connected    configured   unknown
c0::rmt/0             tape         connected    configured   unknown
c1                    scsi-bus     connected    configured   unknown
c1::dsk/c1t3d0        disk         connected    configured   unknown
```

SPARC: Troubleshooting SCSI Configuration Problems

This section provides error messages and possible solutions for troubleshooting SCSI configuration problems. For more information on troubleshooting SCSI configuration problems, see `cfgadm`(1M).

Error Message

```
cfgadm: Component system is busy, try again: failed to offline:
     device path
          Resource              Information
```

```
        ------------------  -------------------------
        /dev/dsk/c1t0d0s0   mounted filesystem "/file-system"
```

Cause
: You attempted to remove or replace a device with a mounted file system.

Solution
: Unmount the file system that is listed in the error message and retry the `cfgadm` operation.

If you use the `cfgadm` command to remove a system resource, such as a swap device or a dedicated dump device, an error messages similar to the following is displayed if the system resource is still active.

Error Message

```
cfgadm: Component system is busy, try again: failed to offline:
     device path
          Resource              Information
     ------------------  -------------------------
     /dev/dsk/device-name   swap area
```

Cause
: You attempted to remove or replace one or more configured swap areas.

Solution
: Unconfigure the swap areas on the device that is specified and retry the `cfgadm` operation.

Error Message

```
cfgadm: Component system is busy, try again: failed to offline:
     device path
          Resource              Information
     ------------------  -------------------------
     /dev/dsk/device-name   dump device (swap)
```

Cause
: You attempted to remove or replace a dump device that is configured on a swap area.

Solution
: Unconfigure the dump device that is configured on the swap area and retry the `cfgadm` operation.

Error Message

```
cfgadm: Component system is busy, try again: failed to offline:
     device path
          Resource              Information
     ------------------  -------------------------
     /dev/dsk/device-name   dump device (dedicated)
```

Cause
: You attempted to remove or replace a dedicated dump device.

Solution
  Unconfigure the dedicate dump device and retry the `cfgadm` operation.

▼ How to Resolve a Failed SCSI Unconfigure Operation

Use this procedure if one or more target devices are busy and the SCSI unconfigure operation fails. Otherwise, future dynamic reconfiguration operations on this controller and target devices will fail with a `dr in progress` message.

Steps

1. **Become superuser, if not done already.**
2. **Type the following command to reconfigure the controller.**

```
# cfgadm -c configure device-name
```

PCI Hot-Plugging With the `cfgadm` Command (Task Map)

Task	Description	For Instructions
1. Display PCI Slot Configuration Information	Display the status of PCI hot-pluggable devices and slots on the system.	"x86: How to Display PCI Slot Configuration Information" on page 94
2. Remove a PCI adapter card	Unconfigure the card, disconnect power from the slot, and remove the card from the system.	"x86: How to Remove a PCI Adapter Card" on page 95
3. Add a PCI adapter card	Insert the adapter card into a hot-pluggable slot, connect power to the slot, and configure the card.	"x86: How to Add a PCI Adapter Card" on page 96
4. Troubleshooting PCI configuration problems	Identify error message and possible solutions to resolve PCI configuration problems.	"x86: Troubleshooting PCI Configuration Problems" on page 97

x86: PCI Hot-Plugging With the `cfgadm` Command

This section provides step-by-step instructions for hot-plugging PCI adapter cards on x86 based systems.

In the examples, only PCI attachment points are listed, for brevity. The attachment points that are displayed on your system depend on your system configuration.

▼ x86: How to Display PCI Slot Configuration Information

The `cfgadm` command displays the status of PCI hot-pluggable devices and slots on a system. For more information, see `cfgadm`(1M).

Steps

1. **Become superuser.**
2. **Display PCI slot configuration information.**

```
# cfgadm
Ap_Id                Type         Receptacle   Occupant     Condition
pci1:hpc0_slot0      unknown      empty        unconfigured unknown
pci1:hpc0_slot1      unknown      empty        unconfigured unknown
pci1:hpc0_slot2      unknown      empty        unconfigured unknown
pci1:hpc0_slot3      ethernet/hp  connected    configured   ok
pci1:hpc0_slot4      unknown      empty        unconfigured unknown
```

Display specific PCI device information.

```
# cfgadm -s "cols=ap_id:type:info" pci
Ap_Id                Type         Information
pci1:hpc0_slot0      unknown      Slot 7
pci1:hpc0_slot1      unknown      Slot 8
pci1:hpc0_slot2      unknown      Slot 9
pci1:hpc0_slot3      ethernet/hp  Slot 10
pci1:hpc0_slot4      unknown      Slot 11
```

The logical `Ap_Id`, `pci1:hpc0_slot0`, is the logical `Ap_Id` for hot-pluggable slot, `Slot 7`. The component `hpc0` indicates the hot-pluggable adapter card for this slot, and `pci1` indicates the PCI bus instance. The `Type` field indicates the type of PCI adapter card that is present in the slot.

▼ x86: How to Remove a PCI Adapter Card

Steps

1. **Become superuser.**

2. **Determine which slot the PCI adapter card is in.**

```
# cfgadm
Ap_Id                Type         Receptacle   Occupant     Condition
pci1:hpc0_slot0      unknown      empty        unconfigured unknown
pci1:hpc0_slot1      unknown      empty        unconfigured unknown
pci1:hpc0_slot2      unknown      empty        unconfigured unknown
pci1:hpc0_slot3      ethernet/hp  connected    configured   ok
pci1:hpc0_slot4      unknown      empty        unconfigured unknown
```

3. **Stop the application that has the device open.**

   For example, if the device is an Ethernet card, use the `ifconfig` command to bring down the interface and unplumb the interface.

4. **Unconfigure the device.**

```
# cfgadm -c unconfigure pci1:hpc0_slot3
```

5. **Confirm that the device has been unconfigured.**

```
# cfgadm
Ap_Id                Type         Receptacle   Occupant     Condition
pci1:hpc0_slot0      unknown      empty        unconfigured unknown
pci1:hpc0_slot1      unknown      empty        unconfigured unknown
pci1:hpc0_slot2      unknown      empty        unconfigured unknown
pci1:hpc0_slot3      ethernet/hp  connected    unconfigured unknown
pci1:hpc0_slot4      unknown      empty        unconfigured unknown
```

6. **Disconnect the power to the slot.**

```
# cfgadm -c disconnect pci1:hpc0_slot3
```

7. **Confirm that the device has been disconnected.**

```
# cfgadm
Ap_Id                Type         Receptacle   Occupant     Condition
pci1:hpc0_slot0      unknown      empty        unconfigured unknown
pci1:hpc0_slot1      unknown      empty        unconfigured unknown
pci1:hpc0_slot2      unknown      empty        unconfigured unknown
pci1:hpc0_slot3      ethernet/hp  disconnected unconfigured unknown
pci1:hpc0_slot4      unknown      empty        unconfigured unknown
```

8. **Open the slot latches and remove the PCI adapter card.**

▼ x86: How to Add a PCI Adapter Card

Steps

1. **Become superuser.**
2. **Identify the hot-pluggable slot and open latches.**
3. **Insert the PCI adapter card into a hot-pluggable slot.**
4. **Determine which slot the PCI adapter card is in once it is inserted. Close the latches.**

```
# cfgadm
Ap_Id                Type         Receptacle   Occupant     Condition
pci1:hpc0_slot0      unknown      empty        unconfigured unknown
pci1:hpc0_slot1      unknown      empty        unconfigured unknown
pci1:hpc0_slot2      unknown      empty        unconfigured unknown
pci1:hpc0_slot3      ethernet/hp  disconnected unconfigured unknown
pci1:hpc0_slot4      unknown      empty        unconfigured unknown
```

5. **Connect the power to the slot.**

```
# cfgadm -c connect pci1:hpc0_slot3
```

6. **Confirm that the slot is connected.**

```
# cfgadm
Ap_Id                Type         Receptacle   Occupant     Condition
pci1:hpc0_slot0      unknown      empty        unconfigured unknown
pci1:hpc0_slot1      unknown      empty        unconfigured unknown
pci1:hpc0_slot2      unknown      empty        unconfigured unknown
pci1:hpc0_slot3      ethernet/hp  connected    unconfigured unknown
pci1:hpc0_slot4      unknown      empty        unconfigured unknown
```

7. **Configure the PCI adapter card.**

```
# cfgadm -c configure pci1:hpc0_slot3
```

8. **Verify the configuration of the PCI adapter card in the slot.**

```
# cfgadm
Ap_Id                Type         Receptacle   Occupant     Condition
pci1:hpc0_slot0      unknown      empty        unconfigured unknown
pci1:hpc0_slot1      unknown      empty        unconfigured unknown
pci1:hpc0_slot2      unknown      empty        unconfigured unknown
pci1:hpc0_slot3      ethernet/hp  connected    configured   unknown
pci1:hpc0_slot4      unknown      empty        unconfigured unknown
```

9. **Configure any supporting software if this device is a new device.**

   For example, if this device is an Ethernet card, use the `ifconfig` command to set up the interface.

x86: Troubleshooting PCI Configuration Problems

Error Message

```
cfgadm: Configuration operation invalid: invalid transition
```

Cause
An invalid transition was attempted.

Solution
Check whether the `cfgadm -c` command was issued appropriately. Use the `cfgadm` command to check the current receptacle and occupant state and to make sure that the `Ap_Id` is correct.

Error Message

```
cfgadm: Attachment point not found
```

Cause
The specified attachment point was not found.

Solution
Check whether the attachment point is correct. Use the `cfgadm` command to display a list of available attachment points. Also check the physical path to see if the attachment point is still there.

Note – In addition to the `cfgadm` command, several other commands are helpful during hot-pluggable operations. The `prtconf` command displays whether Solaris recognizes the hardware. After adding hardware, use the `prtconf` command to verify that the hardware is recognized. After a configure operation, use the `prtconf -D` command to verify that the driver is attached to the newly installed hardware device.

Reconfiguration Coordination Manager (RCM) Script Overview

The Reconfiguration Coordination Manager (RCM) is the framework that manages the dynamic removal of system components. By using RCM, you can register and release system resources in an orderly manner.

You can use the new RCM script feature to write your own scripts to shut down your applications, or to cleanly release the devices from your applications during dynamic reconfiguration. The RCM framework launches a script automatically in response to a reconfiguration request, if the request impacts the resources that are registered by the script.

You can also release resources from applications manually before you could dynamically remove the resource. Or, you could use the `cfgadm` command with the `-f` option to force a reconfiguration operation, but this option might leave your applications in an unknown state. Also, the manual release of resources from applications commonly causes errors.

The RCM script feature simplifies and better controls the dynamic reconfiguration process. By creating an RCM script, you can do the following:

- Automatically release a device when you dynamically remove a device. This process also closes the device if the device is opened by an application.
- Run site-specific tasks when you dynamically remove a device from the system.

What Is an RCM Script?

An RCM script is as follows:

- An executable shell script (Perl, `sh`, `csh`, or `ksh`) or binary program that the RCM daemon runs. Perl is the recommended language.
- A script that runs in its own address space by using the user ID of the script file owner.
- A script that is run by the RCM daemon when you use the `cfgadm` command to dynamically reconfigure a system resource.

What Can an RCM Script Do?

You can use an RCM script to release a device from an application when you dynamically remove a device. If the device is currently open, the RCM script also closes the device.

For example, an RCM script for a tape backup application can inform the tape backup application to close the tape drive or shut down the tape backup application.

How Does the RCM Script Process Work?

You can invoke a script as follows:

```
$ script-name command [args ...]
```

A script performs the following basic steps:

1. Takes the RCM command from command-line arguments.
2. Executes the command.
3. Writes the results to `stdout` as name-value pairs.

4. Exits with the appropriate exit status.

The RCM daemon runs one instance of a script at a time. For example, if a script is running, the RCM daemon does not run the same script until the first script exits.

RCM Script Commands

You must include the following RCM commands in an RCM script:

- `scriptinfo` - Gathers script information
- `register` - Registers interest in resources
- `resourceinfo` - Gathers resource information

You might include some or all of the following RCM commands:

- `queryremove` - Queries whether the resource can be released
- `preremove` - Releases the resource
- `postremove` - Provides post-resource removal notification
- `undoremove` - Undoes the actions done in `preremove`

For a complete description of these RCM commands, see `rcmscript`(4).

RCM Script Processing Environment

When you dynamically remove a device, the RCM daemon runs the following:

- The script's `register` command to gather the list of resources (device names) that are identified in the script.
- The script's `queryremove/preremove` commands prior to removing the resource if the script's registered resources are affected by the dynamic remove operation.
- The script's `postremove` command if the remove operation succeeds. However, if the remove operation fails, the RCM daemon runs the script's `undoremove` command.

RCM Script Tasks

The following sections describe the RCM script tasks for application developers and system administrators.

Application Developer RCM Script (Task Map)

The following task map describes the tasks for an application developer who is creating an RCM script.

Task	Description	For Instructions
1. Identify resources your application uses	Identify the resources (device names) your application uses that you could potentially dynamically remove.	`cfgadm`(1M)
2. Identify commands to release the resource	Identify the commands for notifying the application to cleanly release the resource from the application.	Application documentation
3. Identify commands for post-removal of the resource	Include the commands for notifying the application of the resource removal.	`rcmscript`(4)
4. Identify commands if the resource removal fails	Include the commands for notifying the application of the available resource.	`rcmscript`(4)
5. Write the RCM script	Write the RCM script based on the information identified in the previous tasks.	"Tape Backup RCM Script Example" on page 103
6. Install the RCM script	Add the script to the appropriate script directory.	"How to Install an RCM Script" on page 102
7. Test the RCM script	Test the script by running the script commands manually and by initiating a dynamic reconfiguration operation.	"How to Test an RCM Script" on page 102

System Administrator RCM Script (Task Map)

The following task map describes the tasks for a system administrator who is creating an RCM script to do site customization.

Task	Description	For Instructions
1. Identify resources to be dynamically removed	Identify the resources (device names) to be potentially removed by using the `cfgadm -l` command.	`cfgadm`(1M)

Task	Description	For Instructions
2. Identify applications to be stopped	Identify the commands for stopping the applications cleanly.	Application documentation
3. Identify commands for pre-removal and post-removal of the resource	Identify the actions to be taken before and after the resource is removed.	`rcmscript`(4)
4. Write the RCM script	Write the RCM script based on the information identified in the previous tasks.	"Tape Backup RCM Script Example" on page 103
5. Install the RCM script	Add the script to the appropriate script directory.	"How to Install an RCM Script" on page 102
6. Test the RCM script	Test the script by running the script commands manually and by initiating a dynamic reconfiguration operation.	"How to Test an RCM Script" on page 102

Naming an RCM Script

A script must be named as *vendor*,*service* where the following applies:

vendor	Is the stock symbol of the vendor that provides the script, or any distinct name that identifies the vendor.
service	Is the name of the service that the script represents.

Installing or Removing an RCM Script

You must be superuser (root) to install or remove an RCM script. Use this table to determine where you should install your RCM script.

TABLE 6–1 RCM Script Directories

Directory Location	Script Type
`/etc/rcm/scripts`	Scripts for specific systems
`/usr/platform/‘uname -i‘/lib/rcm/scripts`	Scripts for a specific hardware implementation
`/usr/platform/‘uname -m‘/lib/rcm/scripts`	Scripts for a specific hardware class
`/usr/lib/rcm/scripts`	Scripts for any hardware

▼ How to Install an RCM Script

Steps

1. **Become superuser.**
2. **Copy the script to the appropriate directory as described in Table 6–1.**

   For example:

   ```
   # cp SUNW,sample.pl /usr/lib/rcm/scripts
   ```

3. **Change the user ID and the group ID of the script to the desired values.**

   ```
   # chown user:group /usr/lib/rcm/scripts/SUNW,sample.pl
   ```

4. **Send `SIGHUP` to the RCM daemon.**

   ```
   # pkill -HUP -x -u root rcm_daemon
   ```

▼ How to Remove an RCM Script

Steps

1. **Become superuser.**
2. **Remove the script from the RCM script directory.**

   For example:

   ```
   # rm /usr/lib/rcm/scripts/SUNW,sample.pl
   ```

3. **Send `SIGHUP` to the RCM daemon.**

   ```
   # pkill -HUP -x -u root rcm_daemon
   ```

▼ How to Test an RCM Script

Steps

1. **Set environment variables, such as `RCM_ENV_FORCE`, in the command-line shell before running your script.**

   For example, in the Korn shell, use the following:

   ```
   $ export RCM_ENV_FORCE=TRUE
   ```

2. **Test the script by running the script commands manually from the command line.**

   For example:

   ```
   $ script-name scriptinfo
   $ script-name register
   $ script-name preremove resource-name
   ```

```
$ script-name postremove resource-name
```

3. **Make sure each RCM script command in your script prints appropriate output to `stdout`.**
4. **Install the script in the appropriate script directory.**

   For more information, see "How to Install an RCM Script" on page 102.
5. **Test the script by initiating a dynamic remove operation:**

   For example, assume your script registers the device, `/dev/dsk/c1t0d0s0`. Try these commands.

   ```
   $ cfgadm -c unconfigure c1::dsk/c1t0d0
   $ cfgadm -f -c unconfigure c1::dsk/c1t0d0
   $ cfgadm -c configure c1::dsk/c1t0d0
   ```

Caution – Make sure that you are familiar with these commands because they can alter the state of the system and can cause system failures.

Tape Backup RCM Script Example

This example illustrates how to use an RCM script for tape backups.

What the Tape Backup RCM Script Does

The tape backup RCM script performs the following steps:

1. Sets up a dispatch table of RCM commands.
2. Calls the dispatch routine that corresponds to the specified RCM command and exits with status 2 for unimplemented RCM commands.
3. Sets up the `scriptinfo` section:

   ```
   rcm_script_func_info=Tape backup appl script for DR
   ```
4. Registers all tape drives in the system by printing all tape drive device names to `stdout`.

   ```
   rcm_resource_name=/dev/rmt/$f
   ```

   If an error occurs, prints the error information to `stdout`.

   ```
   rcm_failure_reason=$errmsg
   ```
5. Sets up the resource information for the tape device.

   ```
   rcm_resource_usage_info=Backup Tape Unit Number $unit
   ```

6. Sets up the `preremove` information by checking if the backup application is using the device. If the backup application is not using the device, the dynamic reconfiguration operation continues. If the backup application is using the device, the script checks `RCM_ENV_FORCE`. If `RCM_ENV_FORCE` is set to `FALSE`, the script denies the dynamic reconfiguration operation and prints the following message:

   ```
   rcm_failure_reason=tape backup in progress pid=...
   ```

   If `RCM_ENV_FORCE` is set to `TRUE`, the backup application is stopped, and the reconfiguration operation proceeds.

Outcomes of the Tape Backup Reconfiguration Scenarios

Here are the various outcomes if you use the `cfgadm` command to remove a tape device without the RCM script.

- If you use the `cfgadm` command and the backup application is not using the tape device, the operation succeeds.
- If you use the `cfgadm` command and the backup application is using the tape device, the operation fails.

Here are the various outcomes if you use the `cfgadm` command to remove a tape device with the RCM script.

- If you use the `cfgadm` command and the backup application is not using the tape device, the operation succeeds.
- If you use the `cfgadm` command without the `-f` option and the backup application is using the tape device, the operation fails with an error message similar to the following:

  ```
  tape backup in progress pid=...
  ```

- If you use the `cfgadm -f` command and the backup application is using the tape device, the script stops the backup application and the `cfgadm` operation succeeds.

Example—Tape Backup RCM Script

```
#! /usr/bin/perl -w
   #
   # A sample site customization RCM script.
   #
   # When RCM_ENV_FORCE is FALSE this script indicates to RCM that it cannot
   # release the tape drive when the tape drive is being used for backup.
   #
   # When RCM_ENV_FORCE is TRUE this script allows DR removing a tape drive
   # when the tape drive is being used for backup by killing the tape
   # backup application.
   #
```

```
    use strict;

    my ($cmd, %dispatch);
    $cmd = shift(@ARGV);
# dispatch table for RCM commands
    %dispatch = (
            "scriptinfo"    =>      \&do_scriptinfo,
            "register"      =>      \&do_register,
            "resourceinfo"  =>      \&do_resourceinfo,
            "queryremove"   =>      \&do_preremove,
            "preremove"     =>      \&do_preremove
    );


    if (defined($dispatch{$cmd})) {
            &{$dispatch{$cmd}};
    } else {
            exit (2);
    }

    sub do_scriptinfo
    {
            print "rcm_script_version=1\n";
            print "rcm_script_func_info=Tape backup appl script for DR\n";
            exit (0);
    }

    sub do_register
{
            my ($dir, $f, $errmsg);

            $dir = opendir(RMT, "/dev/rmt");
            if (!$dir) {
                 $errmsg = "Unable to open /dev/rmt directory: $!";
                 print "rcm_failure_reason=$errmsg\n";
                 exit (1);
            }

            while ($f = readdir(RMT)) {
                # ignore hidden files and multiple names for the same device
                if (($f !~ /^\./) && ($f =~ /^[0-9]+$/)) {
                        print "rcm_resource_name=/dev/rmt/$f\n";
                    }

            }

            closedir(RMT);
            exit (0);
    }
sub do_resourceinfo
    {
      my ($rsrc, $unit);

      $rsrc = shift(@ARGV);
      if ($rsrc =~ /^\/dev\/rmt\/([0-9]+)$/) {
           $unit = $1;
           print "rcm_resource_usage_info=Backup Tape Unit Number $unit\n";
           exit (0);
       } else {
           print "rcm_failure_reason=Unknown tape device!\n";
            exit (1);
        }
    }

    sub do_preremove
    {
            my ($rsrc);

            $rsrc = shift(@ARGV);

            # check if backup application is using this resource
            #if (the backup application is not running on $rsrc) {
                    # allow the DR to continue
            #        exit (0);
            #}
            #
            # If RCM_ENV_FORCE is FALSE deny the operation.
            # If RCM_ENV_FORCE is TRUE kill the backup application in order
            # to allow the DR operation to proceed
            #
            if ($ENV{RCM_ENV_FORCE} eq 'TRUE') {
                 if ($cmd eq 'preremove') {
                         # kill the tape backup application
                 }
                 exit (0);
            } else {
               #
               # indicate that the tape drive can not be released
               # since the device is being used for backup by the
               # tape backup application
               #
               print "rcm_failure_reason=tape backup in progress pid=...\n"
;
               exit (3);
            }
    }
```

CHAPTER 7

Using USB Devices (Overview)

This chapter provides an overview of Universal Serial Bus (USB) devices in the Solaris environment.

This is a list of the overview information in this chapter.

- "What's New in USB Devices?" on page 107
- "Overview of USB Devices" on page 114
- "About USB in the Solaris Environment" on page 118

For step-by-step instructions on using USB devices in the Solaris environment, see Chapter 8.

For general information about dynamic reconfiguration and hot-plugging, see Chapter 6.

For information on configuring USB printers, see "What's New in Printing?" in *System Administration Guide: Advanced Administration*.

What's New in USB Devices?

The following sections describe USB device enhancements in this Solaris release.

USB Dual Framework

The USBA framework, found in the Solaris 9 12/03 release, was originally developed for USB 1.1 devices. A new framework, called USBA 1.0, was created to meet more demanding requirements of USB 2.0 devices. The framework operates USB 1.1 devices as well. This Solaris release provides both frameworks, hence the name *dual framework*.

The purpose of the dual framework is to facilitate a smoother transition from the original framework to the newer framework. The original USBA framework operates devices connected to a system's USB 1.1 ports, while the new USBA 1.0 framework operates devices connected to a system's USB 2.0 ports. All Sun motherboard ports are USB 1.1 ports, while most PCI card ports support USB 2.0.

For specific details on the how the USB dual framework works, go to `http://www.sun.com/desktop/whitepapers.html`.

USB Framework Compatibility Issues

A driver written for one USB framework will not work on the other USB framework. Most Sun-supplied USB drivers provide versions for both frameworks.

Compatibility problems might occur if you attempt to plug a USB device into a port, directed by a framework that does not recognize a proper driver for that device because the driver is incompatible. When a framework tries to attach a framework-incompatible driver for a device, you will see console messages similar to the following:

```
The driver for device binding name is not for USBA1.0
```

This message will appear, for example, when a device operated by a non-Sun driver, which is compatible with USBA 1.0 framework, is plugged into a port supported by the original USBA framework. The USBA 1.0 framework recognizes the device and tries to map the correct driver, but the driver is rejected because it is incompatible with the framework operating the port.

For information on identifying your USB framework configuration, see "How to Display USB Device Information (`prtconf`)" on page 128.

Solaris Support for USB Devices

The following table describes Solaris support for USB 1.1 and USB 2.0 devices:

	Solaris 8 HW* Releases	Solaris 9 Releases
USB 1.1	SPARC and x86	SPARC and x86
USB 1.1 audio devices	Not supported on a USB 2.0 hub	Not supported on a USB 2.0 hub
USB 2.0	SPARC	SPARC and x86 (Solaris 9 4/04)
USB 2.0 audio devices	Not supported	Not supported

	Solaris 8 HW* Releases	Solaris 9 Releases
USB 2.0 storage devices	Supported on a USB 2.0 hub	Supported on a USB 2.0 hub (Solaris 9 4/04)

*This is not the Solaris 8 releases, but the Solaris 8 HW releases, starting with the Solaris 8 HW 5/03 release. The patch number for the USB dual framework found in the Solaris 8 HW 5/03 release is 109896.

Note – In the Solaris 9 9/04 release only, USB 1.1 devices will operate on USB 2.0 hubs that are connected to 2.0 ports.

The following table provides a summary of USB support on Sun SPARC hardware:

System Type	Solaris Releases	USB Device and Speed Support
Sun Blade 100, 150, 1000, and 2000	Solaris 9 releases, before the Solaris 9 4/04 release, and Solaris 8 releases before the Solaris HW 5/03 release	All USB devices at 12 Mb/sec
Sun Blade 100, 150, 1000, and 2000	Solaris 9 4/04 and Solaris 8 HW 5/03	USB 1.1 devices at 12 Mb/sec (connected to any USB ports)
		USB 2.0 devices at 12 Mb/sec (connected to motherboard ports)
		USB 2.0 devices at 480 Mb/sec (connected to ports on add-on PCI USB 2.0 card)
Sun Blade 1500 and 2500	Solaris 9 4/04 and Solaris 8 HW 5/03	USB 1.1 devices at 12 Mb/sec (connected to any USB ports)
		USB 2.0 devices at 12 Mb/sec (connected to motherboard ports)
		USB 2.0 devices at 480 Mb/sec (connected to ports on PCI combo card)
Other Sun SPARC PCI platforms	Solaris 9 4/04 and Solaris 8 HW 5/03	USB 1.1 devices at 12 Mb/sec
		USB 2.0 devices at 480 Mb/sec (connected to ports on add-on PCI USB 2.0 card)

For information about PCI cards verified on the Solaris release, go to: `http://www.sun.com/io_technologies/USB.html`

Sun Microsystems platforms that provide support for USB devices include the following:

- SPARC based systems with OHCI host controllers that support USB 1.1 provide low- and full-speed devices:
  - Sun Blade™ systems that run the Solaris 8 or 9 releases.
  - Netra™ X 1/T1 and some Sun Fire™ systems that run the Solaris 9 release.
- SPARC based systems with OHCI and EHCI host controllers, such as the Sun Blade 1500 or 2500 systems, provide high-speed support for USB 2.0 devices and low- and full-speed support for USB 1.1 devices. Systems include any PCI-based sun4u system that run the Solaris 8 HW 5/03 release or Solaris 9 4/04 release, including the systems listed above when they are equipped with a USB 2.0 PCI card.
- x86 based systems that run the Solaris 8 or 9 x86 Platform Edition with UHCI host controllers provide USB 1.1 support.

For additional USB support information, see "Overview of USB Devices" on page 114.

SPARC: USB 2.0 Features

This Solaris release includes the following USB 2.0 features:

- **Better performance** – Increased data throughput for devices connected to USB 2.0 controllers, up to 40 times faster than USB 1.1 devices.

  You will be able to take advantage of the high-speed USB protocol when accessing high-speed mass storage devices, such as DVDs and hard drives.
- **Compatibility** – Backward compatibility with 1.0 and 1.1 devices and drivers so that you can use the same cables, connectors, and software interfaces.

For a description of USB devices and terminology, see "Overview of USB Devices" on page 114.

USB 2.0 Device Features and Compatibility Issues

USB 2.0 devices are defined as high-speed devices that follow the USB 2.0 specification. You can refer to the USB 2.0 specification at http://www.usb.org.

Some of the USB devices that are supported on SPARC based and x86 based systems in this Solaris release are as follows:

- Mass storage devices – CD-RWs, hard disks, DVD, digital cameras, Zip, diskettes, and tape drives
- Keyboard, mouse devices, speakers and microphones
- Audio devices

For a full listing of USB devices that have been verified on the Solaris release, go to:

`http://www.sun.com/io_technologies/USB.html`

Additional storage devices might work by modifying the `scsa2usb.conf` file. For more information, see the `scsa2usb(7D)` man page.

Solaris USB 2.0 device support includes the following features:

- Increased USB bus speed from 12 Mbps to 480 Mbps. This increase means devices that support the USB 2.0 specification can run significantly faster than their USB 1.1 counterparts, when they are connected to a USB 2.0 port.

  A USB 2.0 port is defined on SPARC systems as follows:

  - A port on a USB 2.0 PCI card
  - A port on a USB 2.0 hub that is connected to USB 2.0 port
- USB 2.0 is Solaris Ready on all PCI-based SPARC platforms. A USB 2.0 PCI card is needed to provide USB 2.0 ports. For a list of USB 2.0 PCI cards that have been verified for the Solaris release, go to `http://www.sun.com/io_technologies/USB.html`.
- USB 1.1 devices work as they have in the past, even if you have both USB 1.1 and USB 2.0 devices on the same system.
- While USB 2.0 devices operate on a USB 1.x port, their performance is significantly better when connected to a USB 2.0 port.
- A USB 2.0 host controller has one high-speed Enhanced Host Controller (EHCI) and one or more low- or full-speed OpenHCI Host Controller (OHCI) embedded controllers. Devices connected to a USB 2.0 port are dynamically assigned to either an EHCI or OHCI controller, depending on whether or not they support USB 2.0.

Note – USB 2.0 storage devices connected to a port on a USB 2.0 PCI card, and that were used with a prior Solaris release in the same hardware configuration, can change device names after upgrading to this release. This change occurs because these devices are now seen as USB 2.0 devices and are taken over by the EHCI controller. The controller number, *w* in `/dev/[`*r*`]dsk/c`*w*`t`*x*`d`*y*`s`*z*, is changed for these devices.

For more information on USB 2.0 device support, see the `ehci(7D)` and `usba(7D)` man pages.

USB 2.0 Cables

- Maximum cable length supported is 5 meters.
- Do not use cable extenders. For best results, use a self-powered hub to extend cable length.
- For more information, go to `http://www.usb.org/channel/training/warning/`

Bus-Powered Devices

Bus-powered hubs use power from the USB bus to which they are connected, to power devices connected to them. Special care must be taken to not overload these hubs, since the power these hubs offer to their downstream devices is limited.

- Do not cascade bus-powered hubs. For example, do not connect one bus-powered hub to another bus-powered hub.
- Avoid connecting bus-powered devices to bus-powered hubs, except for low-speed, low-power devices, such as keyboards or mice. Connecting high-powered devices such as disks, speakers, or microphones to a bus-powered hub could cause power-shortages for all devices connected to that hub. This scenario could cause these devices to behave unpredictably.

USB Mass Storage Devices

All USB storage devices in this Solaris release are now accessed as removable media devices. This change has the following advantages:

- USB storage devices with standard MS-DOS or Windows (FAT) file systems are now supported.
- You can use the user-friendly `rmformat` command instead of the `format` command to format and partition all USB storage devices. If the functionality of the `format` command is needed, use the `format -e` command.
- You can use the `fdisk` command if you need to do `fdisk`-style partitioning.
- Non-root users can now access USB storage devices, since the root-privileged `mount` command is no longer needed. The device is automatically mounted by `vold` and is available under the `/rmdisk` directory. If a new device is connected while the system is down, do a reconfiguration boot with the `boot -r` command so that `vold` recognizes the device. Note that `vold` does not automatically recognize a hot-plugged device. If a new device is connected while the system is up, restart `vold`. For more information, refer to the `vold(1M)` and `scsa2usb(7D)` man pages.
- Disks with FAT file systems can be mounted and accessed. For example:

  ```
  mount -F pcfs /dev/dsk/c2t0d0s0:c /mnt
  ```
- All USB storage devices are now power managed, except for those that support `LOG SENSE` pages. Devices with `LOG SENSE` pages are usually SCSI drives connected through a USB-to-SCSI bridge device. In previous Solaris releases, some USB storage devices were not power managed because they were not seen as removable media.
- Applications might work differently with USB mass storage devices. Keep the following issues in mind when using applications with USB storage devices:
  - Applications might make incorrect assumptions about the size of the media since only smaller devices like diskettes and Zip drives were removable previously.

- Requests by applications to eject media on devices where this would be inapplicable, such as a hard drive, will succeed and do nothing.
- If you prefer the behavior in previous Solaris releases where not all USB mass storage were treated as removable media devices, then you can force the old behavior by updating the `/kernel/drv/scsa2usb.conf` file.

For more information on using USB mass storage devices, see the `scsa2usb(7D)` man page.

Troubleshooting Tips for USB Mass Storage Devices

Keep the following tips in mind if you have problems adding or removing a USB mass storage device.

- If USB devices are added or removed when the system is down, you must perform a reconfiguration boot.

  ```
  ok boot -r
  ```

  If you have problems accessing a device that was connected while the system is running, try the following command:

  ```
  # devfsadm
  ```
- Do not move devices around if the system has been powered down by a suspend operation. For more information, see "SPARC: USB Power Management" on page 120.
- If a device has been hot removed while in use by applications and is no longer available, then stop the applications. Use the `prtconf` command to see whether the device node has been removed.

SPARC: USB Driver Enhancements

This section describes USB driver enhancements in this Solaris release.

- **New generic USB driver –** USB devices can now be accessed and manipulated by applications using standard UNIX® read(2) and write(2) system calls, and without writing a special kernel driver. Additional features include:
  - Applications have access to raw device data and device status.
  - Supports control, bulk, and interrupt (in and out) transfers.

  For more information, refer to the `ugen(7D)` man page and the USB DDK at: http://developers.sun.com/solaris/developer/support/driver/usb.html
- **Digi Edgeport USB support –** Provides support for several Digi Edgeport USB to serial port converter devices.
  - New devices are accessed as `/dev/term/[0-9]*` and `/dev/cua/[0-9]*`.

    - USB serial ports are usable as any other serial port would be, except that they cannot serve as a local serial console. The fact that their data is run through a USB port is transparent to the user.

  For more information, see `usbser_edge(7D)`, or go to http://www.digi.com and http://www.sun.com/io.

- **Documentation and binary support for user-written kernel and userland drivers** – A Solaris USB Driver Development Kit (DDK) is available. For up-to-date information on USB driver development, including information on the DDK, go to: `http://developers.sun.com/solaris/developer/support/driver/usb.html`

The `EHCI` and `OHCI` Drivers

Features of the `EHCI` driver include:

- Complies with enhanced host controller interface that supports USB 2.0.
- Supports high-speed control, bulk, and interrupt transfers.
- Currently, there is no support for high-speed isochronous transactions. For example, USB 2.0 audio devices are not supported.

If there are USB 2.0 and USB 1.x devices on the system, the `EHCI` and `OHCI` drivers *hand-off* device control depending upon the type of device that is connected to the system.

- The USB 2.0 PCI card has one `EHCI` controller and one or more `OHCI` controllers.
- A USB 1.1 device is dynamically assigned to the `OHCI` controller when it is plugged in. A USB 2.0 device is dynamically assigned to the `EHCI` controller when it is plugged in.

Overview of USB Devices

Universal Serial Bus (USB) was developed by the PC industry to provide a low-cost solution for attaching peripheral devices, such as keyboards, mouse devices, and printers, to a system.

USB connectors are designed to fit only one type of cable, one way. The primary design motivation for USB was to alleviate the need for multiple connector types for different devices. This design reduces the clutter on the back panel of a system.

Devices connect to USB ports on external USB hubs, or on a root hub that is located on the computer itself. Since hubs have several ports, several branches of a device tree can stem from a hub.

This table lists specific USB devices that are supported in the Solaris environment.

USB Devices	Systems Supported
HID control on audio devices	SPARC based and x86 based systems.
Hubs	SPARC based and x86 based systems.
Keyboards and mouse devices	SPARC based and x86 based systems.
Mass storage devices	SPARC based and x86 based systems.
	Supported configurations include only one keyboard and mouse. These devices must be connected to an on-board USB host controller.
Printers	SPARC based and x86 based systems.
Speakers and microphones	SPARC based and x86 based systems.
USB serial converters	SPARC based systems.

Commonly Used USB Acronyms

The following table describes the USB acronyms that are used in the Solaris environment. For a complete description of USB components and acronyms, go to http://www.usb.org.

Acronym	Definition
ugen	USB generic driver
USB	Universal Serial Bus
USBA	Universal Serial Bus Architecture (Solaris)
USBAI	USBA Client Driver Interface (Solaris)
HCD	USB host controller driver
EHCI	Enhanced Open Controller Interface
OHCI	Open Host Controller Interface
UHCI	Universal Host Controller Interface

USB Bus Description

The USB specification is openly available and free of royalties. The specification defines the electrical and mechanical interfaces of the bus and the connectors.

USB employs a topology in which hubs provide attachment points for USB devices. The host controller contains the root hub, which is the origin of all USB ports in the system. For more information about hubs, see "USB Host Controller and Root Hub" on page 119.

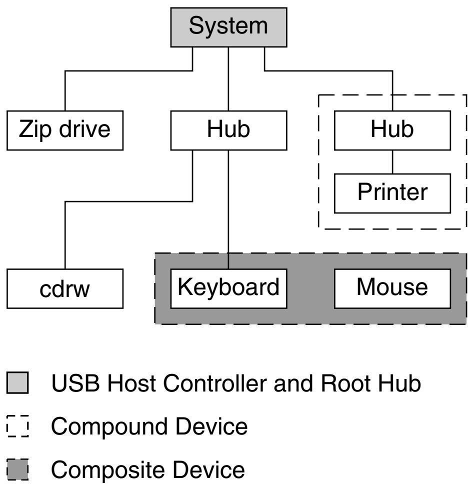

FIGURE 7–1 USB Physical Device Hierarchy

Figure 7–1 shows a system with three active USB ports. The first USB port connects a Zip drive. The second USB port connects an external hub, which in turn, connects a cdrw device and a composite keyboard/mouse device. As a *composite device*, this keyboard contains a USB controller, which operates both the keyboard and an attached mouse. The keyboard and the mouse share a common USB bus address because they are directed by the same USB controller.

Figure 7–1 also shows an example of a hub and a printer as a *compound device*. The hub is an external hub that is enclosed in the same casing as the printer. The printer is permanently connected to the hub. The hub and printer have separate USB bus addresses.

The device tree path name for some of the devices that are displayed in Figure 7–1 are listed in this table.

Zip drive	`/pci@1f,4000/usb@5/storage@1`
Keyboard	`/pci@1f,4000/usb@5/hub@2/device@1/keyboard@0`
Mouse	`/pci@1f,4000/usb@5/hub@2/device@1/mouse@1`
cdrw device	`/pci@1f,4000/usb@5/hub@2/storage@3`
Printer	`/pci@1f,4000/usb@5/hub@3/printer@1`

USB Devices and Drivers

USB devices with similar attributes and services are grouped into device classes. Each device class has a corresponding driver, one for each framework. Devices within a class are managed by the same device driver pair. However, the USB specification also allows for vendor-specific devices that are not part of a specific class.

The Human Interface Device (HID) class contains devices that are user-controlled such as keyboards, mouse devices, and joysticks. The Communication Device class contains devices that connect to a telephone, such as modems or an ISDN interface. Other device classes include the Audio, Monitor, Printer, and Storage Device classes. Each USB device contains descriptors that reflect the class of the device. A device class specifies how its members should behave in configuration and data transfer. You can obtain additional class information from http://www.usb.org.

Solaris USB Architecture (USBA)

USB devices can be represented as two levels of device tree nodes. A device node represents the entire USB *device*. One or more child *interface* nodes represent the individual USB interfaces on the device.

Driver binding is achieved by using the compatible name properties. For more information, refer to 3.2.2.1 of the IEEE 1275 USB binding and *Writing Device Drivers*. A driver can either bind to the entire device and control all the interfaces, or can bind to just one interface. If no vendor or class driver claims the entire device, a generic USB multi-interface driver is bound to the device-level node. This driver attempts to bind drivers to each interface by using compatible names properties, as defined in section 3.3.2.1 of the IEEE 1275 binding specification.

The Solaris USB Architecture (USBA) adheres to the USB 1.1 and USB 2.0 specifications plus Solaris driver requirements. The USBA model is similar to Sun Common SCSI Architecture (SCSA). The USBA is a thin layer that provides a generic USB transport-layer abstraction to client drivers, providing them with services that implement core generic USB functionality.

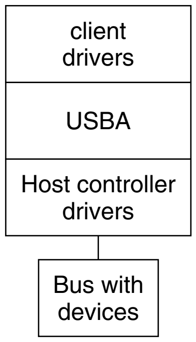

FIGURE 7–2 Solaris USB Architecture (USBA)

About USB in the Solaris Environment

This section describes information you should know about USB in the Solaris environment.

USB Keyboards and Mouse Devices

Only Sun USB keyboards and mouse devices are supported. System configurations with multiple USB keyboards and mouse devices might work, but are not supported in the Solaris environment. See the following items for details.

- A USB keyboard and mouse can be connected anywhere on the bus and can be configured as the console keyboard and mouse. Booting the system is slower if the keyboard and mouse are connected to an external hub.
- Do not move the console keyboard and mouse *during* a reboot or at the `ok` prompt. You can move the console keyboard and mouse to another hub at any time *after* a system reboot. After you plug in a keyboard and mouse, they are fully functional again.
- **SPARC** – The power key on a USB keyboard behaves differently than the power key on the Sun type 5 keyboard. On a USB keyboard, you can suspend or shut down the system by using the SUSPEND/SHUTDOWN key, but you cannot use that key to power up the system.
- The keys just to the left of the keypad do not function on third-party USB keyboards.
- Multiple keyboards are not supported:

  - Multiple keyboards enumerate and are usable, but they are not plumbed as console keyboards.
  - The first keyboard that is probed at boot time becomes the console keyboard. The result of this probing might cause confusion if multiple keyboards are plugged in at boot time.
  - If you unplug the console keyboard, the next available USB keyboard does not become the console keyboard. The next hot-plugged keyboard becomes the console keyboard.
- Multiple mouse devices are not supported:
  - Multiple mouse devices enumerate and are usable, but they are not plumbed as console mouse devices.
  - The first mouse that is probed at boot time becomes the console mouse. The result of this probing might cause confusion if you have multiple mouse devices plugged in at boot time.
  - If you unplug the console mouse, the next available USB mouse does not become the console mouse. The next hot-plugged mouse becomes the console mouse.
- If you have a third-party composite keyboard with a PS/2 mouse, and the composite keyboard/mouse is the first one to be probed, it becomes the console keyboard/mouse even if the PS/2 mouse is not plugged in. Thus, another USB mouse plugged into the system cannot work because it is not configured as the console mouse.
- Support for more than 3 buttons is available on USB or PS/2 mouse devices.
- Wheel mouse scrolling is available on a USB or PS/2 mouse device. This support means that rolling the wheel on a USB or a PS/2 mouse results in a scroll in the application or window under mouse focus. StarOffice™, Mozilla™, and GNOME applications support wheel mouse scrolling. However, other applications might not support wheel mouse scrolling.

USB Host Controller and Root Hub

A USB hub is responsible for the following:

- Monitoring the insertion or removal of a device on its ports
- Power-managing individual devices on its ports
- Controlling power to its ports

The USB host controller has an embedded hub called the *root hub*. The ports that are visible at the system's back panel are the ports of the root hub. The USB host controller is responsible for the following:

- Directing the USB bus. Individual devices cannot arbitrate for the bus.

- Polling the devices by using a polling interval that is determined by the device. The device is assumed to have sufficient buffering to account for the time between the polls.
- Sending data between the USB host controller and its attached devices. Peer-to-peer communication is not supported.

USB Hub Devices

- Do not cascade hubs beyond four levels on either SPARC based or x86 based systems. On SPARC systems, the OpenBoot™ PROM cannot reliably probe beyond four levels of devices.
- Do not plug a bus-powered hub into another bus-powered hub in a cascading style. A bus-powered hub does not have its own power supply.
- Do not connect a device that requires a large amount of power to a bus-powered hub. These devices might not work well on bus-powered hubs or might drain the hub of power for other devices. An example of such a device is a USB diskette device.
- This Solaris release does not support connecting a low- or full-speed device to a USB 2.0 hub that is connected to a USB 2.0 port on SPARC-based systems.

SPARC: USB Power Management

Suspending and resuming of USB devices is fully supported on SPARC systems. However, do not suspend a device that is busy and never remove a device when the system is powered off under a suspend shutdown.

The USB framework makes a best effort to power manage all devices on SPARC-based systems with power management enabled. Power managing a USB device means that the hub driver suspends the port to which the device is connected. Devices that support *remote wake up* can notify the system to wake up everything in the device's path, so that the device can be used. The host system could also wake up the device if an application sends an I/O to the device.

All HID (keyboard, mouse, speakers, microphones), hub, and storage devices are power-managed by default if they support remote wake up capability. A USB printer is power-managed only between two print jobs. Devices that are directed by the generic USB driver (UGEN) are power managed only when they are closed.

When power management is running to reduce power consumption, USB leaf devices are powered down first. After all devices that are connected to a hub's ports are powered down, the hub is powered down after some delay. To achieve the most efficient power management, do not cascade many hubs.

Guidelines for USB Cables

Keep the following guidelines in mind when connecting USB cables:

- Always use USB 2.0 compliant, fully rated (480 Mbit/sec) 20/28 AWG cables for connecting USB 2.0 devices.
- Always use USB 1.0 compliant, fully rated (12 Mbit/sec) 20/28 AWG cables for connecting USB 1.0 or 1.1 devices. Use bus-powered hubs for low-speed devices only. Always use fully rated (12 Mbit/sec) 20/28 AWG cables for connecting USB devices.
- Maximum cable length that is supported is 5 meters.
- Do not use cable extenders. For best results, use a self-powered hub to extend cable length.

For more information, go to `http://www.usb.org/channel/training/warning`.

CHAPTER 8

Using USB Devices (Tasks)

This chapter provides step-by-step instructions for using USB devices in the Solaris environment.

For information on the procedures associated with using USB devices, see the following:

- "Managing USB Devices in the Solaris Environment (Roadmap)" on page 123
- "Using USB Mass Storage Devices (Task Map)" on page 124
- "Hot-Plugging USB Devices (Task Map)" on page 134
- "Using USB Audio Devices (Task Map)" on page 139
- "Hot-Plugging USB Devices With the `cfgadm` Command (Task Map)" on page 143

For overview information about using USB devices, see Chapter 7.

Managing USB Devices in the Solaris Environment (Roadmap)

Use this map to identify all the tasks for managing USB devices in the Solaris environment. Each task points to a series of additional tasks such as using USB devices, hot-plugging USB devices, or adding USB audio devices.

Task	Description	For Instructions
Using USB devices	USB devices must be formatted before file systems can be created and mounted.	"Using USB Mass Storage Devices (Task Map)" on page 124

Task	Description	For Instructions
Hot-plug USB devices	Dynamically add or remove USB devices from your system.	
	You can physically add or remove USB devices to and from your system.	"Hot-Plugging USB Devices (Task Map)" on page 134
	Physically or logically add or remove USB devices to and from your system with the `cfgadm` command.	"Hot-Plugging USB Devices With the `cfgadm` Command (Task Map)" on page 143
Add USB audio devices	Use this map to identify tasks associated with adding USB audio devices.	"Using USB Audio Devices (Task Map)" on page 139

Using USB Mass Storage Devices (Task Map)

Task	Description	For Instructions
Prepare to use a USB mass storage device	Prepare to use a USB mass storage device with `vold` running.	"Preparing to Use a USB Mass Storage Device With `vold` Running" on page 126
	Prepare to use a USB mass storage device without `vold` running.	"How to Prepare to Use USB Mass Storage Devices Without `vold` Running" on page 127
Display USB device information	Use the `prtconf` command to display information about USB devices.	"How to Display USB Device Information (`prtconf`)" on page 128
Format a USB mass storage device	Format a USB mass storage device so that you can put data on it.	"How to Format a USB Mass Storage Device Without `vold` Running" on page 129
Mount a USB mass storage device	Mount a USB mass storage device with `vold` running.	"How to Mount or Unmount a USB Mass Storage Device With `vold` Running" on page 131

Task	Description	For Instructions
	Mount a USB mass storage device without `vold` running.	"How to Mount or Unmount a USB Mass Storage Device Without `vold` Running" on page 132
(Optional) Disable USB device drivers	Disable USB device drivers if you do not want the USB support on your system.	
(Optional) Remove unused USB device links	Remove unused USB device links with the `devfsadm` command.	"How to Remove Unused USB Device Links" on page 134

Using USB Mass Storage Devices

Note – For up-to-date information on using USB mass storage devices in this Solaris release, see "USB Mass Storage Devices" on page 112.

Starting in the Solaris 9 release, removable mass storage devices such as USB CD-RWs, hards disks, DVDs, digital cameras, Zip, Peerless, SmartMedia, CompactFlash, ORB, and USB diskette devices are supported.

For a complete list of USB devices that are supported in the Solaris environment, see http://www.sun.com/io_technologies/USB.html.

These devices can be managed with or without volume management. For information on managing devices with volume management, see `vold`(1M).

Using USB Diskette Devices

USB diskette devices appear as removable media devices like other USB devices. USB diskette devices are not managed by the `fd` (floppy) driver. Applications that issue `ioctl(2)` calls intended for the `fd` (native floppy) driver will fail. Applications that issue only `read(2)` and `write(2)` calls will succeed. Other applications, such as SunPCI and `rmformat`, will also succeed.

Note – CDE's File Manager does not fully support USB diskettes at this time. However, you can open, rename, and format diskettes that contain a UFS file system from File Manager's Removable Media Manager. You can only open diskettes that contain a PCFS file system from the Removable Media Manager. If a diskette contains either type of file system, you can successfully drag and drop files between the diskette and File Manager.

Volume management (`vold`) sees the USB diskette device as a SCSI removable media device. Volume management makes the device available for access under the `/rmdisk` directory.

For more information on how to use USB diskette devices, see Chapter 1.

Using Non-Compliant USB Mass Storage Devices

Some devices might be supported by the USB mass storage driver even though they do not identify themselves as compliant with the USB mass storage class or identify themselves incorrectly. The `scsa2usb.conf` file contains an attribute-override-list that lists the vendor ID, product ID, and revision for matching mass storage devices, as well as fields for overriding the default device attributes. The entries in this list are commented out by default, and can be copied and uncommented to enable support of particular devices.

If you connect a USB mass storage device to a system running this Solaris release and the system is unable to use it, you can check the `/kernel/drv/scsa2usb.conf` file to see if there is a matching, commented entry for this device. Follow the information given in the `scsa2usb.conf` file to see if a particular device can be supported by using the override information. For a listing of recommended USB mass storage devices, go to http://www.sun.com/io_technologies/USB.html.

For more information, see `scsa2usb`(7D).

Preparing to Use a USB Mass Storage Device With `vold` Running

If you are running the Solaris Common Desktop Environment (CDE), USB removable mass storage devices are managed by the Removable Media Manager component of the CDE File Manager. For more information on the CDE File Manager, see `dtfile(1)`.

Note – You must include the `/usr/dt/man` directory in your `MANPATH` variable to display the man pages that are listed in this section. You must also have the `/usr/dt/bin` directory in your path and have CDE running to use these commands, or have a `DISPLAY` variable set to use these commands remotely.

The following table identifies the commands that Removable Media Manager uses to manage storage devices from the CDE environment.

Command	Man Page	Task
`sdtmedia_format`	`sdtmedia_format(1)`	Format and label a device
`sdtmedia_prop`	`sdtmedia_prop(1)`	Display properties of a device
`sdtmedia_prot`	`sdtmedia_prot(1)`	Change device protection
`sdtmedia_slice`	`sdtmedia_slice(1)`	Create or modify slices on a device

After the USB device is formatted, it is usually mounted under the `/rmdisk/`*label* directory. For more information on configuring removable storage devices, see `rmmount.conf`(4) or `vold.conf`(4).

The device nodes are created under the `/vol/dev` directory. For more information, see `scsa2usb(7D)`.

The following procedures describe how to manage USB mass storage devices without `vold` running. The device nodes are created under the `/dev/rdsk` directory for character devices and under the `/dev/dsk` directory for block devices. Device links are created when the devices are hot-plugged. For more information, see `scsa2usb`(7D).

▼ How to Prepare to Use USB Mass Storage Devices Without `vold` Running

You can use USB mass storage devices without the volume management (`vold`) running. Stop `vold` by issuing the following command:

```
# /etc/init.d/volmgt stop
```

Or, use the following procedure to keep `vold` running, but do not register the USB mass storage devices with `vold`.

Steps 1. **Become superuser.**

2. **Remove volume manager registration of USB mass storage devices by commenting the following line in the `/etc/vold.conf` file, like this:**

   ```
   # use rmdisk drive /dev/rdsk/c*s2 dev_rmdisk.so rmdisk%d
   ```

3. **After this line is commented, restart `vold`.**

   ```
   # /etc/init.d/volmgt start
   ```

Caution – If you comment out this line and other SCSI or ATAPI Zip, Peerless or other removable devices are in the system, `vold` registration for these devices would be disabled as well.

For more information, see `vold.conf`(4).

How to Display USB Device Information (`prtconf`)

Use the `prtconf` command to display information about USB devices.

```
$ prtconf
        usb, instance #0
                 hub, instance #2
                     device, instance #8
                         interface (driver not attached)
                     printer (driver not attached)
                     mouse, instance #14
                     device, instance #9
                         keyboard, instance #15
                         mouse, instance #16
                     storage, instance #7
                         disk (driver not attached)
                     communications, instance #10
                         modem (driver not attached)
                         data (driver not attached)
                 storage, instance #0
                     disk (driver not attached)
                 storage, instance #1
                     disk (driver not attached)
```

You can use the `prtconf` command's `-D` option to display additional driver information. This information can be used to tell which ports and devices are being driven by the USBA 1.0 framework on SPARC systems, as displayed in the following example:

```
$ prtconf -D
.
.
```

```
.
SUNW,Sun-Blade-1500
.
.
.
    1 pci, instance #0 (driver name: pcisch)
           isa, instance #0 (driver name: ebus)
.
.
.
    2 usb, instance #0 (driver name: ohci)
           usb, instance #1 (driver name: ohci)
.
.
.
    3 pci, instance #0 (driver name: pci_pci)
    4 usb, instance #0 (driver name: usba10_ohci)
            usb, instance #1 (driver name: usba10_ohci)
            usb, instance #0 (driver name: usba10_ehci)
                storage, instance #9 (driver name: usba10_scsa2usb)
                    disk, instance #9 (driver name: usb_sd)
            firewire, instance #0 (driver name: hci1394)
.
.
.
```

In the output above, note the following configuration characteristics:

- PCI card ports are distinguished by the number of hierarchical `pci` nodes in the output above their `usb` nodes.

  PCI card ports (4) fall under two hierarchical `pci` nodes 1 and 3 because they are driven through both the motherboard and the PCI card. Onboard ports (2) fall under a single PCI node (1) because they are one hardware architectural layer closer to the main system bus.
- The name of a driver associated with a device node indicates which framework is directing the device and the port to which the device is attached. The drivers for all USB instances of (4) begin with `usba10`, indicating that the USBA 1.0 framework is managing those ports and the devices attached to them. Only those ports can support USB 2.0 devices at high speed.

▼ How to Format a USB Mass Storage Device Without `vold` Running

USB mass storage devices, as all others used by the Solaris operating system, must be formatted and contain a file system before they can be used. USB mass storage devices, including diskettes, support both PCFS and UFS file systems. Be sure the disk is formatted before putting either a PCFS or UFS file system on it.

Steps **1. See "How to Prepare to Use USB Mass Storage Devices Without `vold` Running" on page 127 for information on disabling `vold`.**

2. **(Optional) Add the USB diskette device to your system.**

   For information on hot-plugging USB devices, see:

   - "Using USB Audio Devices (Task Map)" on page 139
   - "Hot-Plugging USB Devices With the `cfgadm` Command (Task Map)" on page 143

3. **(Optional) Identify the diskette device.**

   For example:

   ```
   # cd /dev/rdsk
   # devfsadm -C
   # ls -l c*0 | grep usb
   lrwxrwxrwx   1 root  root   55 Mar  5 10:35 c2t0d0s0 ->
   ../../devices/pci@1f,0/usb@c,3/storage@3/disk@0,0:a,raw
   ```

   In this example, the diskette device is `c2t0d0s0`.

4. **Insert a diskette into the diskette drive.**

5. **Format the diskette.**

   ```
   % rmformat -Flong raw-device
   ```

   For example:

   ```
   % rmformat -Flong /dev/rdsk/c2t0d0s0
   ```

6. **Determine the file system type and select one of the following:**

   a. **Create a PCFS file system.**

      ```
      # mkfs -F pcfs -o nofdisk,size=size raw-device
      ```

      Specify the `-size` option in 512–byte blocks.

      The following example shows how to create a PCFS file system on a 1.4 Mbyte diskette.

      ```
      # mkfs -F pcfs -o nofdisk,size=2880 /dev/rdsk/c4t0d0s0
      ```

      The following example shows how to create a UFS file system on a 100 Mbyte Zip drive.

      ```
      # mkfs -F pcfs -o nofdisk,size=204800 /dev/rdsk/c5t0d0s0
      ```

      This command can take several minutes to complete.

   b. **Create a UFS file system.**

      ```
      # newfs raw-device
      ```

      For example:

      ```
      # newfs /dev/rdsk/c4t0d0s0
      ```

Note – UFS file system overhead consumes a significant portion of space on a diskette, due to a diskette's limited storage capacity.

▼ How to Mount or Unmount a USB Mass Storage Device With `vold` Running

Steps

1. **Display device aliases for all removable mass storage devices, including USB mass storage devices.**

```
$ eject -n
.
.
.
cdrom0 -> /vol/dev/rdsk/c0t6d0/audio_cd   (Generic CD device)
zip0 -> /vol/dev/rdsk/c1t0d0/zip100       (USB Zip device)
zip1 -> /vol/dev/rdsk/c2t0d0/fat32        (USB Zip device)
rmdisk0 -> /vol/dev/rdsk/c5t0d0/unnamed_rmdisk (Peerless, HD or floppy)
rmdisk1 -> /vol/dev/rdsk/c4t0d0/clik40    (Generic USB storage)
```

2. **Select one of the following to mount or unmount a USB mass storage device.**

   a. **Mount a USB mass storage device by using the device aliases listed previously.**

   ```
   $ volrmmount -i device-alias
   ```

   This example shows how to mount a USB Zip drive (`/rmdisk/zip0`).

   ```
   $ volrmmount -i zip0
   ```

   b. **Unmount a USB mass storage device.**

   ```
   $ volrmmount -e device-alias
   ```

   This example shows how to unmount a USB Zip drive (`/rmdisk/zip0`).

   ```
   $ volrmmount -e zip0
   ```

3. **Eject a USB device from a generic USB drive.**

```
$ eject device-alias
```

For example:

```
$ eject rmdisk0
```

Note – The `eject` command also unmounts the device if the device is not unmounted already. The command also terminates any active applications that access the device.

▼ How to Mount or Unmount a USB Mass Storage Device Without `vold` Running

Steps

1. **See "How to Prepare to Use USB Mass Storage Devices Without `vold` Running" on page 127 for information on disabling `vold`.**

2. **Become superuser.**

3. **(Optional) Identify the diskette device.**

   For example:

   ```
   # cd /dev/rdsk
   # devfsadm -C
   # ls -l c*0 | grep usb
   lrwxrwxrwx   1 root   root   55 Mar  5 10:35 c2t0d0s0 ->
   ../../devices/pci@1f,0/usb@c,3/storage@3/disk@0,0:a,raw
   ```

   In this example, the diskette device is c2t0d0s0.

4. **Select one of the following to mount or unmount a USB mass storage device.**

   a. **Mount a USB mass storage device.**

   ```
   # mount [ -F fstype ] block-device mount-point
   ```

   This example shows how to mount a device with a UFS file system.

   ```
   # mount /dev/dsk/c1t0d0s2 /mnt
   ```

   This example shows how to mount a device with a PCFS file system.

   ```
   # mount -F pcfs /dev/dsk/c1t0d0s0:c /mnt
   ```

   This example shows how to mount a CD with a read-only HSFS file system.

   ```
   # mount -F hsfs -o ro /dev/dsk/c1t0d0s2 /mnt
   ```

   b. **Unmount a USB mass storage device.**

   First, be sure no one is using the file system on the device.

   For example:

   ```
   # fuser -c -u /mnt
   # umount /mnt
   ```

   c. **Eject the device.**

```
# eject /dev/[r]dsk/cntndnsn
```

For example:

```
# eject /dev/rdsk/c1t0d0s2
```

Disabling Specific USB Drivers

You can disable specific types of USB devices by disabling their client driver. For example, USB printers can be disabled by disabling the `usbprn` driver that directs them. Disabling `usbprn` does not affect other kinds of devices, such as USB storage devices.

Be careful that device types are disabled on both frameworks. You cannot disable device types on one framework only. The following table identifies some USB device types and their corresponding drivers.

Device Type	Driver to Disable
audio	`usb_ac` and `usb_as`
HID (usually keyboard and mouse	`hid`
storage	`scsa2usb`
printer	`usbprn`
serial	`usbser_edge`

If you disable a driver for a USB device that is still connected to the system, you will see a console message similar to the following:

```
usba10: WARNING: usba:    no driver found for device name
```

▼ How to Disable Specific USB Drivers

Steps

1. **Become superuser.**
2. **Record the driver aliases that you are about to remove.**

   ```
   # cp /etc/driver_aliases /etc/driver_aliases.orig
   ```

3. **Identify the specific USB driver alias name.**

   For example:

   ```
   # grep usbprn /etc/driver_aliases
   usbprn "usbif,class7.1.1"
   usbprn "usbif,class7.1.2"
   ```

4. **Remove the driver alias entry.**

   For example:

   ```
   # update_drv -d -i '"usbif,class7.1.1"' usbprn
   # update_drv -d -i '"usbif,class7.1.2"' usbprn
   ```

5. **Reboot the system.**

   ```
   # init 6
   ```

▼ How to Remove Unused USB Device Links

Use this procedure if a USB device is removed while the system is powered off. It is possible that removing the USB device while the system is powered off will leave device links for devices that do not exist.

Steps

1. **Become superuser.**
2. **Close all applications that might be accessing the device.**
3. **Remove the unused links for a specific USB class.**

   For example:

   ```
   # devfsadm -C -c audio
   ```

   Or, just remove the dangling links:

   ```
   # devfsadm -C
   ```

Hot-Plugging USB Devices (Task Map)

Task	Description	For Instructions
Add a USB mass storage device	Add a USB mass storage device with `vold` running.	"How to Add a USB Mass Storage Device With `vold` Running" on page 135
	Add a USB mass storage device without `vold` running.	"How to Add a USB Mass Storage Device Without `vold` Running" on page 136

Task	Description	For Instructions
Remove a USB mass storage device	Remove a USB mass storage device with vold running.	"How to Remove a USB Mass Storage Device With vold Running" on page 136
	Remove a USB mass storage device without vold running.	"How to Remove a USB Mass Storage Device Without vold Running" on page 137
Add a USB camera	Add a USB camera to access digital images.	"How to Add a USB Camera" on page 137

Hot-plugging a device means the device is added or removed without shutting down the operating system or powering off the system. All USB devices are hot-pluggable.

When you hot-plug a USB device, the device is immediately seen in the system's device hierarchy, as displayed in the `prtconf` command output. When you remove a USB device, the device is removed from the system's device hierarchy, unless the device is in use.

If the USB device is in use when it is removed, the hot-plug behavior is a little different. If a device is in use when it is unplugged, the device node remains, but the driver controlling this device stops all activity on the device. Any new I/O activity issued to this device is returned with an error.

In this situation, the system prompts you to plug in the original device. If the device is no longer available, stop the applications. After a few seconds, the port will become available again.

Note – Data integrity might be impaired if you remove an active or open device. Always close the device before removing, except the console keyboard and mouse, which can be moved while active.

▼ How to Add a USB Mass Storage Device With `vold` Running

This procedure describes how to add a USB device with `vold` running.

Steps

1. **Connect the USB mass storage device.**
2. **Instruct `vold` to scan for new devices.**

   ```
   # touch /etc/vold.conf
   ```
3. **Restart `vold`.**

```
# pkill -HUP vold
```

4. **Verify that the device has been added.**

   ```
   $ ls device-alias
   ```

   For more information on volume management device names, see Chapter 1.

▼ How to Add a USB Mass Storage Device Without vold Running

This procedure describes how to add a USB device without `vold` running.

Steps

1. **If needed, see "How to Prepare to Use USB Mass Storage Devices Without `vold` Running" on page 127 for information on disabling `vold`.**
2. **Connect the USB mass storage device.**
3. **Verify that the USB device has been added.**

   Locate the USB disk device links, which may be among device links of non-USB storage devices, as follows:

   ```
   $ cd /dev/rdsk
   $ ls -l c*0 | grep usb
   lrwxrwxrwx   1 root     root          67 Apr 30 15:12 c1t0d0s0 ->
       ../../devices/pci@1f,0/pci@5/pci@0/usb@8,2/storage@1/disk@0,0:a,raw
   ```

▼ How to Remove a USB Mass Storage Device With vold Running

The following procedure uses a Zip drive as an example of removing a USB device with `vold` running.

Steps

1. **Stop any active applications that are using the device.**
2. **Unmount the device.**

   For example:

   ```
   $ volrmmount -e zip0
   ```

3. **Eject the device.**

   For example:

   ```
   $ eject zip0
   ```

4. **Become superuser and stop `vold`.**

```
# /etc/init.d/volmgt stop
```

5. **Remove the USB mass storage device.**

6. **Start `vold`.**

```
# /etc/init.d/volmgt start
```

▼ How to Remove a USB Mass Storage Device Without `vold` Running

This procedure describes how to remove a USB device without `vold` running.

Steps

1. **If needed, see "How to Prepare to Use USB Mass Storage Devices Without `vold` Running" on page 127 for information on disabling `vold`.**

2. **Become superuser.**

3. **Stop any active applications that are using the device.**

4. **Remove the USB device.**

   a. **Unmount the device.**

      For example:

      ```
      # umount /mnt
      ```

   b. **Remove the device.**

▼ How to Add a USB Camera

Use this procedure to add a USB camera.

Steps

1. **Become superuser.**

2. **Plug in and turn on the USB camera.**

   The system creates a logical device for the camera. After the camera is plugged in, output is written to the `/var/adm/messages` file to acknowledge the device's connection. The camera is seen as a storage device to the system.

3. **Examine the output that is written to the `/var/adm/messages` file.**

   Examining this output enables you to determine which logical device was created so that you can then use that device to access your images. The output will look similar to the following:

   ```
   # more /var/adm/messages
   Jul 15 09:53:35 buffy usba: [ID 349649 kern.info]    OLYMPUS, C-3040ZOOM,
   ```

```
 000153719068
Jul 15 09:53:35 buffy genunix: [ID 936769 kern.info] scsa2usb1 is
/pci@0,0/pci925,1234@7,2/storage@2
Jul 15 09:53:36 buffy scsi: [ID 193665 kern.info] sd3 at scsa2usb1:
target 0 lun 0
```

Match the device with a mountable `/dev/dsk` link entry, by doing the following:

```
# ls -l /dev/dsk/c*0 | grep /pci@0,0/pci925,1234@7,2/storage@2
 lrwxrwxrwx   1 root     root          58 Jul 15  2002 c3t0d0p0 ->
 ../../devices/pci@0,0/pci925,1234@7,2/storage@2/disk@0,0:a
```

4. **Mount the USB camera file system.**

   The camera's file system is most likely a PCFS file system. In order to mount the file system on the device created, the slice that represents the disk must be specified. The slice is normally `s0` for a SPARC system, and `p0` for an x86 system.

   For example, to mount the file system on an x86 system, execute the following command:

   ```
   # mount -F pcfs /dev/dsk/c3t0d0p0:c /mnt
   ```

   To mount the file system on a SPARC system, execute the following command:

   ```
   # mount -F pcfs /dev/dsk/c3t0d0s0:c /mnt
   ```

   For information on mounting file systems, see Chapter 17.

   For information on mounting different PCFS file systems, see `mount_pcfs`(1M).

5. **Verify that the image files are available.**

   For example:

   ```
   # ls /mnt/DCIM/100OLYMP/
   P7220001.JPG*  P7220003.JPG*  P7220005.JPG*
   P7220002.JPG*  P7220004.JPG*  P7220006.JPG*
   ```

6. **View and manipulate the image files created by the USB camera.**

   ```
   # /usr/dt/bin/sdtimage P7220001.JPG &
   ```

7. **Unmount the file system before disconnecting the camera.**

   For example:

   ```
   # umount /mnt
   ```

8. **Turn off and disconnect the camera.**

Using USB Audio Devices (Task Map)

Task	Description	For Instructions
Add USB audio devices	Add a USB microphone and speakers.	"How to Add USB Audio Devices" on page 141
Identify your system's primary audio device	Identify which audio device is your primary audio device.	"How to Identify Your System's Primary Audio Device" on page 141
Change the primary USB audio device	You might want to make one particular audio device the primary audio device if you remove or change your USB audio devices.	"How to Change the Primary USB Audio Device" on page 142
Remove unused USB device links	If you remove a USB audio device while the system is powered off, the `/dev/audio` device might be pointing to a `/dev/sound/*` device that doesn't exist.	"How to Remove Unused USB Device Links" on page 134
Solve USB audio problems	Use this section if no sound comes from the USB speakers.	"Solving USB Audio Problems" on page 142

Using USB Audio Devices

This Solaris release provides USB audio support that is implemented by a pair of cooperating drivers, `usb_ac` and `usb_as`. The audio control driver, `usb_ac`, is a USBA (Solaris USB Architecture) compliant client driver that provides the controlling interface to user applications. The audio streaming driver, `usb_as`, is provided to process audio data messages during play and record. It sets sample frequency and precision, and encodes requests from the `usb_ac` driver. Both drivers comply to the USB audio class 1.0 specification.

Some audio devices can set volume under software control. A STREAMS module, `usb_ah`, is pushed on top of the HID driver for managing this function.

Solaris supports USB audio devices that are play-only, record-only, or record and play. Hot-plugging of USB audio devices is supported.

- USB audio devices are supported on SPARC Ultra and x86 platforms that have USB connectors.

- USB audio devices that are supported in the Solaris 8 10/01, Solaris 8 2/02, or Solaris 9 releases must support a fixed 44100 or 48000 Hz sampling frequency to play or record.
- For fully supported audio data format information, see `usb_ac`(7D).

The primary audio device is `/dev/audio`. You can verify that `/dev/audio` is pointing to USB audio by using the following command:

```
% mixerctl
Device /dev/audioctl:
  Name    = USB Audio
  Version = 1.0
  Config  = external

Audio mixer for /dev/audioctl is enabled
```

After you connect your USB audio devices, you access them with the `audioplay` and `audiorecord` command through the `/dev/sound/`*N* device links.

Note that the `/dev/audio` and `/dev/sound/N` devices can refer to speakers, microphones, or combo devices. If you refer to the incorrect device type, the command will fail. For example, the `audioplay` command will fail if you try to use it with a microphone.

You can select a specific default audio device for most Sun audio applications, such as `audioplay` and `audiorecord`, by setting the `AUDIODEV` shell variable or by specifying the `-d` option for these commands. However, setting `AUDIODEV` does not work for third-party applications that have `/dev/audio` hardcoded as the audio file.

When you plug in a USB audio device, it automatically becomes the primary audio device, `/dev/audio`, unless `/dev/audio` is in use. For instructions on changing `/dev/audio` from onboard audio to USB audio and vice versa, refer to "How to Change the Primary USB Audio Device" on page 142, and `usb_ac`(7D).

Hot-Plugging Multiple USB Audio Devices

If a USB audio device is plugged into a system, it becomes the primary audio device, `/dev/audio`. It remains the primary audio device even after the system is rebooted. If additional USB audio devices are plugged in, the last one becomes the primary audio device.

For additional information on troubleshooting USB audio device problems, see `usb_ac(7D).`

▼ How to Add USB Audio Devices

Use the following procedure to add USB audio devices.

Steps

1. **Plug in the USB speaker.**

   The primary audio device, `/dev/audio`, points to the USB speaker.

   ```
   % ls -l /dev/audio
   lrwxrwxrwx   1 root     root    10 Feb 13 08:46 /dev/audio -> usb/audio0
   ```

2. **(Optional) Remove the speaker. Then plug it back in.**

   If you remove the speaker, the `/dev/audio` device reverts back to onboard audio.

   ```
   % ls -l /dev/audio
   lrwxrwxrwx   1 root     root   7 Feb 13 08:47 /dev/audio -> sound/0
   ```

3. **Add a USB microphone.**

   ```
   % ls -l /dev/audio
   lrwxrwxrwx   1 root     root    10 Feb 13 08:54 /dev/audio -> usb/audio1
   ```

▼ How to Identify Your System's Primary Audio Device

This procedure assumes that you have already connected the USB audio devices.

Step

● **Examine your system's new audio links.**

For example:

```
% ls -lt /dev/audio*
lrwxrwxrwx   1 root     root      7 Jul 23 15:46 /dev/audio -> usb/audio0
lrwxrwxrwx   1 root     root     10 Jul 23 15:46 /dev/audioctl ->
usb/audioctl0/
% ls -lt /dev/sound/*
lrwxrwxrwx   1 root     root     74 Jul 23 15:46 /dev/sound/1 ->
../../devices/pci@1f,4000/usb@5/hub@1/device@3/sound-control@0:sound,a...
lrwxrwxrwx   1 root     root     77 Jul 23 15:46 /dev/sound/1ctl ->
../../devices/pci@1f,4000/usb@5/hub@1/device@3/sound-control@0:sound,a...
lrwxrwxrwx   1 root     other    66 Jul 23 14:21 /dev/sound/0 ->
../../devices/pci@1f,4000/ebus@1/SUNW,CS4231@14,200000:sound,audio
lrwxrwxrwx   1 root     other    69 Jul 23 14:21 /dev/sound/0ctl ->
../../devices/pci@1f,4000/ebus@1/SUNW,CS4231@14,200000:sound,audioctl
%
```

Notice that the primary audio device, `/dev/audio`, is pointing to the newly plugged in USB audio device, `/dev/usb/audio0`.

You can also examine your system's USB audio devices with the `prtconf` command and look for the USB device information.

```
% prtconf
.
.
.
usb, instance #0
   hub, instance #0
        mouse, instance #0
        keyboard, instance #1
        device, instance #0
            sound-control, instance #0
            sound, instance #0
            input, instance #0
.
.
.
```

How to Change the Primary USB Audio Device

- If you want the onboard audio device to become the primary audio device, remove the USB audio devices. The `/dev/audio` link will then point to the `/dev/sound/0` entry. If the `/dev/sound/0` entry is not the primary audio device, then either shutdown the system and use the `boot -r` command, or run the `devfsadm -i` command as root.
- If you want the USB audio device to become primary audio device, just plug it in and check the links.

Troubleshooting USB Audio Device Problems

This section describes how to troubleshoot USB audio device problems.

Solving USB Audio Problems

Sometimes USB speakers do not produce any sound even though the driver is attached and the volume is set to high. Hot-plugging the device might not change this behavior.

The workaround is to power cycle the USB speakers.

Key Points of Audio Device Ownership

Keep the following key points of audio device ownership in mind when working with audio devices.

- When you plug in a USB audio device and you are logged in on the console, the console is the owner of the `/dev/*` entries. This situation means you can use the audio device as long as you are logged into the console.

- If you are not logged into the console when you plug in a USB audio device, root becomes the owner of the device. However, if you log into the console and attempt to access the USB audio device, device ownership changes to the console. For more information, see `logindevperm`(4).
- When you remotely login with the `rlogin` command and attempt to access the USB audio device, the ownership does not change. This means that, for example, unauthorized users cannot listen to conversations over a microphone owned by someone else.

Hot-Plugging USB Devices With the `cfgadm` Command (Task Map)

Task	Description	For Instructions
Display USB bus information	Display information about USB devices and buses.	"How to Display USB Bus Information (`cfgadm`)" on page 145
Unconfigure a USB device	Logically unconfigure a USB device that is still physically connected to the system.	"How to Unconfigure a USB Device" on page 146
Configure a USB device	Configure a USB device that was previously unconfigured.	"How to Configure a USB Device" on page 146
Logically disconnect a USB device	You can logically disconnect a USB device if you are not physically near the system.	"How to Logically Disconnect a USB Device" on page 146
Logically connect a USB device	Logically connect a USB device that was previously logically disconnected or unconfigured.	"How to Logically Connect a USB Device" on page 147
Disconnect a USB device subtree	Disconnect a USB device subtree, which is the hierarchy (or tree) of devices below a hub.	"How to Logically Disconnect a USB Device Subtree" on page 147
Reset a USB device	Reset a USB device to logically remove and recreate the device.	"How to Reset a USB Device" on page 148

Task	Description	For Instructions
Change the default configuration of a multi-configuration USB device	Change the default configuration of a multi-configuration USB device.	"How to Change the Default Configuration of a Multi-Configuration USB Device" on page 148

Hot-Plugging USB Devices With the `cfgadm` Command

You can add and remove a USB device from a running system without using the `cfgadm` command. However, a USB device can also be *logically* hot-plugged without physically removing the device. This scenario is convenient when you are working remotely and you need to disable or reset a non-functioning USB device. The `cfgadm` command also provides a way to display the USB device tree including manufacturer and product information.

The `cfgadm` command displays information about attachment points, which are locations in the system where dynamic reconfiguration operations can occur. An attachment point consists of:

- An occupant, which represents a hardware resource, such as a USB device, that might be configured into the system, and
- A receptacle, which is the location that accepts the occupant, such as a USB port.

Attachment points are represented by logical and physical attachment point IDs (`Ap_Ids`). The physical `Ap_Id` is the physical pathname of the attachment point. The logical `Ap_Id` is a user-friendly alternative for the physical `Ap_Id`. For more information on `Ap_Ids`, see `cfgadm_usb`(1M).

The `cfgadm` command provides the following USB device status information.

Receptacle State	Description
empty/unconfigured	The device is not physically connected.
disconnected/unconfigured	The device is logically disconnected and unavailable, even though the device could still be physically connected.
connected/unconfigured	The device is logically connected, but unavailable. The device is visible in `prtconf` output.
connected/configured	The device is connected and available.

The following sections describe how to hot-plug a USB device through the software with the `cfgadm` command. All of the sample USB device information in these sections has been truncated to focus on relevant information.

How to Display USB Bus Information (`cfgadm`)

Use the `cfgadm` command to display USB bus information. For example:

```
% cfgadm
Ap_Id                        Type         Receptacle   Occupant     Condition
usb0/4.5                     usb-hub      connected    configured   ok
usb0/4.5.1                   usb-device   connected    configured   ok
usb0/4.5.2                   usb-printer  connected    configured   ok
usb0/4.5.3                   usb-mouse    connected    configured   ok
usb0/4.5.4                   usb-device   connected    configured   ok
usb0/4.5.5                   usb-storage  connected    configured   ok
usb0/4.5.6                   usb-communi  connected    configured   ok
usb0/4.5.7                   unknown      empty        unconfigured ok
usb0/4.6                     usb-storage  connected    configured   ok
usb0/4.7                     usb-storage  connected    configured   ok
```

In the preceding example, `usb0/4.5.1` identifies a device connected to port 1 of the second-level external hub, which is connected to port 5 of first-level external hub, which is connected to the first USB controller's root hub, port 4.

Use the following `cfgadm` command to display specific USB device information. For example:

```
% cfgadm -l -s "cols=ap_id:info"
Ap_Id        Information
 usb0/4.5.1  Mfg: Inside Out Networks Product: Edgeport/421 NConfigs: 1
Config: 0  : ...
 usb0/4.5.2  Mfg: <undef> Product: <undef>   NConfigs: 1 Config: 0 ...
 usb0/4.5.3  Mfg: Mitsumi Product: Apple USB Mouse NConfigs: 1 Config: 0 ...
 usb0/4.5.4  Mfg: NMB  Product: NMB USB KB/PS2 M NConfigs: 1 Config: 0
 usb0/4.5.5  Mfg: Hagiwara Sys-Com  Product: SmartMedia R/W  NConfigs: 1
Config: 0 : ...
 usb0/4.5.6  Mfg: 3Com Inc.  Product: U.S.Robotics 56000 Voice USB Modem
NConfigs: 2 ...
 usb0/4.5.7
 usb0/4.6    Mfg: Iomega  Product: USB Zip 250  NConfigs: 1  Config: 0
 : Default
 usb0/4.7    Mfg: Iomega  Product: USB Zip 100  NConfigs: 1  Config: 0
 : Default
```

For examples of using the `prtconf` command to display USB configuration information, see "How to Display USB Device Information (`prtconf`)" on page 128.

▼ How to Unconfigure a USB Device

You can unconfigure a USB device that is still physically connected to the system, but a driver will never attach to it. Note that a USB device remains in the `prtconf` output even after that device is unconfigured.

Steps

1. **Become superuser.**
2. **Unconfigure the USB device.**

```
# cfgadm -c unconfigure usb0/4.7
Unconfigure the device: /devices/pci@8,700000/usb@5,3/hub@4:4.7
This operation will suspend activity on the USB device
Continue (yes/no)? y
```

3. **Verify that the device is unconfigured.**

```
# cfgadm
Ap_Id                    Type         Receptacle   Occupant     Condition
usb0/4.5                 usb-hub      connected    configured   ok
usb0/4.5.1               usb-device   connected    configured   ok
usb0/4.5.2               usb-printer  connected    configured   ok
usb0/4.5.3               usb-mouse    connected    configured   ok
usb0/4.5.4               usb-device   connected    configured   ok
usb0/4.5.5               usb-storage  connected    configured   ok
usb0/4.5.6               usb-communi  connected    configured   ok
usb0/4.5.7               unknown      empty        unconfigured ok
usb0/4.6                 usb-storage  connected    configured   ok
usb0/4.7                 usb-storage  connected    unconfigured ok
```

▼ How to Configure a USB Device

Steps

1. **Become superuser.**
2. **Configure a USB device.**

```
# cfgadm -c configure usb0/4.7
```

3. **Verify that the USB device is configured.**

```
# cfgadm usb0/4.7
Ap_Id                  Type         Receptacle  Occupant    Condition
usb0/4.7               usb-storage  connected   configured  ok
```

▼ How to Logically Disconnect a USB Device

If you want to remove a USB device from the system and the `prtconf` output, but you are not physically near the system, just logically disconnect the USB device. The device is still physically connected, but it is logically disconnected, unusable, and not visible to the system.

Steps

1. **Become superuser.**
2. **Disconnect a USB device.**

```
# cfgadm -c disconnect -y usb0/4.7
```

3. **Verify that the device is disconnected.**

```
# cfgadm usb0/4.7
Ap_Id                  Type      Receptacle    Occupant      Condition
usb0/4.7               unknown   disconnected  unconfigured  ok
```

▼ How to Logically Connect a USB Device

Use this procedure to logically connect a USB device that was previously logically disconnected or unconfigured.

Steps

1. **Become superuser.**
2. **Connect a USB device.**

```
# cfgadm -c configure usb0/4.7
```

3. **Verify that the device is connected.**

```
# cfgadm usb0/4.7
Ap_Id                  Type         Receptacle  Occupant    Condition
usb0/4.7               usb-storage  connected   configured  ok
```

The device is now available and visible to the system.

▼ How to Logically Disconnect a USB Device Subtree

Use this procedure to disconnect a USB device subtree, which is the hierarchy (or tree) of devices below a hub.

Steps

1. **Become superuser.**
2. **Remove a USB device subtree.**

```
# cfgadm -c disconnect -y usb0/4
```

3. **Verify that the USB device subtree is disconnected.**

```
# cfgadm usb0/4
Ap_Id                  Type       Receptacle   Occupant     Condition
usb0/4                 unknown    disconnected unconfigured ok
```

▼ How to Reset a USB Device

If a USB device behaves erratically, use the `cfgadm` command to reset the device, which logically removes and recreates the device.

Steps

1. **Become superuser.**
2. **Make sure the device is not in use.**
3. **Reset the device.**

   ```
   # cfgadm -x usb_reset -y usb0/4.7
   ```

4. **Verify that the device is connected.**

   ```
   # cfgadm usb0/4.7
   Ap_Id                  Type         Receptacle   Occupant     Condition
   usb0/4.7               usb-storage  connected    configured   ok
   ```

▼ How to Change the Default Configuration of a Multi-Configuration USB Device

Keep the following in mind when working with multi-configuration USB devices:

- A USB device configuration defines how a device presents itself to the operating system. This is different from system device configurations discussed in other `cfgadm` sections.
- Some USB devices support multiple configurations, but only one configuration can be active at a time.
- Multi-configuration devices can be identified by examining the `cfgadm -lv` output. *Nconfigs* will be greater than 1.
- The default USB configuration is configuration 1. The current configuration is reflected in `cfgadm -lv` output as *Config*.
- Changes to the default configuration will persist across reboots, hot-removes, and reconfiguration of the device, as long as it is reconnected to the same port.

Steps

1. **Make sure the device is not in use.**
2. **Change the default USB configuration.**

   For example:

   ```
   # cfgadm -x usb_config -o config=2 usb0/4
      Setting the device: /devices/pci@1f,0/usb@c,3:4
      to USB configuration 2
      This operation will suspend activity on the USB device
      Continue (yes/no)? yes
   ```

3. **Verify the device change.**

   For example:

```
# cfgadm -lv usb0/4
Ap_Id  Receptacle   Occupant     Condition  Information When  Type
     Busy     Phys_Id
usb0/4 connected    unconfigured ok         Mfg: Sun  2000
Product: USB-B0B0 aka Robotech
With 6 EPPS High Clk Mode   NConfigs: 7  Config: 2  : EVAL Board Setup
unavailable
usb-device   n        /devices/pci@1f,0/usb@c,3:4
```

   `Config` now shows 2.

CHAPTER 9

Accessing Devices (Overview)

This chapter provides information about how to access the devices on a system.

This is a list of the overview information in this chapter.

- "Accessing Devices" on page 151
- "Logical Disk Device Names" on page 153
- "Logical Tape Device Names" on page 156
- "Logical Removable Media Device Names" on page 157

For overview information about configuring devices, see Chapter 5.

Accessing Devices

You need to know how to specify device names when using commands to manage disks, file systems, and other devices. In most cases, you can use logical device names to represent devices that are connected to the system. Both logical and physical device names are represented on the system by logical and physical device files.

How Device Information Is Created

When a system is booted for the first time, a device hierarchy is created to represent all the devices connected to the system. The kernel uses the device hierarchy information to associate drivers with their appropriate devices, and provides a set of pointers to the drivers that perform specific operations. For more information on device hierarchy, see *OpenBoot 3.x Command Reference Manual*.

How Devices Are Managed

The devfsadm command manages the special device files in the /dev and /devices directories. By default, the devfsadm command attempts to load every driver in the system and attach to all possible device instances. Then, devfsadm creates the device files in the /devices directory and the logical links in the /dev directory. In addition to managing the /dev and /devices directories, the devfsadm command also maintains the path_to_inst(4) instance database.

Both reconfiguration boot processing and updating the /dev and /devices directories in response to dynamic reconfiguration events is handled by devfsadmd, the daemon version of the devfsadm command. This daemon is started from the /etc/rc* scripts when a system is booted.

Since the devfsadmd daemon automatically detects device configuration changes generated by any reconfiguration event, there is no need to run this command interactively.

For more information, see devfsadm(1M).

Device Naming Conventions

Devices are referenced in three ways in the Solaris environment.

- Physical device name – Represents the full device pathname in the device information hierarchy. Physical device names are displayed by using the following commands:
  - dmesg
  - format
  - sysdef
  - prtconf

  Physical device files are found in the /devices directory.
- Instance name – Represents the kernel's abbreviation name for every possible device on the system. For example, sd0 and sd1 represent the instance names of two disk devices. Instance names are mapped in the /etc/path_to_inst file and are displayed by using the following commands:
  - dmesg
  - sysdef
  - prtconf
- Logical device name – Used with most file system commands to refer to devices. For a list of file commands that use logical device names, see Table 9–1. Logical device files in the /dev directory are symbolically linked to physical device files in the /devices directory.

Logical Disk Device Names

Logical device names are used to access disk devices when you:

- Add a new disk to the system
- Move a disk from one system to another system
- Access or mount a file system residing on a local disk
- Back up a local file system

Many administration commands take arguments that refer to a disk slice or file system.

Refer to a disk device by specifying the subdirectory to which it is symbolically linked, either `/dev/dsk` or `/dev/rdsk`, followed by a string identifying the particular controller, disk, and slice.

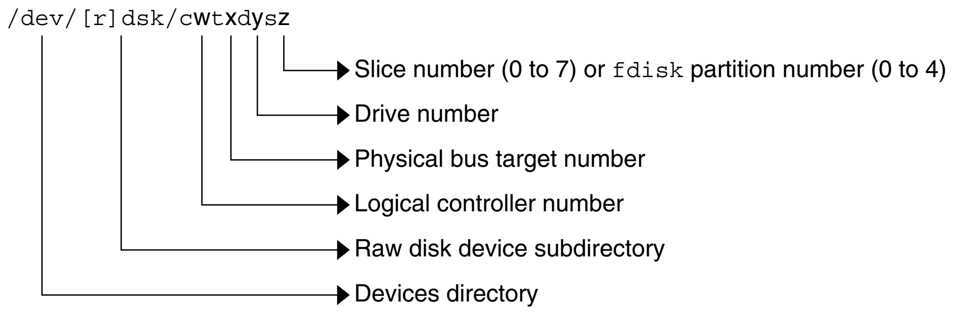
/dev/[r]dsk/cwtxdysz
Slice number (0 to 7) or fdisk partition number (0 to 4)
Drive number
Physical bus target number
Logical controller number
Raw disk device subdirectory
Devices directory

FIGURE 9–1 Logical Device Names

Specifying the Disk Subdirectory

Disk and file administration commands require the use of either a *raw* (or *character*) device interface, or a *block* device interface. The distinction is made by how data is read from the device.

Raw device interfaces transfer only small amounts of data at a time. Block device interfaces include a buffer from which large blocks of data are read at once.

Different commands require different interfaces.

- When a command requires the raw device interface, specify the `/dev/rdsk` subdirectory. (The “r” in `rdsk` stands for “raw.”)
- When a command requires the block device interface, specify the `/dev/dsk` subdirectory.

- When you are not sure whether a command requires use of `/dev/dsk` or `/dev/rdsk`, check the man page for that command.

The following table shows which interface is required for some commonly used disk and file system commands.

TABLE 9–1 Device Interface Type Required by Some Frequently Used Commands

Command	Interface Type	Example of Use
`df`(1M)	Block	`df /dev/dsk/c0t3d0s6`
`fsck`(1M)	Raw	`fsck -p /dev/rdsk/c0t0d0s0`
`mount`(1M)	Block	`mount /dev/dsk/c1t0d0s7 /export/home`
`newfs`(1M)	Raw	`newfs /dev/rdsk/c0t0d1s1`
`prtvtoc`(1M)	Raw	`prtvtoc /dev/rdsk/c0t0d0s2`

Specifying the Slice

The string that you use to identify a specific slice on a specific disk depends on the controller type, either direct or bus-oriented. The following table describes the different types of direct or bus-oriented controllers on different platforms.

TABLE 9–2 Controller Types

Direct controllers	Bus-Oriented Controllers
IDE (x86)	SCSI (SPARC/x86)
	FCAL (SPARC)
	ATA (SPARC/x86)

The conventions for both types of controllers are explained in the following subsections.

Note – Controller numbers are assigned automatically at system initialization. The numbers are strictly logical and imply no direct mapping to physical controllers.

x86: Disks With Direct Controllers

To specify a slice on a disk with an IDE controller on an x86 based system, follow the naming convention shown in the following figure.

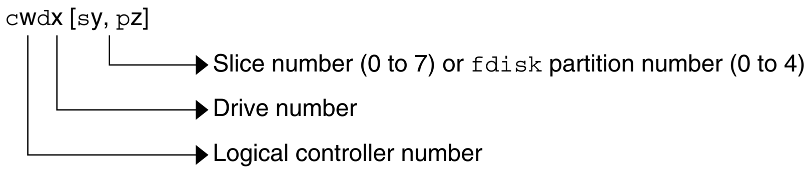
cwdx [sy, pz]
Slice number (0 to 7) or fdisk partition number (0 to 4)
Drive number
Logical controller number

FIGURE 9–2 x86: Disks with Direct Controllers

To indicate the entire Solaris `fdisk` partition, specify slice 2 (`s2`).

If you have only one controller on your system, *w* is usually 0.

SPARC: Disks With Bus-Oriented Controllers

To specify a slice on a disk with a bus-oriented controller, SCSI for instance, on a SPARC based system, follow the naming convention shown in the following figure.

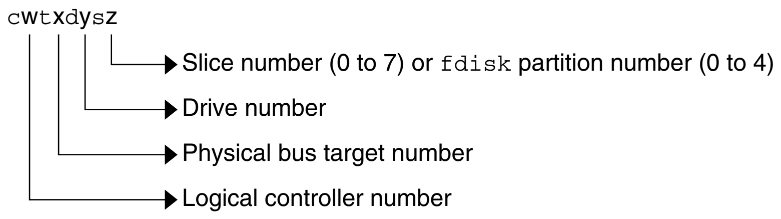
cwtxdysz
Slice number (0 to 7) or fdisk partition number (0 to 4)
Drive number
Physical bus target number
Logical controller number

FIGURE 9–3 SPARC: Disks With Bus-Oriented Controllers

On a SPARC based system with directly connected disks such as the IDE disks on a Ultra10, the naming convention is the same as that for systems with bus-oriented controllers.

If you have only one controller on your system, *w* is usually 0.

For SCSI controllers, *x* is the target address set by the switch on the back of the unit, and *y* is the logical unit number (LUN) of the drive attached to the target. If the disk has an embedded controller, *y* is usually 0.

To indicate the whole disk, specify slice 2 (`s2`).

x86: Disks With SCSI Controllers

To specify a slice on a disk with a SCSI controller on an x86 based system, follow the naming convention shown in the following figure.

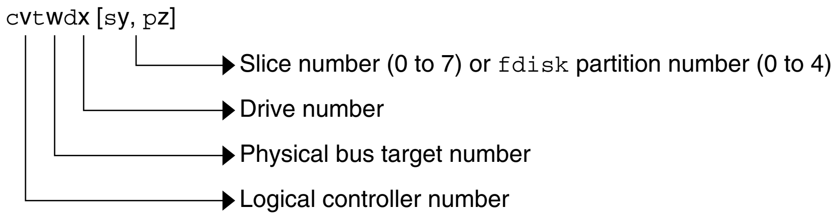
cvtwdx [sy, pz]
Slice number (0 to 7) or fdisk partition number (0 to 4)
Drive number
Physical bus target number
Logical controller number

FIGURE 9–4 x86: Disks with SCSI Controllers

If you have only one controller on your system, *v* is usually 0.

For SCSI controllers, *w* is the target address set by the switch on the back of the unit, and *x* is the logical unit number (LUN) of the drive attached to the target. If the disk has an embedded controller, *x* is usually 0.

To indicate the entire Solaris `fdisk` partition, specify slice 2 (`s2`).

Logical Tape Device Names

Logical tape device files are found in the `/dev/rmt/*` directory as symbolic links from the `/devices` directory.

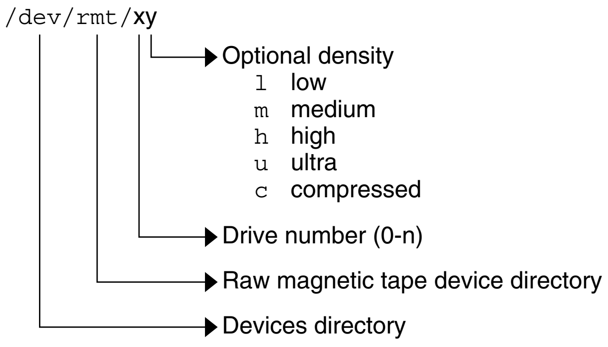

FIGURE 9–5 Logical Tape Device Names

The first tape device connected to the system is 0 (`/dev/rmt/0`). Tape density values (`l`, `m`, `h`, `c`, and `u`) are described in Chapter 28.

Logical Removable Media Device Names

Since removable media is managed by volume management (`vold`), the logical device name is usually not used unless you want to mount the media manually.

The logical device name that represents the removable media devices on a system are described in Chapter 2.

CHAPTER 10

Managing Disks (Overview)

This chapter provides overview information about Solaris disk slices and introduces the `format` utility.

This is a list of overview information in this chapter.

For instructions on how to add a disk to your system, see Chapter 12 or Chapter 13.

What's New in Disk Management in the Solaris 9 Update Releases?

This section describes new disk management features in this Solaris release.

SPARC: Multiterabyte Disk Support With EFI Disk Label

Solaris 9 4/03 – Provides support for disks that are larger than 1 terabyte on systems that run a 64-bit Solaris kernel.

You can download the EFI specification at
http://www.intel.com/technology/efi/main_specification.htm.

The EFI label provides support for physical disks and virtual disk volumes. This release also includes updated disk utilities for managing disks greater than 1 terabyte. The UFS file system is compatible with the EFI disk label, and you can create a UFS file system greater than 1 terabyte. For information on creating a multiterabyte UFS file system, see "SPARC: Support of Multiterabyte UFS File Systems" on page 243.

The unbundled Sun QFS file system is also available if you need to create file systems greater than 1 terabyte. For information on the Sun QFS file system, see http://docs.sun.com/db/doc/816-2542-10.

The Solaris Volume Manager software can also be used to manage disks greater than 1 terabyte in this Solaris release. For information on using Solaris Volume Manager, see *Solaris Volume Manager Administration Guide*.

The VTOC label is still available for disks less than 1 terabyte in size. If you are only using disks smaller than 1 terabyte on your systems, managing disks will be the same as in previous Solaris releases. In addition, you can use the `format-e` command to label a disk less than 1 terabyte with an EFI label. For more information, see Example 11–6.

Comparison of the EFI Label and the VTOC Label

The EFI disk label differs from the VTOC disk label in the following ways:

- Provides support for disks greater than 1 terabyte in size.
- Provides usable slices 0–6, where slice 2 is just another slice.
- Partitions (or slices) cannot overlap with the primary or backup label, nor with any other partitions. The size of the EFI label is usually 34 sectors, so partitions start at sector 34. This feature means no partition can start at sector zero (0).
- No cylinder, head, or sector information is stored in the label. Sizes are reported in blocks.
- Information that was stored in the alternate cylinders area, the last two cylinders of the disk, is now stored in slice 8.
- If you use the `format` utility to change partition sizes, the `unassigned` partition tag is assigned to partitions with sizes equal to zero. By default, the `format` utility assigns the `usr` partition tag to any partition with a size greater than zero. You can use the partition change menu to reassign partition tags after the partitions are changed. However, you cannot change a partition with a non-zero size to the `unassigned` partition tag.

Restrictions of the EFI Disk Label

Keep the following restrictions in mind when determining whether to use disks greater than 1 terabyte is appropriate for your environment:

- The SCSI driver, `ssd`, currently only supports up to 2 terabytes. If you need greater disk capacity than 2 terabytes, use a volume management product like Solaris Volume Manager to create a larger device.
- Layered software products intended for systems with EFI-labeled disks might be incapable of accessing a disk with an EFI disk label.
- A disk with an EFI disk label is not recognized on systems running previous Solaris releases.
- The EFI disk label is not supported on IDE disks.
- You cannot boot from a disk with an EFI disk label.
- You cannot use the Solaris Management Console's Disk Manager Tool to manage disks with EFI labels. Use the `format` utility to partition disks with EFI labels. Then, you can use the Solaris Management Console's Enhanced Storage Tool to manage volumes and disksets with EFI-labeled disks.
- The EFI specification prohibits overlapping slices. The whole disk is represented by *cxtydz*.
- Provides information about disk or partition sizes in sectors and blocks, but not in cylinders and heads.
- The following `format` options are either not supported or are not applicable on disks with EFI labels:
  - The `save` option is not supported because disks with EFI labels do not need an entry in the `format.dat` file.
  - The `backup` option is not applicable because the disk driver finds the primary label and writes it back to the disk.

Installing a System With an EFI-Labeled Disk

The Solaris installation utilities automatically recognize disks with EFI labels, but cannot use the Solaris installation utilities to repartition these disks. You must use the `format` utility to repartition this disk before or after installation. The Solaris Upgrade and Live Upgrade utilities also recognize a disk with an EFI label. However, you cannot boot a system from an EFI-labeled disk.

After the Solaris release is installed on a system with an EFI-labeled disk, the partition table looks similar to the following:

```
Current partition table (original):
Total disk sectors available: 2576924638 + 16384 (reserved sectors)

Part      Tag    Flag     First Sector          Size          Last Sector
  0       root    wm                34         1.20TB           2576924636
  1 unassigned    wm                 0            0                0
  2 unassigned    wm                 0            0                0
  3 unassigned    wm                 0            0                0
  4 unassigned    wm                 0            0                0
  5 unassigned    wm                 0            0                0
  6 unassigned    wm              0            0                0
  8   reserved    wm     2576924638        8.00MB       2576941021
```

Managing Disks With EFI Disks Labels

Use the following table to locate information on managing disks with EFI disk labels.

Task	For More Information
If the system is already installed, connect the disk to the system and perform a reconfiguration boot.	"SPARC: Adding a System Disk or a Secondary Disk (Task Map)" on page 201
Repartition the disk with the `format` utility, if necessary.	"SPARC: How to Create Disk Slices and Label a Disk" on page 204
Create disk volumes, and if needed, create soft partitions with Solaris Volume Manager.	Chapter 2, "Storage Management Concepts," in *Solaris Volume Manager Administration Guide*
Create UFS file systems for the new disk with the `newfs` command.	"SPARC: How to Create a UFS File System" on page 209
Or, create a QFS file system.	http://docs.sun.com/db/coll/20445.2

Cloning a Disk with an EFI Label

In previous Solaris releases, slice 2 (`s2`) was used to represent the whole disk. You could use the `dd` command to clone or copy disks by using syntax similar to the following:

```
dd if=/dev/rdsk/c0t0d0s2 of=/dev/rdsk/c0t2d0s2 bs=128k
```

Now, you must use a slightly different procedure to clone or copy disks larger than 1 terabyte so that the UUID of cloned disks are unique. For example:

1. Use the `dd` command to clone the disk with an EFI label:

   ```
   # dd if=/dev/rdsk/c0t0d0 of=/dev/rdsk/c0t2d0 bs=128k
   ```

2. Pipe the `prtvtoc` output of the disk to be copied to the `fmthard` command to create a new label for the cloned disk.

   ```
   # prtvtoc /dev/rdsk/c0t0d0 | fmthard -s - /dev/rdsk/c0t2d0
   ```

Caution – If you do not create a new label for the cloned disk, other software products might corrupt data on EFI-labeled disks if they encounter duplicate UUIDs.

Troubleshooting Problems With EFI Disk Labels

Use the following error messages and solutions to troubleshooting problems with EFI-labeled disks.

Error Message

```
The capacity of this LUN is too large.
Reconfigure this LUN so that it is < 2TB.
```

Cause
: You attempted to create a partition on a SCSI device that is larger than 2 terabytes.

Solution
: Create a partition on a SCSI device that is less than 2 terabytes.

Error Message

```
Dec  3 09:26:48 holoship scsi: WARNING: /sbus@a,0/SUNW,socal@d,10000/
sf@1,0/ssd@w50020f23000002a4,0 (ssd1):
Dec  3 09:26:48 holoship disk has 2576941056 blocks, which is too large
for a 32-bit kernel
```

Cause
: You attempted to boot a system running a 32-bit SPARC kernel with a disk greater than 1 terabyte.

Solution
: Boot a system running a 64-bit SPARC kernel with a disk greater than 1 terabyte.

Error Message

```
Dec  3 09:12:17 holoship scsi: WARNING: /sbus@a,0/SUNW,socal@d,10000/
sf@1,0/ssd@w50020f23000002a4,0 (ssd1):
Dec  3 09:12:17 holoship corrupt label - wrong magic number
```

Cause
: You attempted to add this disk to a system running an older Solaris release.

Solution
: Add this disk to a system running the Solaris release that supports the EFI disk label.

Where to Find Disk Management Tasks

Use these references to find step-by-step instructions for managing disks.

Disk Management Task	For More Information
Format a disk and examine a disk label	Chapter 11
Add a new disk to a SPARC system	Chapter 12
Add a new disk to an x86 system	Chapter 13
Hot-Plug a SCSI or PCI disk	Chapter 6

Overview of Disk Management

The management of disks in the Solaris environment usually involves setting up the system and running the Solaris installation program to create the appropriate disk slices and file systems and to install the operating system. Occasionally, you might need to use the `format` utility to add a new disk drive or replace a defective one.

Note – The Solaris operating system runs on two types of hardware, or platforms—SPARC and x86. The Solaris operating system runs on both 64–bit and 32–bit address spaces. The information in this document pertains to both platforms and address spaces unless called out in a special chapter, section, note, bullet, figure, table, example, or code example.

Disk Terminology

Before you can effectively use the information in this section, you should be familiar with basic disk architecture. In particular, you should be familiar with the following terms:

Disk Term	Description
Track	A concentric ring on a disk that passes under a single stationary disk head as the disk rotates.
Cylinder	The set of tracks with the same nominal distance from the axis about which the disk rotates.
Sector	Section of each disk platter. A sector holds 512 bytes.
Block	A data storage area on a disk. A disk block is 512 bytes.
Disk controller	A chip and its associated circuitry that controls the disk drive.
Disk label	The first sector of a disk that contains disk geometry and partition information.
Device driver	A device driver is a kernel module that controls a hardware or virtual device.

For additional information, see the product information from your disk's manufacturer.

About Disk Slices

Files stored on a disk are contained in file systems. Each file system on a disk is assigned to a *slice*, which is a group of sectors set aside for use by that file system. Each disk slice appears to the operating system (and to the system administrator) as though it were a separate disk drive.

For information about file systems, see Chapter 15.

Note – Slices are sometimes referred to as partitions. This book uses *slice* but certain interfaces, such as the `format` utility, refer to slices as partitions.

When setting up slices, remember these rules:

- Each disk slice holds only one file system.
- No file system can span multiple slices.

Slices are set up slightly differently on SPARC and x86 platforms. The following table summarizes the differences.

TABLE 10–1 Slice Differences on Platforms

SPARC Platform	x86 Platform
Whole disk is devoted to Solaris environment.	Disk is divided into `fdisk` partitions, one `fdisk` partition per operating system.
VTOC – Disk is divided into 8 slices, numbered 0–7.	**VTOC** – The Solaris `fdisk` partition is divided into 10 slices, numbered 0–9.
EFI – Disk is divided into 7 slices, numbered 0–6.	

Solaris Volume Manager, previously Solstice DiskSuite™, has a partitioning feature, *soft partitioning*, that enables more than eight partitions per disk.

For general information about Solaris Volume Manager, see Chapter 2, "Storage Management Concepts," in *Solaris Volume Manager Administration Guide*. For information on soft partitioning, see Chapter 12, "Soft Partitions (Overview)," in *Solaris Volume Manager Administration Guide*.

SPARC: Disk Slices

The following table describes the slices on a SPARC based system.

TABLE 10–2 SPARC: Customary Disk Slices

Slice	File System	Usually Found on Client or Server Systems?	Comments
0	root (/)	Both	Holds files and directories that make up the operating system. **EFI** – You cannot boot from a disk with an EFI label.
1	swap	Both	Provides virtual memory, or *swap space*.
2	—	Both	**VTOC** – Refers to the entire disk, by convention. The size of this slice should not be changed. **EFI** – Optional slice to be defined based on your site's needs.

TABLE 10–2 SPARC: Customary Disk Slices *(Continued)*

Slice	File System	Usually Found on Client or Server Systems?	Comments
3	`/export`	Both	Optional slice that can be defined based on your site's needs. Can be used on a server to hold alternative versions of operating systems that are required by client systems.
4		Both	Optional slice to be defined based on your site's needs.
5		Both	Optional slice to be defined based on your site's needs. Can be used to hold application software added to a system. If a slice is not allocated for the `/opt` file system during installation, the `/opt` directory is put in slice 0.
6	`/usr`	Both	Holds operating system commands (also known as *executables*). This slice also holds documentation, system programs (`init` and `syslogd`, for example) and library routines.
7	`/home` or `/export/home`	Both	**VTOC** – Holds files that are created by users. **EFI** – Not applicable.
8	N/A	N/A	**VTOC** – Not applicable. **EFI** – A reserved slice created by default. This area is similar to the VTOC's alternate cylinders. Do not modify nor delete this slice.

x86: Disk Slices

On x86 based systems, disks are divided into `fdisk` partitions. An `fdisk` partition is a section of the disk that reserved for a particular operating system, such as the Solaris release.

The Solaris release places ten slices, numbered 0–9, on a Solaris `fdisk` partition as shown in the following table.

TABLE 10–3 x86: Customary Disk Slices

Slice	File System	Usually Found on Client or Server Systems?	Purpose
0	root (/)	Both	Holds the files and directories that make up the operating system.
1	swap	Both	Provides virtual memory, or *swap space*.
2	—	Both	Refers to the entire disk, by convention. The size of this slice should not be changed.
3	`/export`	Both	Optional slice to be defined based on your site's needs. Can be used on a server to hold alternative versions of operating systems that are required by client systems.
4			Optional slice to be defined based on your site's needs.
5		Both	Optional slice to be defined based on your site's needs. Can be used to hold application software added to a system. If a slice is not allocated for the `/opt` file system during installation, the `/opt` directory is put in slice 0.
6	`/usr`	Both	Holds operating system commands (also known as *executables*). This slice also holds documentation, system programs (`init` and `syslogd`, for example) and library routines.
7	`/home` or `/export/home`	Both	Holds files that are created by users.
8	—	Both	Contains information necessary for to boot the Solaris environment from the hard disk. The slice resides at the beginning of the Solaris `fdisk` partition (although the slice number itself does not indicate this fact), and is known as the boot slice.

TABLE 10–3 x86: Customary Disk Slices *(Continued)*

Slice	File System	Usually Found on Client or Server Systems?	Purpose
9	—	Both	Provides an area that is reserved for alternate disk blocks. Slice 9 is known as the alternate sector slice.

Using Raw Data Slices

The SunOS operating system stores the disk label in block 0 of each disk. So, third-party database applications that create raw data slices must not start at block 0, or the disk label will be overwritten and the data on the disk will be inaccessible.

Do not use the following areas of the disk for raw data slices, which are sometimes created by third-party database applications:

- Block 0 where the disk label is stored
- Slice 2, which represents the entire disk with a VTOC label

Slice Arrangements on Multiple Disks

Although a single large disk can hold all slices and their corresponding file systems, two or more disks are often used to hold a system's slices and file systems.

Note – A slice cannot be split between two or more disks. However, multiple swap slices on separate disks are allowed.

For instance, a single disk might hold the root (`/`) file system, a swap area, and the `/usr` file system, while another disk holds the `/export/home` file system and other file systems that contain user data.

In a multiple disk arrangement, the disk that contains the operating system software and swap space (that is, the disk that holds the root (`/`) and `/usr` file systems and the slice for swap space) is called the *system disk*. Other disks are called *secondary disks* or *non-system disks*.

When you arrange a system's file systems on multiple disks, you can modify file systems and slices on the secondary disks without having to shut down the system or reload operating system software.

When you have more than one disk, you also increase input-output (I/O) volume. By distributing disk load across multiple disks, you can avoid I/O bottlenecks.

Determining Which Slices to Use

When you set up a disk's file systems, you choose not only the size of each slice, but also which slices to use. Your decisions about these matters depend on the configuration of the system to which the disk is attached and the software you want to install on the disk.

System configurations that need disk space are as follows:

- Servers
- Standalone systems

Each system configuration can use slices in a different way. The following table lists some examples.

TABLE 10–4 System Configurations and Slices

Slice	Servers	Standalone Systems
0	root	root
1	swap	swap
2	—	—
3	`/export`	—
6	`/usr`	`/usr`
7	`/export/home`	`/home`

For more information about system configurations, see "Overview of System Types" in *System Administration Guide: Basic Administration*.

Note – The Solaris installation program provides default slice sizes based on the software you select for installation.

The `format` Utility

Read the following overview of the `format` utility and its uses before proceeding to the "how-to" or reference sections.

The `format` utility is a system administration tool that is used to prepare hard disk drives for use on your Solaris system.

The following table shows the features and associated benefits that the `format` utility provides.

TABLE 10–5 Features and Benefits of the `format` Utility

Feature	Benefit
Searches your system for all attached disk drives	Reports on the following: ■ Target location ■ Disk geometry ■ Whether the disk is formatted ■ If the disk has mounted partitions
Retrieves disk labels	Convenient for repair operations
Repairs defective sectors	Allows administrators to repair disk drives with recoverable errors instead of sending the drive back to the manufacturer
Formats and analyzes a disk	Creates sectors on the disk and verifies each sector
Partitions a disk	Divides a disk into slices so individual file systems can be created on separate slices
Labels a disk	Writes disk name and configuration information to the disk for future retrieval (usually for repair operations)

The `format` utility options are fully described in Chapter 14.

When to Use the `format` Utility

Disk drives are partitioned and labeled by the Solaris installation program when you install the Solaris release. You can use the `format` utility to do the following:

- Display slice information
- Divide a disk into slices
- Add a disk drive to an existing system
- Format a disk drive
- Label a disk
- Repair a disk drive
- Analyze a disk for errors

The main reason a system administrator uses the `format` utility is to divide a disk into disk slices. These steps are covered in Chapter 12 and Chapter 13.

See the following section for guidelines on using the `format` utility.

Guidelines for Using the `format` Utility

TABLE 10–6 The `format` Utility Guidelines

Task	Guidelines	For More Information
Format a disk	■ Any existing data is destroyed when you reformat a disk. ■ The need for formatting a disk drive has dropped as more and more manufacturers ship their disk drives formatted and partitioned. You might not need to use the `format` utility when you add a disk drive to an existing system. ■ If a disk has been relocated and is displaying a lot of disk errors, you can attempt to reformat it, which will automatically remap any bad sectors.	"How to Format a Disk" on page 183
Replace a system disk	■ Data from the damaged system disk must be restored from a backup medium. Otherwise, the system will have to be reinstalled by using the installation program.	"SPARC: How to Connect a System Disk and Boot" on page 202 or "x86: How to Connect a System Disk and Boot" on page 212 or, if the system must be reinstalled, *Solaris 9 9/04 Installation Guide*
Divide a disk into slices	■ Any existing data is destroyed when you repartition and relabel a disk with existing slices. ■ Existing data must be copied to backup media before the disk is repartitioned and restored.	"SPARC: How to Create Disk Slices and Label a Disk" on page 204 or "x86: How to Create Disk Slices and Label a Disk" on page 220
Add a secondary disk to an existing system	■ Any existing data must be restored from backup media if the secondary disk is reformatted or repartitioned.	"SPARC: How to Connect a Secondary Disk and Boot" on page 203 or "x86: How to Connect a Secondary Disk and Boot" on page 213

TABLE 10–6 The `format` Utility Guidelines *(Continued)*

Task	Guidelines	For More Information
Repair a disk drive	■ Some customer sites prefer to replace rather than repair defective drives. If your site has a repair contract with the disk drive manufacturer, you might not need to use the `format` utility to repair disk drives. ■ The repair of a disk drive usually means that a bad sector is added to a defect list. New controllers remap bad sectors automatically with no system interruption. ■ If the system has an older controller, you might need to remap a bad sector and restore any lost data.	"Repairing a Defective Sector" on page 197

Formatting a Disk

In most cases, disks are formatted by the manufacturer or reseller. So, they do not need to be reformatted when you install the drive. To determine if a disk is formatted, use the `format` utility. For more information, see "How to Determine if a Disk is Formatted" on page 182.

If you determine that a disk is not formatted, use the `format` utility to format the disk.

When you format a disk, you accomplishes two steps:

- The disk media is prepared for use
- A list of disk defects based on a surface analysis is compiled

Caution – Formatting a disk is a destructive process because it overwrites data on the disk. For this reason, disks are usually formatted only by the manufacturer or reseller. If you think disk defects are the cause of recurring problems, you can use the `format` utility to do a surface analysis. However, be careful to use only the commands that do not destroy data. For details, see "How to Format a Disk" on page 183.

A small percentage of total disk space that is available for data is used to store defect and formatting information. This percentage varies according to disk geometry, and decreases as the disk ages and develops more defects.

Formatting a disk might take anywhere from a few minutes to several hours, depending on the type and size of the disk.

About Disk Labels

A special area of every disk is set aside for storing information about the disk's controller, geometry, and slices. That information is called the disk's *label*. Another term that is used to described the disk label is the VTOC (Volume Table of Contents) on a disk with a VTOC label. To *label* a disk means to write slice information onto the disk. You usually label a disk after you change its slices.

If you fail to label a disk after you create slices, the slices will be unavailable because the operating system has no way of "knowing" about the slices.

Partition Table

An important part of the disk label is the *partition table*, which identifies a disk's slices, the slice boundaries (in cylinders), and the total size of the slices. You can display a disk's partition table by using the `format` utility. The following table describes partition table terminology.

TABLE 10–7 Partition Table Terminology

Partition Term	Value	Description
Number	`0-7`	**VTOC** – Partitions or slices, numbered 0–7. **EFI** – Partitions or slices, numbered 0–6.
Tag	`0=UNASSIGNED 1=BOOT 2=ROOT 3=SWAP 4=USR 5=BACKUP 7=VAR 8=HOME 11=RESERVED`	A numeric value that usually describes the file system mounted on this partition.
Flags	`wm`	The partition is writable and mountable.
	`wu rm`	The partition is writable and unmountable. This is the default state of partitions that are dedicated for swap areas. (However, the `mount` command does not check the "not mountable" flag.)
	`rm`	The partition is read only and mountable.

Partition flags and tags are assigned by convention and require no maintenance.

For more information on displaying the partition table, see "How to Display Disk Slice Information" on page 185 or "How to Examine a Disk Label" on page 189.

Displaying Partition Table Information

The following is an example of a partition table from a 4.0-Gbyte disk with a VTOC label displayed from the `format` utility:

```
Total disk cylinders available: 8892 + 2 (reserved cylinders)

Part      Tag    Flag     Cylinders        Size            Blocks
  0       root    wm    1110 - 4687        1.61GB    (0/3578/0) 3381210
  1       swap    wu       0 - 1109      512.00MB    (0/1110/0) 1048950
  2     backup    wm       0 - 8891        4.01GB    (0/8892/0) 8402940
  3 unassigned    wm       0               0         (0/0/0)          0
  4 unassigned    wm       0               0         (0/0/0)          0
  5 unassigned    wm       0               0         (0/0/0)          0
  6 unassigned    wm       0               0         (0/0/0)          0
  7       home    wm    4688 - 8891        1.89GB    (0/4204/0) 3972780
```

The partition table displayed by the `format` utility contains the following information:

Column Name	Description
`Part`	Partition (or slice number). See Table 10–7 for a description of this column.
`Tag`	Partition tag. See Table 10–7 for a description of this column.
`Flags`	Partition flag. See Table 10–7 for a description of this column.
`Cylinders`	The starting and ending cylinder number for the slice.
`Size`	The slice size in Mbytes.
`Blocks`	The total number of cylinders and the total number of sectors per slice in the far right column.
First Sector	**EFI** – The starting block number.
Last Sector	**EFI** – The ending block number.

The following is an example of a EFI disk label displayed by using the `prtvtoc` command.

```
# prtvtoc /dev/rdsk/c4t1d0s0
* /dev/rdsk/c4t1d0s0 partition map
*
* Dimensions:
*     512 bytes/sector
* 2576941056 sectors
* 2576940989 accessible sectors
*
* Flags:
*   1: unmountable
*  10: read-only
```

```
*
*                            First      Sector     Last
* Partition  Tag  Flags      Sector      Count     Sector   Mount Directory
       0      2    00           34   629145600  629145633
       1      4    00    629145634   629145600 1258291233
       6      4    00   1258291234  1318633404 2576924637
       8     11    00   2576924638       16384 2576941021
* Flags:
*   1: unmountable
*  10: read-only
*
```

The `prtvtoc` command provides the following information:

Column Name	Description
`Dimensions`	This section describes the physical dimensions of the disk drive.
`Flags`	This section describes the flags listed in the partition table section. For a description of partition flags, see Table 10–7.
`Partition` (or Slice) Table	This section contains the following information:
`Partition`	Partition (or slice number). For a description of this column, see Table 10–7.
`Tag`	Partition tag. For a description of this column, see Table 10–7.
`Flags`	Partition flag. For a description of this column, see Table 10–7.
`First Sector`	The first sector of the slice.
`Sector Count`	The total number of sectors in the slice.
`Last Sector`	The last sector of the slice.
`Mount Directory`	The last mount point directory for the file system.

Dividing a Disk Into Slices

The `format` utility is most often used by system administrators to divide a disk into slices. The steps are as follows:

- Determining which slices are needed
- Determining the size of each slice
- Using the `format` utility to divide the disk into slices
- Labeling the disk with new slice information
- Creating the file system for each slice

The easiest way to divide a disk into slices is to use the `modify` command from the `partition` menu of the `format` utility. The `modify` command allows you to create slices by specifying the size of each slice without having to keep track of the starting cylinder boundaries. The `modify` command also keeps tracks of any disk space that remains in the “free hog” slice.

Using the Free Hog Slice

When you use the `format` utility to change the size of one or more disk slices, you designate a temporary slice that will expand and shrink to accommodate the resizing operations.

This temporary slice donates, or “frees,” space when you expand a slice, and receives, or “hogs,” the discarded space when you shrink a slice. For this reason, the donor slice is sometimes called the *free hog*.

The free hog slice exists only during installation or when you run the `format` utility. There is no permanent free hog slice during day-to-day operations.

For information on using the free hog slice, see “SPARC: How to Create Disk Slices and Label a Disk” on page 204 or “x86: How to Create Disk Slices and Label a Disk” on page 220.

CHAPTER 11

Administering Disks (Tasks)

This chapter contains disk administration procedures. Many procedures described in this chapter are optional if you are already familiar with how disks are managed on systems running the Solaris release.

For information on the procedures associated with administering disks, see "Administering Disks (Task Map)" on page 179.

For overview information about disk management, see Chapter 10.

Administering Disks (Task Map)

Task	Description	For Instructions
Identify the disks on a system	If you are not sure of the types of disks on a system, use the `format` utility to identify the disk types.	"How to Identify the Disks on a System" on page 180
Format the disk	Determine whether a disk is already formatted by using the `format` utility.	"How to Determine if a Disk is Formatted" on page 182
	In most cases, disks are already formatted. Use the `format` utility if you need to format a disk.	"How to Format a Disk" on page 183
Display slice information	Display slice information by using the `format` utility.	"How to Display Disk Slice Information" on page 185

Task	Description	For Instructions
Label the disk	Create the disk label by using the `format` utility.	"How to Label a Disk" on page 187
Examine the disk label	Examine the disk label by using the `prtvtoc` command.	"How to Examine a Disk Label" on page 189
Recover a corrupted disk label	You can attempt to recover a disk label that was damaged due to a system or power failure.	"How to Recover a Corrupted Disk Label" on page 191
Create a `format.dat` entry	Create a `format.dat` entry to support a third-party disk.	"How to Create a `format.dat` Entry" on page 194
Automatically configure a SCSI disk	You can automatically configure a SCSI disk with the SCSI-2 specification for disk device mode sense pages even if the specific drive type is not listed in the `/etc/format.dat` file.	"How to Automatically Configure a SCSI Drive" on page 195
Repair a defective disk sector	Identify a defective disk sector by using the `format` utility.	"How to Identify a Defective Sector by Using Surface Analysis" on page 197
If necessary, fix a defective disk sector	Fix a defective disk sector by using the `format` utility.	"How to Repair a Defective Sector" on page 198

Identifying Disks on a System

Use the `format` utility to discover the types of disks that are connected to a system. You can also use the `format` utility to verify that a disk is known to the system. For information on using the `format` utility, see Chapter 14.

▼ How to Identify the Disks on a System

Steps

1. **Become superuser or assume an equivalent role.**

2. **Identify the disks that are recognized on the system with the `format` utility.**

```
# format
```

The format utility displays a list of disks that it recognizes under `AVAILABLE DISK SELECTIONS`.

Example 11–1 Identifying the Disks on a System

The following `format` output is from a system with one disk.

```
# format
Searching for disks...done


AVAILABLE DISK SELECTIONS:
       0. c0t0d0 <ST34321A cyl 8892 alt 2 hd 15 sec 63>
          /pci@1f,0/pci@1,1/ide@3/dad@0,0
Specify disk (enter its number):
```

The `format` output associates a disk's physical and logical device name to the disk's marketing name, which appears in angle brackets <>. See the example below. This method is an easy way to identify which logical device names represent the disks that are connected to your system. For a description of logical and physical device names, see Chapter 9.

The following example uses a wildcard to display the disks that are connected to a second controller.

```
# format /dev/rdsk/c2*
AVAILABLE DISK SELECTIONS:
       0. /dev/rdsk/c2t10d0s0 <SUN9.0G cyl 4924 alt 2 hd 27 sec 133>
          /sbus@3,0/SUNW,fas@3,8800000/sd@a,0
       1. /dev/rdsk/c2t11d0s0 <SUN9.0G cyl 4924 alt 2 hd 27 sec 133>
          /sbus@3,0/SUNW,fas@3,8800000/sd@b,0
       2. /dev/rdsk/c2t14d0s0 <SUN18G cyl 7506 alt 2 hd 19 sec 248>
          /sbus@3,0/SUNW,fas@3,8800000/sd@e,0
       3. /dev/rdsk/c2t15d0s0 <SUN18G cyl 7506 alt 2 hd 19 sec 248>
          /sbus@3,0/SUNW,fas@3,8800000/sd@f,0
Specify disk (enter its number):
```

The following example identifies the disks on a SPARC based system.

```
# format
0. c0t3d0 <SUN2.1G cyl 2733 alt 2 hd 19 sec 80>
   /iommu@0,10000000/sbus@0,10001000/espdma@5,8400000/esp@5,8800000/sd@3,0
Specify disk (enter its number):
```

The `format` output identifies that disk 0 (target 3) is connected to the first SCSI host adapter (`espdma@...`), which is connected to the first SBus device (`sbus@0...`). The output also associates both the physical and logical device name to the disk's marketing name, `SUN2.1G`.

The following example shows how to identify the disks on an x86 based system.

```
# format
AVAILABLE DISK SELECTIONS:
  0. c0d0 <DEFAULT cyl 615 alt 2 hd 64 sec 63>
```

```
     /pci@0,0/pci-ide@7,1/ata@0/cmdk@0,0
  1. c0d1 <DEFAULT cyl 522 alt 2 hd 32 sec 63>
     /pci@0,0/pci-ide@7,1/ata@0/cmdk@1,0
  2. c1d0 <DEFAULT cyl 817 alt 2 hd 256 sec 63>
     /pci@0,0/pci-ide@7,1/ata@1/cmdk@0,0
Specify disk (enter its number):
```

The `format` output identifies that disk 0 is connected to the first PCI host adapter (`pci-ide@7...`), which is connected to the ATA device (`ata...`). The `format` output on an x86 based system does not identify disks by their marketing names.

See Also Check the following table if the `format` utility did not recognize a disk.

- Go to Chapter 12 or Chapter 13.
- Go to "Creating a `format.dat` Entry" on page 194.
- Go to "How to Label a Disk" on page 187.
- Connect the disk to the system by using your disk hardware documentation.

Formatting a Disk

Disks are formatted by the manufacturer or reseller. They usually do not need to be reformatted when you install the drive.

A disk must be formatted before you can do the following:

- Write data to it. However, most disks are already formatted.
- Use the Solaris installation program to install the system.

Caution – Formatting a disk is a destructive process because it overwrites data on the disk. For this reason, disks are usually formatted only by the manufacturer or reseller. If you think disk defects are the cause of recurring problems, you can use the `format` utility to do a surface analysis. However, be careful to use only the commands that do not destroy data.

▼ How to Determine if a Disk is Formatted

Steps

1. **Become superuser or assume an equivalent role.**

2. **Invoke the `format` utility.**

   ```
   # format
   ```

3. **Type the number of the disk that you want to check from the list displayed on your screen.**

```
Specify disk (enter its number): 0
```

4. **Verify that the disk you chose is formatted by noting the following message.**

   ```
   [disk formatted]
   ```

Example 11–2 Determining if a Disk Is Formatted

The following example shows that disk `c1t0d0` is formatted.

```
# format /dev/rdsk/c1*
AVAILABLE DISK SELECTIONS:
       0. /dev/rdsk/c1t0d0s0 <SUN18G cyl 7506 alt 2 hd 19 sec 248>
          /sbus@2,0/QLGC,isp@2,10000/sd@0,0
       1. /dev/rdsk/c1t1d0s0 <SUN18G cyl 7506 alt 2 hd 19 sec 248>
          /sbus@2,0/QLGC,isp@2,10000/sd@1,0
       2. /dev/rdsk/c1t8d0s0 <SUN18G cyl 7506 alt 2 hd 19 sec 248>
          /sbus@2,0/QLGC,isp@2,10000/sd@8,0
       3. /dev/rdsk/c1t9d0s0 <SUN18G cyl 7506 alt 2 hd 19 sec 248>
          /sbus@2,0/QLGC,isp@2,10000/sd@9,0
Specify disk (enter its number): 0
selecting /dev/rdsk/c1t0d0s0
[disk formatted]
```

▼ How to Format a Disk

Steps

1. **Become superuser or assume an equivalent role.**

2. **Invoke the `format` utility.**

   ```
   # format
   ```

3. **Type the number of the disk that you want to format from the list displayed on your screen.**

   ```
   Specify disk (enter its number): 0
   ```

   Caution – Do not select the system disk. If you format your system disk, you delete the operating system and any data on this disk.

4. **To begin formatting the disk, type `format` at the `format>` prompt. Confirm the command by typing `y`.**

   ```
   format> format
   Ready to format.  Formatting cannot be interrupted
   and takes 23 minutes (estimated). Continue? yes
   ```

5. **Verify that the disk format is successful by noting the following messages.**

```
Beginning format. The current time Tue ABC xx xx:xx:xx xxxx

Formatting...
done

Verifying media...
        pass 0 - pattern = 0xc6dec6de
   2035/12/18

        pass 1 - pattern = 0x6db6db6d
   2035/12/18

Total of 0 defective blocks repaired.
```

Example 11–3 Formatting a Disk

The following example shows how to format the disk `c0t3d0`.

```
# format
Searching for disks...done


AVAILABLE DISK SELECTIONS:
       0. c0t0d0 <SUNW18G cyl 7506 alt 2 hd 19 sec 248
          /pci@1f,0/pci@1,1/scsi@2/sd@0,0
       1. c0t1d0 <FUJITSU  MAN3367M SUN36G  1804 43d671f>
          /pci@1f,0/pci@1,1/scsi@2/sd@1,0
       2. c0t2d0 <FUJITSU  MAN3367M SUN36G  1804 43d671f>
          /pci@1f,0/pci@1,1/scsi@2/sd@2,0
       3. c0t3d0 <FUJITSU  MAN3367M SUN36G  1804 43d671f>
          /pci@1f,0/pci@1,1/scsi@2/sd@3,0
       4. c0t4d0 <FUJITSU  MAN3367M SUN36G  1804 43d671f>
          /pci@1f,0/pci@1,1/scsi@2/sd@4,0
       5. c0t5d0 <FUJITSU  MAN3367M SUN36G  1804 43d671f>
          /pci@1f,0/pci@1,1/scsi@2/sd@5,0
       6. c0t6d0 <FUJITSU  MAN3367M SUN36G  1804 43d671f>
          /pci@1f,0/pci@1,1/scsi@2/sd@6,0
Specify disk (enter its number): 6
selecting c0t6d0
[disk formatted]
format> format
Ready to format.  Formatting cannot be interrupted
and takes 332 minutes (estimated). Continue? y
Beginning format. The current time is Wed Jan  7 16:16:05 2004

Formatting...
   99% complete (00:00:21 remaining) done

Verifying media...
        pass 0 - pattern = 0xc6dec6de
   71132922

        pass 1 - pattern = 0x6db6db6d
   71132922
```

```
Total of 0 defective blocks repaired.
format> quit
```

Displaying Disk Slices

You can use the `format` utility to check whether a disk has the appropriate disk slices. If you determine that a disk does not contain the slices you want to use, use the `format` utility to re-create them and label the disk. For information on creating disk slices, see "SPARC: How to Create Disk Slices and Label a Disk" on page 204 or "x86: How to Create Disk Slices and Label a Disk" on page 220.

Note – The `format` utility uses the term *partition* instead of *slice*.

▼ How to Display Disk Slice Information

Steps

1. **Become superuser or assume an equivalent role.**
2. **Invoke the `format` utility.**

   ```
   # format
   ```

3. **Type the number of the disk for which you want to display slice information from the list displayed on your screen.**

   ```
   Specify disk (enter its number):1
   ```

4. **Select the partition menu.**

   ```
   format> partition
   ```

5. **Display the slice information for the current disk drive.**

   ```
   partition> print
   ```

6. **Exit the `format` utility.**

   ```
   partition> q
   format> q
   #
   ```

7. **Verify the displayed slice information by identifying specific slice tags and slices.**

If the screen output shows that no slice sizes are assigned, the disk probably does not have slices.

Example 11–4 Displaying Disk Slice Information

The following example displays slice information for disk with a VTOC label.

```
# format
Searching for disks...done
Specify disk (enter its number):1
Selecting c0t0d0
format> partition
partition> print
Current partition table (original):
Total disk cylinders available: 8892 + 2 (reserved cylinders)

Part      Tag    Flag     Cylinders        Size            Blocks
  0       root    wm    1110 - 4687        1.61GB    (0/3578/0) 3381210
  1       swap    wu       0 - 1109      512.00MB    (0/1110/0) 1048950
  2     backup    wm       0 - 8891        4.01GB    (0/8892/0) 8402940
  3 unassigned    wm       0               0         (0/0/0)          0
  4 unassigned    wm       0               0         (0/0/0)          0
  5 unassigned    wm       0               0         (0/0/0)          0
  6 unassigned    wm       0               0         (0/0/0)          0
  7       home    wm    4688 - 8891        1.89GB    (0/4204/0) 3972780
partition> q
format> q
#
```

For a detailed description of the slice information in these examples, see Chapter 10.

The following example shows the slice information on a disk with an EFI label.

```
# format
Searching for disks...done
Specify disk (enter its number): 9
selecting c4t1d0
[disk formatted]
format> partition
partition> print
Current partition table (original):
partition> q
format> q
Part      Tag    Flag     First Sector         Size          Last Sector
  0       root    wm                34       300.00GB         629145633
  1        usr    wm         629145634       300.00GB        1258291233
  2 unassigned    wm                 0              0             0
  3 unassigned    wm                 0              0             0
  4 unassigned    wm                 0              0             0
  5 unassigned    wm                 0              0             0
  6        usr    wm        1258291234       628.77GB        2576924637
  8   reserved    wm        2576924638         8.00MB        2576941021
```

Creating and Examining a Disk Label

The labeling of a disk is usually done during system installation or when you are creating new disk slices. You might need to relabel a disk if the disk label becomes corrupted (for example, from a power failure).

The `format` utility attempts to automatically configure any unlabeled SCSI disk. If the `format` utility is able to automatically configure an unlabeled disk, it displays a message like the following:

```
c0t0d1: configured with capacity of 4.00GB
```

Tip – For information on labeling multiple disks with the same disk label, see "Label Multiple Disks by Using the `prtvtoc` and `fmthard` Commands" on page 200.

▼ How to Label a Disk

You can use the following procedure to label at disk with a VTOC label or a disk greater than 1 terabyte with an EFI label. If you want to put an EFI label on disk smaller than 1 terabyte, see Example 11–6.

Steps

1. **Become superuser or assume an equivalent role.**

2. **Invoke the `format` utility.**

   ```
   # format
   ```

3. **Type the number of the disk that you want to label from the list displayed on your screen.**

   ```
   Specify disk (enter its number):1
   ```

4. **Select one of the following.**

   a. **If the disk is unlabeled and was successfully configured, go to step 5 to label the disk.**

      The `format` utility will ask if you want to label the disk.

   b. **If the disk is labeled and you want to change the disk type, or if the `format` utility was not able to automatically configure the disk, follow steps 6-7 to set the disk type and label the disk.**

5. **Label the disk by typing `y` at the `Label it now?` prompt.**

   ```
   Disk not labeled. Label it now? y
   ```

The disk is now labeled. Go to step 10 to exit the `format` utility.

6. **Enter `type` at the `format>` prompt.**

   ```
   format> type
   ```

   The Available Drive Types menu is displayed.

7. **Select a disk type from the list of possible disk types.**

   ```
   Specify disk type (enter its number)[12]: 12
   ```

   Or, select `0` to automatically configure a SCSI-2 disk. For more information, see "How to Automatically Configure a SCSI Drive" on page 195.

8. **Label the disk. If the disk is not labeled, the following message is displayed.**

   ```
   Disk not labeled. Label it now? y
   ```

   Otherwise, you are prompted with this message:

   ```
   Ready to label disk, continue? y
   ```

9. **Verify the disk label.**

   ```
   format> verify
   ```

10. **Exit the `format` utility.**

    ```
    partition> q
    format> q
    #
    ```

Example 11–5 Labeling a Disk

The following example shows how to automatically configure and label a 1.05-Gbyte disk.

```
# format
    c1t0d0: configured with capacity of 1002.09MB

AVAILABLE DISK SELECTIONS:
      0. c0t3d0 <SUN1.05 cyl 2036 alt 2 hd 14 sec 72>
     /iommu@f,e0000000/sbus@f,e0001000/espdma@f,400000/esp@f,800000/sd@1,0
      1. c1t0d0 <SUN1.05 cyl 2036 alt 2 hd 14 sec 72>
     /iommu@f,e0000000/sbus@f,e0001000/espdma@f,400000/esp@f,800000/sd@1,0
Specify disk (enter its number): 1
Disk not labeled.  Label it now?  yes
format> verify
#
```

Example 11–6 Labeling a Disk Less Than 1 Terabyte with an EFI Label

The following example shows how to use the `format -e` command to label a disk less than 1 terabyte with an EFI label. Remember to verify that your layered software products will continue to work on systems with EFI-labeled disks. For general information on EFI label restrictions, see "Restrictions of the EFI Disk Label" on page 160.

```
# format -e
Searching for disks...done
AVAILABLE DISK SELECTIONS:
       1. c1t0d0 <SUNW18g cyl 7506 alt 2 hd 19 sec 248>
          /sbus@2,0/QLGC,isp@2,10000/sd@0,0
       2. c1t1d0 <SUNW18g cyl 7506 alt 2 hd 19 sec 248>
          /sbus@2,0/QLGC,isp@2,10000/sd@1,0
       3. c1t8d0 <SUNW18g cyl 7506 alt 2 hd 19 sec 248>
          /sbus@2,0/QLGC,isp@2,10000/sd@8,0
       4. c1t9d0 <SUNW18g cyl 7506 alt 2 hd 19 sec 248>
          /sbus@2,0/QLGC,isp@2,10000/sd@9,0
Specify disk (enter its number): 4
selecting c1t9d0
[disk formatted]
format> label
[0] SMI Label
[1] EFI Label
Specify Label type[0]: 1
Ready to label disk, continue? yes
format> quit
```

▼ How to Examine a Disk Label

Examine disk label information by using the `prtvtoc` command. For a detailed description of the disk label and the information that is displayed by the `prtvtoc` command, see Chapter 10.

Steps

1. **Become superuser or assume an equivalent role.**

2. **Display the disk label information.**

   ```
   # prtvtoc /dev/rdsk/device-name
   ```

   device-name is the raw disk device you want to examine.

Example 11–7 Examining a Disk Label

The following example shows the disk label information for disk with a VTOC label.

```
# prtvtoc /dev/rdsk/c0t0d0s0
* /dev/rdsk/c0t0d0s0 partition map
*
* Dimensions:
```

```
*      512 bytes/sector
*       63 sectors/track
*       15 tracks/cylinder
*      945 sectors/cylinder
*     8894 cylinders
*     8892 accessible cylinders
*
* Flags:
*    1: unmountable
*   10: read-only
*
*                           First     Sector    Last
* Partition  Tag  Flags    Sector      Count    Sector  Mount Directory
       0      2    00    1048950   3381210   4430159   /
       1      3    01          0   1048950   1048949
       2      5    00          0   8402940   8402939
       7      8    00    4430160   3972780   8402939   /export/home
```

The following example shows the disk label information for disk with an EFI label.

```
# prtvtoc /dev/rdsk/c3t1d0s0
* /dev/rdsk/c3t1d0s0 partition map
*
* Dimensions:
*      512 bytes/sector
* 2479267840 sectors
* 2479267773 accessible sectors
*
* Flags:
*    1: unmountable
*   10: read-only
*
*                           First      Sector       Last
* Partition  Tag  Flags    Sector      Count        Sector  Mount Directory
       0      2    00           34      262144      262177
       1      3    01       262178      262144      524321
       6      4    00       524322  2478727100  2479251421
       8     11    00  2479251422       16384  2479267805
```

Recovering a Corrupted Disk Label

Sometimes, a power or system failure causes a disk's label to become unrecognizable. A corrupted disk label doesn't always mean that the slice information or the disk's data must be recreated or restored.

The first step to recovering a corrupted disk label is to label the disk with the correct geometry and disk type information. You can complete this step through the normal disk labeling method, by using either automatic configuration or manual disk type specification.

If the `format` utility recognizes the disk type, the next step is to search for a backup label to label the disk. Labeling the disk with the backup label labels the disk with the correct partitioning information, the disk type, and disk geometry.

▼ How to Recover a Corrupted Disk Label

Steps

1. **Boot the system to single-user mode.**

   If necessary, boot the system from a local CD-ROM or the network in single-user mode to access the disk.

   See Chapter 10, "Booting a System (Tasks)," in *System Administration Guide: Basic Administration* or Chapter 11, "Booting a System (Tasks)," in *System Administration Guide: Basic Administration*for information on booting the system.

2. **Relabel the disk.**

   ```
   # format
   ```

   At this point, the `format` utility attempts to automatically configure any unlabeled SCSI disk. If the `format` utility is able to configure the unlabeled and corrupted disk, it will display:

   ```
   cwtxdy: configured with capacity of abcMB
   ```

   The `format` utility then displays the list of disks on the system.

3. **Type the number of the disk that you need to recover from the list displayed on your screen.**

   ```
   Specify disk (enter its number): 1
   ```

4. **Select one of the following to determine how to label the disk.**

   a. **If the disk was configured successfully, follow steps 5 and 6. Then go to step 12.**

   b. **If the disk was not configured successfully, follow steps 7-11. Then go to step 12.**

5. **Search for the backup label.**

   ```
   format> verify
   Warning: Could not read primary label.
   Warning: Check the current partitioning and 'label' the disk or
   use the 'backup' command.
   Backup label contents:
   Volume name = <        >
   ascii name  = <SUN1.05 cyl 2036 alt 2 hd 14 sec 72>
   pcyl        = 2038
   ncyl        = 2036
   acyl        =    2
   ```

```
nhead       =   14
nsect       =   72
Part      Tag    Flag     Cylinders        Size            Blocks
  0       root    wm       0 -  300      148.15MB    (301/0/0)   303408
  1       swap    wu     301 -  524      110.25MB    (224/0/0)   225792
  2     backup    wm       0 - 2035     1002.09MB    (2036/0/0) 2052288
  3 unassigned    wm       0               0         (0/0/0)          0
  4 unassigned    wm       0               0         (0/0/0)          0
  5 unassigned    wm       0               0         (0/0/0)          0
  6        usr    wm     525 - 2035      743.70MB    (1511/0/0) 1523088
  7 unassigned    wm       0               0         (0/0/0)          0
```

6. **If the `format` utility was able to find a backup label and the backup label contents appear satisfactory, use the `backup` command to label the disk with the backup label.**

   ```
   format> backup
   Disk has a primary label, still continue? y

   Searching for backup labels...found.
   Restoring primary label
   ```

   The disk label has been recovered. Go to step 12.

7. **If the `format` utility was not able to automatically configure the disk, specify the disk type by using the `type` command.**

   ```
   format> type
   ```

   The Available Drives Type menu is displayed.

8. **Select `0` to automatically configure the disk, or select a disk type from the list of possible disk types.**

   ```
   Specify disk type (enter its number)[12]: 12
   ```

9. **If the disk was successfully configured, reply with `no` when the `format` utility asks if you want to label the disk.**

   ```
   Disk not labeled.  Label it now?  no
   ```

10. **Use the `verify` command to search for backup labels.**

    ```
    format> verify
    Warning: Could not read primary label.
    Warning: Check the current partitioning and 'label' the disk
    or use the 'backup' command.
    .
    .
    .
    ```

11. **If the `format` utility was able to find a backup label and the backup label contents appear satisfactory, use the `backup` command to label the disk with the backup label.**

    ```
    format> backup
    Disk has a primary label, still continue? y
    ```

```
Searching for backup labels...found.
Restoring primary label
```

The disk label has been recovered.

12. **Exit the `format` utility.**

    ```
    format> q
    ```

13. **Verify the file systems on the recovered disk by using the `fsck` command.**

    For information on using the `fsck` command, see Chapter 20.

Adding a Third-Party Disk

The Solaris environment supports many third-party disks. However, you might need to supply either a device driver, a `format.dat` entry, or both for the disk to be recognized. Other options for adding disks are as follows:

- If you are adding a SCSI disk, you might to try the `format` utility's automatic configuration feature. For more information, see "Automatically Configuring SCSI Disk Drives" on page 194.
- You might try hot-plugging a PCI, SCSI, or USB disk. For more information, see Chapter 5.

If the third-party disk is designed to work with standard SunOS-compatible device drivers, then creation of an appropriate `format.dat` entry should be enough to allow the disk to be recognized by the `format` utility. In other cases, you need to load a third-party device driver to support the disk.

Note – Sun cannot guarantee that its `format` utility will work properly with all third-party disk drivers. If the disk driver is not compatible with the Solaris `format` utility, the disk drive vendor should supply you with a custom format program.

This section discusses what to do if some of this software support is missing. Typically, you discover that software support is missing when you invoke the `format` utility and find that the disk type is not recognized.

Supply the missing software as described in this section, and then refer to the appropriate configuration procedure for adding system disks or secondary disks in Chapter 12 or Chapter 13.

Creating a `format.dat` Entry

Unrecognized disks cannot be formatted without precise information about the disk's geometry and operating parameters. This information is supplied in the `/etc/format.dat` file.

Note – SCSI-2 drives do not require a `format.dat` entry. The `format` utility automatically configures the SCSI-2 drivers if the drives are powered on during a reconfiguration boot. For step-by-step instructions on configuring a SCSI disk drive automatically, see "How to Automatically Configure a SCSI Drive" on page 195.

If your disk is unrecognized, use a text editor to create an entry in `format.dat` for the disk. You need to gather all the pertinent technical specifications about the disk and its controller before you start. This information should have been provided with the disk. If not, contact the disk manufacturer or your supplier.

▼ How to Create a `format.dat` Entry

Steps

1. **Become superuser or assume an equivalent role.**
2. **Make a copy of the `/etc/format.dat` file.**

   ```
   # cp /etc/format.dat /etc/format.dat.gen
   ```

3. **Modify the `/etc/format.dat` file to include an entry for the third-party disk by using the `format.dat` information that is described in Chapter 14.**

   Use the disk's hardware product documentation to gather the required information.

Automatically Configuring SCSI Disk Drives

The `format` utility automatically configures SCSI disk drives even if that specific type of drive is not listed in the `/etc/format.dat` file. This feature enables you to format, create slices for, and label any disk driver that is compliant with the SCSI-2 specification for disk device mode sense pages.

Other options for adding disks are:

- If you are adding a SCSI disk, you might to try the `format` utility's automatic configuration feature. For more information, see "Automatically Configuring SCSI Disk Drives" on page 194.
- You might try hot-plugging a PCI, SCSI, or USB disk. For more information, see Chapter 5.

The following steps are involved in configuring a SCSI drive by using automatic configuration:

- Shutting down the system
- Attaching the SCSI disk drive to the system
- Turning on the disk drive
- Performing a reconfiguration boot
- Using the `format` utility to automatically configure the SCSI disk drive

After the reconfiguration boot, invoke the `format` utility. The `format` utility will attempt to configure the disk and, if successful, alert the user that the disk was configured. For step-by-step instructions on configuring a SCSI disk drive automatically, see "How to Automatically Configure a SCSI Drive" on page 195.

Here's an example of a partition table for a 1.3-Gbyte SCSI disk drive that was displayed by the `format` utility.

```
Part    Tag    Flag     Cylinders      Size         Blocks
   0     root    wm       0 -    96    64.41MB      (97/0/0)
   1     swap    wu      97 -   289   128.16MB     (193/0/0)
   2   backup    wu       0 - 1964      1.27GB    (1965/0/0)
   6      usr    wm     290 - 1964      1.09GB    (1675/0/0)
```

For more information on using SCSI automatic configuration, see Chapter 14.

▼ How to Automatically Configure a SCSI Drive

Steps

1. **Become superuser or equivalent role.**
2. **Create the `/reconfigure` file that will be read when the system is booted.**

   ```
   # touch /reconfigure
   ```

3. **Shut down the system.**

   ```
   # shutdown -i0 -gn -y
   ```

`-in`	Brings the system down to init level 0, the power-down state.
`-gn`	Notifies logged-in users that they have *n* seconds before the system begins to shut down.
`-y`	Specifies that the command should run without user intervention.

   The `ok` prompt is displayed after the system is shut down.

4. **Turn off the power to the system and all external peripheral devices.**
5. **Make sure that the disk you are adding has a different target number than the other devices on the system.**

   You will often find a small switch located at the back of the disk for this purpose.
6. **Connect the disk to the system and check the physical connections.**

   Refer to the disk's hardware installation guide for installation details.
7. **Turn on the power to all external peripherals.**
8. **Turn on the power to the system.**

   The system boots and displays the login prompt.
9. **Log back in as superuser or assume an equivalent role.**
10. **Invoke the `format` utility and select the disk that you want to configure automatically.**

```
# format
Searching for disks...done
c1t0d0: configured with capacity of 1002.09MB
AVAILABLE DISK SELECTIONS:
0. c0t1d0 <SUN1.05 cyl 2036 alt 2 hd 14 sec 72>
   /iommu@f,e0000000/sbus@f,e0001000/espdma@f,400000/esp@f,800000/sd@1,0
1. c0t3d0 <SUN1.05 cyl 2036 alt 2 hd 14 sec 72>
   /iommu@f,e0000000/sbus@f,e0001000/espdma@f,400000/esp@f,800000/sd@3,0
Specify disk (enter its number): 1
```

11. **Type `yes` to the prompt to label the disk.**

    Typing `y` causes the disk label to be generated and written to the disk by SCSI automatic configuration.

```
Disk not labeled. Label it now? y
```

12. **Verify the disk label.**

```
format> verify
```

13. **Exit the `format` utility.**

```
format> q
```

Repairing a Defective Sector

If a disk on your system has a defective sector, you can repair it by following procedures in this section. You might become aware of defective sectors when you do the following:

- Run surface analysis on a disk

  For more information on the analysis feature of the `format` utility, see "The `analyze` Menu" on page 229.

  The defective area reported while your system is running might not be accurate. Since the system does disk operations many sectors at a time, it is often hard to pinpoint exactly which sector caused a given error. To find the exact sector(s), use "How to Identify a Defective Sector by Using Surface Analysis" on page 197.

- Get multiple error messages from the disk driver concerning a particular portion of the disk while your system is running.

  Messages that are related to disk errors look like the following:

  ```
  WARNING: /io-unit@f,e0200000/sbi@0,0/QLGC,isp@1,10000/sd@3,0 (sd33):
      Error for command 'read' Error Level: Retryable
      Requested Block 126, Error Block: 179
      Sense Key: Media Error
      Vendor 'name':
      ASC = 0x11 (unrecovered read error), ASCQ = 0x0, FRU = 0x0
  ```

  The preceding console message indicates that block 179 might be defective. Relocate the bad block by using the `format` utility's `repair` command or use the `analyze` command with the repair option enabled.

▼ How to Identify a Defective Sector by Using Surface Analysis

Steps

1. **Become superuser or assume an equivalent role.**
2. **Unmount the file system in the slice that contains the defective sector.**

   ```
   # umount /dev/dsk/device-name
   ```

   For more information, see `mount`(1M).
3. **Invoke the `format` utility.**

   ```
   # format
   ```

4. **Select the affected disk.**

   ```
   Specify disk (enter its number):1
   selecting c0t2d0:
   [disk formatted]
   Warning: Current Disk has mounted partitions.
   ```

5. **Select the analyze menu.**

   ```
   format> analyze
   ```

6. **Set up the analysis parameters by typing `setup` at the `analyze>` prompt.**

   Use the parameters shown here:

   ```
   analyze> setup
   Analyze entire disk [yes]? n
   Enter starting block number [0, 0/0/0]: 12330
   Enter ending block number [2052287, 2035/13/71]: 12360
   Loop continuously [no]? y
   Repair defective blocks [yes]? n
   Stop after first error [no]? n
   Use random bit patterns [no]? n
   Enter number of blocks per transfer [126, 0/1/54]: 1
   Verify media after formatting [yes]? y
   Enable extended messages [no]? n
   Restore defect list [yes]? y
   Create defect label [yes]? y
   ```

7. **Use the `read` command to find the defect.**

   ```
   analyze> read
   Ready to analyze (won't harm SunOS). This takes a long time,
   but is interruptible with Control-C. Continue? y
           pass 0
      2035/12/1825/7/24
           pass 1
   Block 12354  (18/4/18), Corrected media error (hard data ecc)
      25/7/24
   ^C
   Total of 1 defective blocks repaired.
   ```

▼ How to Repair a Defective Sector

Steps

1. **Become superuser or assume an equivalent role.**

2. **Invoke the `format` utility.**

   ```
   # format
   ```

3. **Select the disk that contains the defective sector.**

```
Specify disk (enter its number): 1
selecting c0t3d0
[disk formatted]
format>
```

4. **Select the `repair` command.**

   ```
   format> repair
   ```

5. **Type the defective block number.**

   ```
   Enter absolute block number of defect: 12354
      Ready to repair defect, continue? y
      Repairing block 12354 (18/4/18)...ok.
   format>
   ```

   If you are unsure of the format that is used to identify the defective sector, see "How to Identify a Defective Sector by Using Surface Analysis" on page 197 for more information.

Tips and Tricks for Managing Disks

Use the following tips to help you manage disks more efficiently.

Debugging `format` Sessions

Invoke `format -M` to enable extended and diagnostic messages for ATA and SCSI devices.

In this example, the series of numbers under `Inquiry:` represent the hexadecimal value of the `inquiry` data that is displayed to the right of the numbers.

```
# format -M
Searching for disks...done
AVAILABLE DISK SELECTIONS:
  0. c0t1d0 <SUN1.05 cyl 2036 alt 2 hd 14 sec 72>
     /iommu@f,e0000000/sbus@f,e0001000/espdma@f,400000/esp@f,800000/sd@1,0
  1. c0t3d0 <SUN1.05 cyl 2036 alt 2 hd 14 sec 72>
     /iommu@f,e0000000/sbus@f,e0001000/espdma@f,400000/esp@f,800000/sd@3,0

Specify disk (enter its number): 0
selecting c0t3d0
[disk formatted]
format> inquiry
Inquiry:
00 00 02 02 8f 00 00 12 53 45 41 47 41 54 45 20     ........NAME....
```

```
53 54 31 31 32 30 30 4e 20 53 55 4e 31 2e 30 35     ST11200N SUN1.05
38 33 35 38 30 30 30 33 30 32 30 39 00 00 00 00     835800030209....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00     ................
00 43 6f 70 79 72 69 67 68 74 20 28 63 29 20 31     .Copyright (c) 1
39 39 32 20 53 65 61 67 61 74 65 20 41 6c 6c 20     992 NAME    All
72 69 67 68 74 73 20 72 65 73 65 72 76 65 64 20     rights reserved
30 30 30                                            000
Vendor:   name
Product:  ST11200N SUN1.05
Revision: 8358
format>
```

Label Multiple Disks by Using the `prtvtoc` and `fmthard` Commands

Use the `prtvtoc` and `fmthard` commands to label multiple disks with the same disk geometry.

Use the following `for loop` in a script to copy a disk label from one disk and replicate it on multiple disks.

```
# for i in x y z
> do
> prtvtoc /dev/rdsk/cwtxdysz | fmthard -s - /dev/rdsk/cwt${i}d0s2
> done
```

EXAMPLE 11–8 Labeling Multiple Disks

In this example, the disk label from `c2t0d0s0` is copied to four other disks.

```
# for i in 1 2 3 5
> do
> prtvtoc /dev/rdsk/c2t0d0s0 | fmthard -s - /dev/rdsk/c2t${i}d0s2
> done
fmthard:  New volume table of contents now in place.
fmthard:  New volume table of contents now in place.
fmthard:  New volume table of contents now in place.
fmthard:  New volume table of contents now in place.
#
```

CHAPTER 12

SPARC: Adding a Disk (Tasks)

This chapter describes how to add a disk to a SPARC based system.

For information on the procedures associated with adding a disk to a SPARC based system, see "SPARC: Adding a System Disk or a Secondary Disk (Task Map)" on page 201.

For overview information about disk management, see Chapter 10. For step-by-step instructions on adding a disk to an x86 based system, see Chapter 13.

SPARC: Adding a System Disk or a Secondary Disk (Task Map)

The following task map identifies the procedures for adding a disk to a SPARC based system.

Task	Description	For Instructions
1. Connect the disk and boot	*System Disk* Connect the new disk and boot from a local or remote Solaris CD or DVD.	"SPARC: How to Connect a System Disk and Boot" on page 202
	Secondary Disk Connect the new disk and perform a reconfiguration boot so that the system will recognize the disk.	"SPARC: How to Connect a Secondary Disk and Boot" on page 203

Task	Description	For Instructions
2. Create slices and label the disk	Create disk slices and label the disk if the disk manufacturer has not already done so.	"SPARC: How to Create Disk Slices and Label a Disk" on page 204
3. Create file systems	Create UFS file systems on the disk slices with the `newfs` command. You must create the root (/) or `/usr` file system, or both, for a system disk.	"SPARC: How to Create a UFS File System" on page 209
4. Restore file systems	Restore the root (/) or `/usr` file system, or both, on the system disk. If necessary, restore file systems on the secondary disk.	Chapter 25
5. Install boot block	*System Disk Only.* Install the boot block on the root (/) file system, so that the system can boot.	"SPARC: How to Install a Boot Block on a System Disk" on page 209

SPARC: Adding a System Disk or a Secondary Disk

A system disk contains the root (/) or `/usr` file systems, or both. If the disk that contains either of these file systems becomes damaged, you have two ways to recover:

- You can reinstall the entire Solaris environment.
- Or, you can replace the system disk and restore your file systems from a backup medium.

A secondary disk does not contain the root (/) and `/usr` file systems. A secondary disk usually contains space for user files. You can add a secondary disk to a system for more disk space, or you can replace a damaged secondary disk. If you replace a secondary disk on a system, you can restore the old disk's data on the new disk.

▼ SPARC: How to Connect a System Disk and Boot

This procedure assumes that the system is shut down.

Steps

1. **Disconnect the damaged system disk from the system.**

2. **Make sure that the disk you are adding has a different target number than the other devices on the system.**

You will often find a small switch located at the back of the disk for this purpose.

3. **Connect the replacement system disk to the system and check the physical connections.**

   Refer to the disk's hardware installation guide for installation details.

4. **Follow the instructions in the following table, depending on whether you are booting from a local Solaris CD or DVD or a remote Solaris CD or DVD from the network.**

Boot Type	Action
From a Solaris CD or DVD in a local drive	1. Make sure the CD or DVD is in the drive.
	2. Boot from the media to single-user mode:
	`ok` **`boot cdrom -s`**
From the network	Boot from the network to single-user mode:
	`ok` **`boot net -s`**

   After a few minutes, the root prompt (#) is displayed.

See Also After you boot the system, you can create slices and a disk label on the disk. Go to "SPARC: How to Create Disk Slices and Label a Disk" on page 204.

▼ SPARC: How to Connect a Secondary Disk and Boot

Steps

1. **Become superuser or assume an equivalent role.**

2. **If the disk type is unsupported by the Solaris software, add the device driver for the disk by following the instructions included with the hardware.**

   For information on creating a `format.dat` entry for the disk, see "How to Create a `format.dat` Entry" on page 194, if necessary.

3. **Create the `/reconfigure` file that will be read when the system is booted.**

```
# touch /reconfigure
```

   The `/reconfigure` file causes the SunOS software to check for the presence of any newly installed peripheral devices when you power on or boot your system later.

4. **Shut down the system.**

```
# shutdown -i0 -gn -y
```

`-i0`	Changes to run level 0, the power-down state.
`-g`*n*	Notifies logged-in users that they have *n* seconds before the system begins to shut down.
`-y`	Specifies that the command should run without user intervention.

The `ok` prompt is displayed after the Solaris operating system is shut down.

5. **Turn off the power to the system and all external peripheral devices.**
6. **Make sure that the disk you are adding has a different target number than the other devices on the system.**

   You will often find a small switch located at the back of the disk for this purpose.
7. **Connect the disk to the system and check the physical connections.**

   Refer to the disk's hardware installation guide for installation details.
8. **Turn on the power to all external peripherals.**
9. **Turn on the power to the system.**

   The system boots and displays the login prompt.

See Also After you boot the system, you can create slices and a disk label on the disk. Go to "SPARC: How to Create Disk Slices and Label a Disk" on page 204.

▼ SPARC: How to Create Disk Slices and Label a Disk

Steps

1. **Become superuser or assume an equivalent role.**
2. **Invoke the `format` utility.**

   ```
   # format
   ```

   A list of available disks is displayed. For more information, see `format`(1M).
3. **Type the number of the disk that you want to repartition from the list displayed on your screen.**

   ```
   Specify disk (enter its number): disk-number
   ```

   disk-number is the number of the disk that you want to repartition.
4. **Select the `partition` menu.**

   ```
   format> partition
   ```
5. **Display the current partition (slice) table.**

```
partition> print
```

6. **Start the modification process.**

   ```
   partition> modify
   ```

7. **Set the disk to all free hog.**

   ```
   Choose base (enter number) [0]? 1
   ```

   For more information about the free hog slice, see "Using the Free Hog Slice" on page 177.

8. **Create a new partition table by answering `y` when prompted to continue.**

   ```
   Do you wish to continue creating a new partition table based on
   above table[yes]? y
   ```

9. **Identify the free hog partition (slice) and the sizes of the slices when prompted.**

   When adding a system disk, you must set up slices for:

   - root (slice 0) and swap (slice 1)
   - `/usr` (slice 6)

   After you identify the slices, the new partition table is displayed.

   For an example of creating disk slices, see Example 12–1.

10. **Make the displayed partition table the current partition table by answering `y` when asked.**

    ```
    Okay to make this the current partition table[yes]? y
    ```

    If you do not want the current partition table and you want to change it, answer `no` and go to Step 6.

11. **Name the partition table.**

    ```
    Enter table name (remember quotes): "partition-name"
    ```

    partition-name is the name for the new partition table.

12. **Label the disk with the new partition table after you have finished allocating slices on the new disk.**

    ```
    Ready to label disk, continue? yes
    ```

13. **Quit the `partition` menu.**

    ```
    partition> q
    ```

14. **Verify the disk label.**

    ```
    format> verify
    ```

15. **Exit the `format` menu.**

```
format> q
```

Example 12–1 SPARC: Creating Disk Slices and Labeling a System Disk

The following example shows the `format` utility being used to divide a 18-Gbyte disk into three slices: one slice for the root (/) file system, one slice for the swap area, and one slice for the `/usr` file system.

```
# format
AVAILABLE DISK SELECTIONS:
       0. /dev/rdsk/c1t0d0s0 <SUN18G cyl 7506 alt 2 hd 19 sec 248>
          /sbus@2,0/QLGC,isp@2,10000/sd@0,0
       1. /dev/rdsk/c1t1d0s0 <SUN18G cyl 7506 alt 2 hd 19 sec 248>
          /sbus@2,0/QLGC,isp@2,10000/sd@1,0
       2. /dev/rdsk/c1t8d0s0 <SUN18G cyl 7506 alt 2 hd 19 sec 248>
          /sbus@2,0/QLGC,isp@2,10000/sd@8,0
       3. /dev/rdsk/c1t9d0s0 <SUN18G cyl 7506 alt 2 hd 19 sec 248>
          /sbus@2,0/QLGC,isp@2,10000/sd@9,0
Specify disk (enter its number): 0
selecting c1t0d0
[disk formatted]
format> partition
partition> print
partition> modify
Select partitioning base:
    0. Current partition table (original)
    1. All Free Hog
Part      Tag    Flag     Cylinders        Size            Blocks
  0       root    wm       0               0         (0/0/0)           0
  1       swap    wu       0               0         (0/0/0)           0
  2     backup    wu       0 - 7505       16.86GB    (7506/0/0) 35368272
  3 unassigned    wm       0               0         (0/0/0)           0
  4 unassigned    wm       0               0         (0/0/0)           0
  5 unassigned    wm       0               0         (0/0/0)           0
  6        usr    wm       0               0         (0/0/0)           0
  7 unassigned    wm       0               0         (0/0/0)           0

Choose base (enter number) [0]? 1
table based on above table[yes]? yes
Free Hog partition[6]? 6
Enter size of partition '0' [0b, 0c, 0.00mb, 0.00gb]: 4gb
Enter size of partition '1' [0b, 0c, 0.00mb, 0.00gb]: 4gb
Enter size of partition '3' [0b, 0c, 0.00mb, 0.00gb]:
Enter size of partition '4' [0b, 0c, 0.00mb, 0.00gb]:
Enter size of partition '5' [0b, 0c, 0.00mb, 0.00gb]:
Enter size of partition '7' [0b, 0c, 0.00mb, 0.00gb]:
Part      Tag    Flag     Cylinders        Size            Blocks
  0       root    wm       0 - 1780        4.00GB    (1781/0/0)  8392072
  1       swap    wu    1781 - 3561        4.00GB    (1781/0/0)  8392072
  2     backup    wu       0 - 7505       16.86GB    (7506/0/0) 35368272
  3 unassigned    wm       0               0         (0/0/0)           0
  4 unassigned    wm       0               0         (0/0/0)           0
  5 unassigned    wm       0               0         (0/0/0)           0
  6        usr    wm    3562 - 7505        8.86GB    (3944/0/0) 18584128
  7 unassigned    wm       0               0         (0/0/0)           0

Okay to make this the current partition table[yes]? yes
Enter table name (remember quotes): "disk0"
Ready to label disk, continue? yes
partition> quit
format> verify
format> quit
```

Example 12–2 SPARC: Creating Disk Slices and Labeling a Secondary Disk

The following example shows the `format` utility being used to divide a 18-Gbyte disk into one slice for the `/export/home` file system.

```
# format /dev/rdsk/c1*
AVAILABLE DISK SELECTIONS:
       0. /dev/rdsk/c1t0d0s0 <SUN18G cyl 7506 alt 2 hd 19 sec 248>
          /sbus@2,0/QLGC,isp@2,10000/sd@0,0
       1. /dev/rdsk/c1t1d0s0 <SUN18G cyl 7506 alt 2 hd 19 sec 248>
          /sbus@2,0/QLGC,isp@2,10000/sd@1,0
       2. /dev/rdsk/c1t8d0s0 <SUN18G cyl 7506 alt 2 hd 19 sec 248>
          /sbus@2,0/QLGC,isp@2,10000/sd@8,0
       3. /dev/rdsk/c1t9d0s0 <SUN18G cyl 7506 alt 2 hd 19 sec 248>
          /sbus@2,0/QLGC,isp@2,10000/sd@9,0
Specify disk (enter its number): 1
selecting c1t1d0
[disk formatted]
format> partition
partition> print
partition> modify
Select partitioning base:
    0. Current partition table (original)
    1. All Free Hog
Choose base (enter number) [0]? 1
Part      Tag    Flag     Cylinders        Size            Blocks
  0       root    wm       0               0         (0/0/0)           0
  1       swap    wu       0               0         (0/0/0)           0
  2     backup    wu       0 - 7505       16.86GB    (7506/0/0) 35368272
  3 unassigned    wm       0               0         (0/0/0)           0
  4 unassigned    wm       0               0         (0/0/0)           0
  5 unassigned    wm       0               0         (0/0/0)           0
  6        usr    wm       0               0         (0/0/0)           0
  7 unassigned    wm       0               0         (0/0/0)           0

Do you wish to continue creating a new partition
table based on above table[yes]? y
Free Hog partition[6]? 7
Enter size of partition '0' [0b, 0c, 0.00mb, 0.00gb]:
Enter size of partition '1' [0b, 0c, 0.00mb, 0.00gb]:
Enter size of partition '3' [0b, 0c, 0.00mb, 0.00gb]:
Enter size of partition '4' [0b, 0c, 0.00mb, 0.00gb]:
Enter size of partition '5' [0b, 0c, 0.00mb, 0.00gb]:
```

```
Enter size of partition '6' [0b, 0c, 0.00mb, 0.00gb]:
Part      Tag    Flag     Cylinders        Size            Blocks
  0       root    wm       0                0         (0/0/0)           0
  1       swap    wu       0                0         (0/0/0)           0
  2     backup    wu       0 - 7505       16.86GB    (7506/0/0) 35368272
  3 unassigned    wm       0                0         (0/0/0)           0
  4 unassigned    wm       0                0         (0/0/0)           0
  5 unassigned    wm       0                0         (0/0/0)           0
  6        usr    wm       0                0         (0/0/0)           0
  7 unassigned    wm       0 - 7505       16.86GB    (7506/0/0) 35368272
Okay to make this the current partition table[yes]? yes
Enter table name (remember quotes): "home"
Ready to label disk, continue? y
partition> q
format> verify
format> q
#
```

The following example shows how to use the format utility to divide a 1.15 terabyte disk with an EFI label into 3 slices.

```
# format
.
.
.
partition> modify
Select partitioning base:
        0. Current partition table (original)
        1. All Free Hog
Choose base (enter number) [0]? 1
Part      Tag    Flag     First Sector          Size          Last Sector
  0       root    wm                0             0                 0
  1        usr    wm                0             0                 0
  2 unassigned    wm                0             0                 0
  3 unassigned    wm                0             0                 0
  4 unassigned    wm                0             0                 0
  5 unassigned    wm                0             0                 0
  6        usr    wm                0             0                 0
  8   reserved    wm       2576924638         8.00MB           2576941021
Do you wish to continue creating a new partition
table based on above table[yes]? y
Free Hog partition[6]? 4
Enter size of partition 0 [0b, 34e, 0mb, 0gb, 0tb]:
Enter size of partition 1 [0b, 34e, 0mb, 0gb, 0tb]:
Enter size of partition 2 [0b, 34e, 0mb, 0gb, 0tb]: 400gb
Enter size of partition 3 [0b, 838860834e, 0mb, 0gb, 0tb]: 400gb
Enter size of partition 5 [0b, 1677721634e, 0mb, 0gb, 0tb]:
Enter size of partition 6 [0b, 1677721634e, 0mb, 0gb, 0tb]:
Part      Tag    Flag     First Sector          Size          Last Sector
  0 unassigned    wm                0             0                 0
  1 unassigned    wm                0             0                 0
  2        usr    wm               34       400.00GB           838860833
  3        usr    wm        838860834       400.00GB           1677721633
  4        usr    wm       1677721634       428.77GB           2576924637
  5 unassigned    wm                 0            0                0
  6 unassigned    wm                 0            0                0
  8   reserved    wm        2576924638        8.00MB          2576941021
Ready to label disk, continue? yes

partition> q
```

See Also After you create disk slices and label the disk, you can create file systems on the disk. Go to "SPARC: How to Create a UFS File System" on page 209.

▼ SPARC: How to Create a UFS File System

Steps

1. **Become superuser or assume an equivalent role.**
2. **Create a file system for each slice.**

   ```
   # newfs /dev/rdsk/cwtxdysz
   ```

   `/dev/rdsk/`*cwtxdysx* is the raw device for the file system to be created.

   For more information about the `newfs` command, see Chapter 16 or `newfs`(1M).
3. **Verify the new file system by mounting it.**

   ```
   # mount /dev/dsk/cwtxdysz /mnt
   # ls lost+found
   ```

See Also

- **System Disk** – You need to restore the root (/) and `/usr` file systems on the disk. Go to Chapter 25.
- After the root (/) and `/usr` file systems are restored, install the boot block. Go to "SPARC: How to Install a Boot Block on a System Disk" on page 209.
- **Secondary Disk** – You might need to restore file systems on the new disk. Go to Chapter 25. If you are not restoring file systems on the new disk, you are finished adding a secondary disk.
- For information on making the file systems available to users, see Chapter 17.

▼ SPARC: How to Install a Boot Block on a System Disk

Steps

1. **Become superuser or assume an equivalent role.**
2. **Install a boot block on the system disk.**

```
# installboot /usr/platform/`uname -i`/lib/fs/ufs/bootblk /dev/rdsk/cwtxdys0
```

`/usr/platform/‘uname -i‘/lib/fs /ufs/bootblk`
: Is the boot block code.

`/dev/rdsk/`*cwtxdy*`s0`
: Is the raw device of the root (/) file system.

For more information, see `installboot`(1M).

3. **Verify that the boot blocks are installed by rebooting the system to run level 3.**

```
# init 6
```

Example 12–3 SPARC: Installing a Boot Block on a System Disk

The following example shows how to install the boot block on an Ultra10 system.

```
# installboot /usr/platform/sun4u/lib/fs/ufs/bootblk /dev/rdsk/c0t0d0s0
```

CHAPTER 13

x86: Adding a Disk (Tasks)

This chapter describes how to add a disk to an x86 based system.

For information on the procedures associated with adding a disk to an x86 based system, see "x86: Adding a System Disk or a Secondary Disk (Task Map)" on page 211.

For overview information about disk management, see Chapter 10. For step-by-step instructions on adding a disk to a SPARC based system, see Chapter 12.

x86: Adding a System Disk or a Secondary Disk (Task Map)

Task	Description	For Instructions
1. Connect the disk and boot	*System Disk* Connect the new disk and boot from a local or remote Solaris CD or DVD.	"x86: How to Connect a System Disk and Boot" on page 212
	Secondary Disk Connect the new disk and perform a reconfiguration boot, so that the system will recognize the disk.	"x86: How to Connect a Secondary Disk and Boot" on page 213

Task	Description	For Instructions
2. Create slices and label the disk	Create disk slices and label the disk if the disk manufacturer has not already done so.	"x86: How to Create a Solaris `fdisk` Partition" on page 215 and "x86: How to Create Disk Slices and Label a Disk" on page 220
3. Create File Systems	Create UFS file systems on the disk slices with the `newfs` command. You must create the root (/) or `/usr` file system (or both) for a system disk.	"x86: How to Create File Systems" on page 222
4. Restore File Systems	Restore the root (/) or `/usr` file system (or both) on the system disk. If necessary, restore file systems on the secondary disk.	Chapter 25
5. Install Boot Block	*System Disk Only.* Install the boot block on the root (/) file system so that the system can boot.	"x86: How to Install a Boot Block on a System Disk" on page 222

x86: Adding a System or Secondary Disk

A system disk contains the root (/) or `/usr` file systems, or both. If the disk that contains either of these file systems becomes damaged, you have two ways to recover:

- You can reinstall the entire Solaris environment.
- Or, you can replace the system disk and restore your file systems from a backup medium.

A secondary disk doesn't contain the root (/) and `/usr` file systems. A secondary disk usually contains space for user files. You can add a secondary disk to a system for more disk space, or you can replace a damaged secondary disk. If you replace a secondary disk on a system, you can restore the old disk's data on the new disk.

▼ x86: How to Connect a System Disk and Boot

This procedure assumes that the system is down.

Steps 1. **Disconnect the damaged system disk from the system.**

2. **Make sure that the disk you are adding has a different target number than the other devices on the system.**

   You will often find a small switch located at the back of the disk for this purpose.

3. **Connect the replacement system disk to the system and check the physical connections.**

   Refer to the disk's hardware installation guide for installation details.

4. **Follow steps a-e if you are booting from a local Solaris CD or DVD or a remote Solaris CD or DVD from the network.**

   If you are booting from the network, skip step a.

   a. **If you are booting from a local Solaris CD or DVD, insert the Solaris installation CD or DVD into the drive.**

   b. **Insert the Solaris boot diskette into the primary diskette drive (DOS drive A).**

   c. **Press any key to reboot the system if the system displays the `Type any key to continue` prompt. Or, use the reset button to restart the system if the system is shut down.**

      The Boot Solaris screen is displayed after a few minutes.

   d. **Select the CD-ROM drive or net(work) as the boot device from the Boot Solaris screen.**

      The Current Boot Parameters screen is displayed.

   e. **Boot the system in single-user mode.**

      ```
      Select the type of installation: b -s
      ```

      After a few minutes, the root prompt (#) is displayed.

See Also After you boot the system, you can create an `fdisk` partition. Go to "x86: How to Create a Solaris `fdisk` Partition" on page 215.

▼ x86: How to Connect a Secondary Disk and Boot

Steps

1. **Become superuser or assume an equivalent role.**

2. **If the disk is unsupported by the Solaris software, add the device driver for the disk by following the instructions included with the hardware.**

3. **Create the `/reconfigure` file that will be read when the system is booted.**

   ```
   # touch /reconfigure
   ```

   The `/reconfigure` file causes the SunOS software to check for the presence of any newly installed peripheral devices when you power on or boot your system later.

4. **Shut down the system.**

   ```
   # shutdown -i0 -gn -y
   ```

`-i0`	Brings the system down to run level 0, the power-down state.
`-gn`	Notifies logged-in users that they have *n* seconds before the system begins to shut down.
`-y`	Specifies that the command should run without user intervention.

   The `Type any key to continue` prompt is displayed.

5. **Turn off the power to the system and all external peripheral devices.**

6. **Make sure that the disk you are adding has a different target number than the other devices on the system.**

   You will often find a small switch located at the back of the disk for this purpose.

7. **Connect the disk to the system and check the physical connections.**

   Refer to the disk's hardware installation guide for installation details.

8. **Turn on the power to all external peripherals.**

9. **Turn on the power to the system.**

   The system boots and displays the login prompt.

See Also After you boot the system, you can create an `fdisk` partition. Go to "x86: How to Create a Solaris `fdisk` Partition" on page 215.

x86: Guidelines for Creating an `fdisk` Partition

Follow these guidelines when you set up the `fdisk` partition.

- The disk can be divided into a maximum of four `fdisk` partitions. One of partitions must be a Solaris partition.
- The Solaris partition must be made the active partition on the disk. The active partition is partition whose operating system will be booted by default at system startup.
- Solaris `fdisk` partitions must begin on cylinder boundaries.
- Solaris `fdisk` partitions must begin at cylinder 1, not cylinder 0, on the first disk because additional boot information, including the master boot record, is written in sector 0.

- The Solaris `fdisk` partition can be the entire disk or you might want to make it smaller to allow room for a DOS partition. You can also make a new `fdisk` partition on a disk without disturbing existing partitions (if there is enough room to create a new one).

---

x86 only – Solaris slices are sometimes called partitions. This book uses the term slice, but some Solaris documentation and programs might refer to a *slice* as a *partition*.

To avoid confusion, Solaris documentation tries to distinguish between `fdisk` partitions (which are supported only on Solaris (x86 Platform Edition)) and the divisions within the Solaris `fdisk` partition, which might be called slices or partitions.

▼ x86: How to Create a Solaris `fdisk` Partition

Steps

1. **Read "x86: Guidelines for Creating an `fdisk` Partition" on page 214.**
2. **Become superuser or assume an equivalent role.**
3. **Invoke the `format` utility.**

   ```
   # format
   ```

   For more information, see `format`(1M).
4. **Type the number of the disk on which to create a Solaris `fdisk` partition from the list displayed on your screen.**

   ```
   Specify disk (enter its number): disk-number
   ```

   disk-number is the number of the disk on which you want to create a Solaris `fdisk` partition.
5. **Select the `fdisk` menu.**

   ```
   format> fdisk
   ```

   The `fdisk` menu that is displayed depends upon whether the disk has existing `fdisk` partitions. Determine the next step using the following table.

Task	Go To	For More Information
Create a Solaris `fdisk` partition to span the entire disk.	Step 6	Example 13–1
Create a Solaris `fdisk` partition and preserve one or more existing non-Solaris `fdisk` partition.	Step 7	Example 13–2

Task	Go To	For More Information
Create a Solaris `fdisk` partition and one or more additional non-Solaris `fdisk` partition.	Step 7	Example 13–3

6. **Create and activate a Solaris `fdisk` partition that spans the entire disk by specifying `y` at the prompt. Then, go to step 14.**

```
The recommended default partitioning for your disk is:

  a 100% “SOLARIS System” partition.

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

7. **Specify `n` at the prompt if you do not want the Solaris `fdisk` partition to span the entire disk.**

```
To select this, please type "y".  To partition your disk
differently, type "n" and the "fdisk" program will let you
select other partitions. n
Total disk size is 2694 cylinders
             Cylinder size is 765 (512 byte) blocks
                                              Cylinders
    Partition   Status    Type      Start   End   Length    %
    =========   ======    ========  =====   ===   ======   ===
THERE ARE NO PARTITIONS CURRENTLY DEFINED SELECT ONE OF THE
FOLLOWING:

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

8. **Select option 1, `Create a partition`, to create an `fdisk` partition.**

```
             Total disk size is 2694 cylinders
             Cylinder size is 765 (512 byte) blocks
                                              Cylinders
    Partition   Status    Type      Start   End   Length    %
    =========   ======    ========  =====   ===   ======   ===
THERE ARE NO PARTITIONS CURRENTLY DEFINED SELECT ONE OF THE
FOLLOWING:

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

9. **Create a Solaris `fdisk` partition by selecting `1(=Solaris)`.**

   ```
   Indicate the type of partition you want to create
     (1=SOLARIS, 2=UNIX, 3=PCIXOS, 4=Other, 8=DOSBIG)
     (5=DOS12, 6=DOS16, 7=DOSEXT, 0=Exit) ? 1
   ```

10. **Identify the percentage of the disk to be reserved for the Solaris `fdisk` partition. Keep in mind the size of any existing `fdisk` partitions when you calculate this percentage.**

    ```
    Indicate the percentage of the disk you want this partition
    to use (or enter "c" to specify in cylinders). nn
    ```

11. **Activate the Solaris `fdisk` partition by typing `y` at the prompt.**

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

    The `Enter Selection:` prompt is displayed after the `fdisk` partition is activated.

12. **Select option 1, `Create a partition`, to create another `fdisk` partition.**

    See steps 9-11 for instructions on creating an `fdisk` partition.

13. **Update the disk configuration and exit the `fdisk` menu from the selection menu.**

    ```
    Selection: 4
    ```

14. **Relabel the disk by using the `label` command.**

    ```
    WARNING: Solaris fdisk partition changed - Please relabel the disk
    format> label
    Ready to label disk, continue? yes
    format>
    ```

15. **Quit the `format` menu.**

    ```
    format> quit
    ```

Example 13–1 x86: Creating a Solaris `fdisk` Partition That Spans the Entire Drive

The following example uses the `format`'s utility's `fdisk` option to create a Solaris `fdisk` partition that spans the entire drive.

```
# format
Searching for disks...done
AVAILABLE DISK SELECTIONS:
       0. c0d0 <DEFAULT cyl 2466 alt 2 hd 16 sec 63>
          /pci@0,0/pci-ide@7,1/ide@0/cmdk@0,0
       1. c0d1 <DEFAULT cyl 522 alt 2 hd 32 sec 63>
          /pci@0,0/pci-ide@7,1/ide@0/cmdk@1,0
       2. c1d0 <DEFAULT cyl 13102 alt 2 hd 16 sec 63>
          /pci@0,0/pci-ide@7,1/ide@1/cmdk@0,0
Specify disk (enter its number): 0
```

```
selecting c0d0
Controller working list found
[disk formatted]
format> fdisk
The recommended default partitioning for your disk is:

  a 100% "SOLARIS System" partition.

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

WARNING: Solaris fdisk partition changed - Please relabel the disk
format> label
Ready to label disk, continue? yes
format> quit
```

Example 13–2 x86: Creating a Solaris `fdisk` Partition While Preserving an Existing `fdisk` Partition

The following example shows how to create a Solaris `fdisk` partition on a disk that has an existing `DOS-BIG` `fdisk` partition.

```
format> fdisk
             Total disk size is 2694 cylinders
             Cylinder size is 765 (512 byte) blocks
                                            Cylinders
     Partition   Status    Type     Start  End   Length    %
     =========   ======   ========  =====   ===   ====== ===
        1                 DOS-BIG       1   538     538   20
SELECT ONE OF THE FOLLOWING:
     1.   Create a partition
     2.   Change Active (Boot from) partition
     3.   Delete a partition
     4.   Exit (Update disk configuration and exit)
     5.   Cancel (Exit without updating disk configuration)
Enter Selection: 1
Indicate the type of partition you want to create
  (1=SOLARIS, 2=UNIX, 3=PCIXOS, 4=Other, 8=DOSBIG)
  (5=DOS12, 6=DOS16, 7=DOSEXT, 0=Exit) ?1
Indicate the percentage of the disk you want this partition
to use (or enter "c" to specify in cylinders). 80
Do you want this to become the Active partition? If so, it will be
activated each time you reset your computer or when you turn it on
again. Please type "y" or "n". y
Partition 2 is now the Active partition Total disk size is 2694
cylinders
             Cylinder size is 765 (512 byte) blocks
                                            Cylinders
    Partition   Status    Type      Start   End   Length    %
    =========   ======    ========  =====   ===   ======   ===
        1                 DOS-BIG       1   538     538     20
        2       Active    SOLARIS     539  2693    2155     80
SELECT ONE OF THE FOLLOWING:
     1.   Create a partition
     2.   Change Active (Boot from) partition
     3.   Delete a partition
     4.   Exit (Update disk configuration and exit)
     5.   Cancel (Exit without updating disk configuration)
Enter Selection: Selection: 4
WARNING: Solaris fdisk partition changed - Please relabel the disk
format> label
Ready to label disk, continue? yes
format> q
```

Example 13–3 x86: Creating a Solaris `fdisk` Partition and an Additional `fdisk` Partition

This following example shows how to create a Solaris `fdisk` partition and a `DOSBIG` `fdisk` partition.

```
format> fdisk
The recommended default partitioning for your disk is:
   a 100% "SOLARIS System" partition.
To select this, please type "y".  To partition your disk
differently, type "n" and the "fdisk" program will let you
select other partitions. n
             Total disk size is 2694 cylinders
             Cylinder size is 765 (512 byte) blocks
                                            Cylinders
    Partition   Status    Type      Start   End   Length    %
    =========   ======    ========  =====   ===   ======   ===
THERE ARE NO PARTITIONS CURRENTLY DEFINED SELECT ONE OF THE FOLLOWING:
     1.   Create a partition
     2.   Change Active (Boot from) partition
     3.   Delete a partition
     4.   Exit (Update disk configuration and exit)
     5.   Cancel (Exit without updating disk configuration)
Enter Selection: 1
Indicate the type of partition you want to create
  (1=SOLARIS, 2=UNIX, 3=PCIXOS, 4=Other, 8=DOSBIG)
  (5=DOS12, 6=DOS16, 7=DOSEXT, 0=Exit) ?8
Indicate the percentage of the disk you want this partition
to use (or enter "c" to specify in cylinders). 20
Do you want this to become the Active partition? If so, it will be
activated each time you reset your computer or when you turn it on
again. Please type "y" or "n". n
             Total disk size is 2694 cylinders
             Cylinder size is 765 (512 byte) blocks
                                           Cylinders
    Partition   Status    Type      Start   End   Length    %
    =========   ======    ========  =====   ===   ======   ===
       1                  DOS-BIG       1   538     538     20
SELECT ONE OF THE FOLLOWING:
     1.   Create a partition
     2.   Change Active (Boot from) partition
     3.   Delete a partition
     4.   Exit (Update disk configuration and exit)
     5.   Cancel (Exit without updating disk configuration)Enter
Selection: 1
Indicate the type of partition you want to create
  (1=SOLARIS, 2=UNIX, 3=PCIXOS, 4=Other, 8=DOSBIG)
  (5=DOS12, 6=DOS16, 7=DOSEXT, 0=Exit) ?1
Indicate the percentage of the disk you want this partition
to use (or enter "c" to specify in cylinders). 80
Do you want this to become the Active partition? If so, it will be
activated each time you reset your computer or when you turn it on
again. Please type "y" or "n". y
Partition 2 is now the Active partition Total disk size is 2694
cylinders
            Cylinder size is 765 (512 byte) blocks
                                         Cylinders
    Partition   Status    Type      Start   End   Length    %
    =========   ======    ========  =====   ===   ======   ===
        1                 DOS-BIG       1   538     538     20
        2       Active    SOLARIS     539  2693    2155     80

SELECT ONE OF THE FOLLOWING:
     1.   Create a partition
     2.   Change Active (Boot from) partition
     3.   Delete a partition
     4.   Exit (Update disk configuration and exit)
     5.   Cancel (Exit without updating disk configuration)
Enter Selection: 4
format> q
```

See Also After you create a Solaris `fdisk` partition on the disk, you can create slices on the disk. Go to "x86: How to Create Disk Slices and Label a Disk" on page 220.

▼ x86: How to Create Disk Slices and Label a Disk

Steps

1. **Become superuser or assume an equivalent role.**
2. **Start the `format` utility.**

   ```
   # format
   ```

3. **Type the number of the disk that you want to repartition from the list displayed on your screen.**

   ```
   Specify disk (enter its number): disk-number
   ```

   disk-number is the number of the disk that you want to repartition.

4. **Select the `partition` menu.**

   ```
   format> partition
   ```

5. **Display the current partition (slice) table.**

   ```
   partition> print
   ```

6. **Start the modification process.**

   ```
   partition> modify
   ```

7. **Set the disk to all free hog.**

   ```
   Choose base (enter number) [0]? 1
   ```

   For more information about the free hog slice, see "Using the Free Hog Slice" on page 177.

8. **Create a new partition table by answering `yes` when prompted to continue.**

   ```
   Do you wish to continue creating a new partition
   table based on above table[yes]? yes
   ```

9. **Identify the free hog partition (slice) and the sizes of the slices when prompted.**

   When adding a system disk, you must set up slices for:

   - root (slice 0) and swap (slice 1) and/or
   - `/usr` (slice 6)

   After you identify the slices, the new partition table is displayed.

10. **Make the displayed partition table the current partition table by answering `yes` when asked.**

    ```
    Okay to make this the current partition table[yes]? yes
    ```

    If you don't want the current partition table and you want to change it, answer `no` and go to Step 6.

11. **Name the partition table.**

    ```
    Enter table name (remember quotes): "partition-name"
    ```

    partition-name is the name for the new partition table.

12. **Label the disk with the new partition table after you have finished allocating slices on the new disk.**

    ```
    Ready to label disk, continue? yes
    ```

13. **Quit the `partition` menu.**

    ```
    partition> quit
    ```

14. **Verify the new disk label.**

    ```
    format> verify
    ```

15. **Exit the `format` menu.**

    ```
    format> quit
    ```

See Also After you create disk slices and label the disk, you can create file systems on the disk. Go to "x86: How to Create File Systems" on page 222.

▼ x86: How to Create File Systems

Steps

1. **Become superuser or assume an equivalent role.**

2. **Create a file system for each slice.**

   ```
   # newfs /dev/rdsk/cwtxdysz
   ```

   `/dev/rdsk/`*cwtxdysz* is the raw device for the file system to be created.

   For more information about the `newfs` command, see Chapter 16 or `newfs`(1M).

3. **Verify the new file system by mounting.**

   ```
   # mount /dev/dsk/cwtxdysz /mnt
   # ls /mnt
   lost+found
   ```

See Also

- **System Disk** – You need to restore the root (`/`) and `/usr` file systems on the disk. Go to Chapter 25.
- After the root (`/`) and `/usr` file systems are restored, install the boot block. Go to "x86: How to Install a Boot Block on a System Disk" on page 222.
- **Secondary Disk** – You might need to restore file systems on the new disk. Go to Chapter 25. If you are not restoring file systems on the new disk, you are finished adding a secondary disk.
- For information on making the file systems available to users, see Chapter 17.

▼ x86: How to Install a Boot Block on a System Disk

Steps

1. **Become superuser or assume an equivalent role.**

2. **Install the boot block on the system disk.**

   ```
   # installboot /usr/platform/‘uname -i‘/lib/fs/ufs/pboot /usr/platform/
   ‘uname -i‘ /lib/fs/ufs/bootblk /dev/rdsk/cwtxdys2
   ```

   `/usr/platform/‘uname -i‘/lib/fs/ufs/pboot`
   Is the partition boot file.

   `/usr/platform/‘uname -i‘/lib/fs/ufs/bootblk`
   Is the boot block code.

   `/dev/rdsk/`*cwtxdy*`s2`
   Is the raw device name that represents the whole disk.

3. **Verify that the boot blocks are installed by rebooting the system to run level 3.**

   ```
   # init 6
   ```

Example 13–4 x86: Installing a Boot Block on a System Disk

```
# installboot /usr/platform/i86pc/lib/fs/ufs/pboot
 /usr/platform/i86pc/lib/fs/ufs/bootblk /dev/rdsk/c0t6d0s2
```

CHAPTER 14

The `format` Utility (Reference)

This chapter describes the `format` utility's menu and commands.

This is a list of the reference information in this chapter.

For a overview of when to use the `format` utility, see Chapter 10.

Recommendations and Requirements for Using The `format` Utility

You must be superuser or a member of an equivalent role to use the `format` utility. If you are not superuser or have assumed an equivalent role, you will see the following error message when trying to use the `format` utility:

```
$ format
Searching for disks...done
No permission (or no disks found)!
```

Keep the following guidelines in mind when using the `format` utility and you want to preserve the existing data:

- Back up all files on the disk drive.
- Save all your defect lists in files by using the `format` utility's `dump` command. The file name should include the drive type, model number, and serial number.
- Save the paper copies of the manufacturer's defect list that was shipped with your drive.

Format Menu and Command Descriptions

The `format` main menu looks like the following:

```
FORMAT MENU:
        disk       - select a disk
        type       - select (define) a disk type
        partition  - select (define) a partition table
        current    - describe the current disk
        format     - format and analyze the disk
        repair     - repair a defective sector
        label      - write label to the disk
        analyze    - surface analysis
        defect     - defect list management
        backup     - search for backup labels
        verify     - read and display labels
        save       - save new disk/partition definitions
        inquiry    - show vendor, product and revision
        volname    - set 8-character volume name
        quit
format>
```

The following table describes the `format` main menu items.

TABLE 14–1 The `format` Main Menu Item Descriptions

Item	Command or Menu?	Description
`disk`	Command	Lists all of the system's drives. Also lets you choose the disk you want to use in subsequent operations. This disk is referred to as the current disk.
`type`	Command	Identifies the manufacturer and model of the current disk. Also displays a list of known drive types. Choose the `Auto configure` option for all SCSI-2 disk drives.
`partition`	Menu	Creates and modifies slices. For more information, see "The `partition` Menu" on page 228.
`current`	Command	Displays the following information about the current disk: ■ Device name and device type ■ Number of cylinders, alternate cylinders, heads and sectors ■ Physical device name

TABLE 14–1 The `format` Main Menu Item Descriptions *(Continued)*

Item	Command or Menu?	Description
`format`	Command	Formats the current disk by using one of these sources of information in this order: 1. Information that is found in the `format.dat` file 2. Information from the automatic configuration process 3. Information that you enter at the prompt if there is no `format.dat` entry This command does not apply to IDE disks. IDE disks are pre–formatted by the manufacturer.
`fdisk`	Menu	x86 platform only: Runs the `fdisk` program to create a Solaris `fdisk` partition.
`repair`	Command	Repairs a specific block on the current disk.
`label`	Command	Writes a new label to the current disk.
`analyze`	Menu	Runs read, write, compare tests. For more information, see "The `analyze` Menu" on page 229.
`defect`	Menu	Retrieves and prints defect lists. For more information, see "The `defect` Menu" on page 231. This feature does not apply to IDE disks. IDE disks perform automatic defect management.
`backup`	Command	**VTOC** – Searches for backup labels. **EFI** – Not supported.
`verify`	Command	Prints the following information about the current disk: ■ Device name and device type ■ Number of cylinders, alternate cylinders, heads and sectors ■ Partition table
`save`	Command	**VTOC** –Saves new disk and partition information. **EFI** – Not applicable.
`inquiry`	Command	Prints the vendor, product name, and revision level of the current drive (SCSI disks only).
`volname`	Command	Labels the disk with a new eight-character volume name.
`quit`	Command	Exits the `format` menu.

The partition Menu

The partition menu looks similar to the following:

```
format> partition
PARTITION MENU:
        0      - change '0' partition
        1      - change '1' partition
        2      - change '2' partition
        3      - change '3' partition
        4      - change '4' partition
        5      - change '5' partition
        6      - change '6' partition
        7      - change '7' partition
        select - select a predefined table
        modify - modify a predefined partition table
        name   - name the current table
        print  - display the current table
        label  - write partition map and label to the disk
        quit
partition>
```

The following table describes the partition menu items.

TABLE 14–2 Descriptions for partition Menu Items

Sub-Command	Description
`change 'n' partition`	Lets you specify the following information for the new slice: ■ Identification tag ■ Permission flags ■ Starting cylinder ■ Size
`select`	Lets you choose a predefined slice table.
`modify`	Lets you change all the slices in the slice table. This command is preferred over the individual `change 'x' partition` commands.
`name`	Lets you specify a name for the current slice table.
`print`	Displays the current slice table.
`label`	Writes the slice map and the label to the current disk.
`quit`	Exits the partition menu.

x86: The fdisk Menu

The fdisk menu appears on x86 based systems only and looks similar to the following.

```
format> fdisk
             Total disk size is 1855 cylinders
             Cylinder size is 553 (512 byte) blocks
                                                Cylinders
      Partition   Status    Type      Start   End   Length    %
      =========   ======    ========  =====   ===   ======   ===
          1                 DOS-BIG       0   370     371     20
          2       Active    SOLARIS     370  1851    1482     80

SELECT ONE OF THE FOLLOWING:
     1.   Create a partition
     2.   Change Active (Boot from) partition
     3.   Delete a partition
     4.   Exit (Update disk configuration and exit)
     5.   Cancel (Exit without updating disk configuration)
Enter Selection:
```

The following table describes the `fdisk` menu items.

TABLE 14–3 x86: Descriptions for `fdisk` Menu Items

Menu Item	Description
`Create a partition`	Creates an `fdisk` partition. You must create a separate partition for each operating system such as Solaris or DOS. There is a maximum of 4 partitions per disk. You are prompted for the size of the `fdisk` partition as a percentage of the disk.
`Change Active partition`	Lets you specify the partition to be used for booting. This menu item identifies where the first stage boot program looks for the second stage boot program.
`Delete a partition`	Deletes a previously created partition. This command destroys all the data in the partition.
`Exit`	Writes a new version of the partition table and exits the `fdisk` menu.
`Cancel`	Exits the `fdisk` menu without modifying the partition table.

The `analyze` Menu

The `analyze` menu looks similar to the following.

```
format> analyze

ANALYZE MENU:
    read     - read only test   (doesn't harm SunOS)
    refresh  - read then write  (doesn't harm data)
    test     - pattern testing  (doesn't harm data)
    write    - write then read      (corrupts data)
    compare  - write, read, compare (corrupts data)
    purge    - write, read, write   (corrupts data)
    verify   - write entire disk, then verify (corrupts data)
    print    - display data buffer
    setup    - set analysis parameters
    config   - show analysis parameters
    quit
analyze>
```

The following table describes the `analyze` menu items.

TABLE 14–4 Descriptions for `analyze` Menu Item

Sub-Command	Description
`read`	Reads each sector on the current disk. Repairs defective blocks as a default.
`refresh`	Reads then writes data on the current disk without harming the data. Repairs defective blocks as a default.
`test`	Writes a set of patterns to the disk without harming the data. Repairs defective blocks as a default.
`write`	Writes a set of patterns to the disk then reads the data on the disk back. Destroys existing data on the disk. Repairs defective blocks as a default.
`compare`	Writes a set of patterns to the disk, reads the data back, and then compares it to the data in the write buffer. Destroys existing data on the disk. Repairs defective blocks as a default.
`purge`	Removes all data from the disk so that the data can't be retrieved by any means. Data is removed by writing three distinct patterns over the entire disk (or a section of the disk). If the verification passes, a hex-bit pattern is written over the entire disk (or a section of the disk). Repairs defective blocks as a default.
`verify`	Writes unique data to each block on the entire disk in the first pass. Reads and verifies the data in the next pass. Destroys existing data on the disk. Repairs defective blocks as a default.
`print`	Displays the data in the read/write buffer.

TABLE 14–4 Descriptions for `analyze` Menu Item *(Continued)*

Sub-Command	Description

`setup`	Lets you specify the following analysis parameters:
	`Analyze entire disk? yes`
	`Starting block number:` *depends on drive*
	`Ending block number:` *depends on drive*
	`Loop continuously? no`
	`Number of passes: 2`
	`    Repair defective blocks? yes`
	`Stop after first error? no`
	`Use random bit patterns? no`
	`Number of blocks per transfer: 126 (0/`*n*`/`*nn*`)`
	`Verify media after formatting? yes`
	`Enable extended messages? no`
	`Restore defect list? yes`
	`Restore disk label? yes`
	`Defaults are shown in bold.`

`config`	Displays the current analysis parameters.
`quit`	Exits the `analyze` menu.

The `defect` Menu

The `defect` menu looks similar to the following:

```
format> defect

DEFECT MENU:
        primary  - extract manufacturer's defect list
        grown    - extract manufacturer's and repaired defects lists
        both     - extract both primary and grown defects lists
        print    - display working list
        dump     - dump working list to file
        quit
defect>
```

The following table describes the `defect` menu items.

TABLE 14–5 The `defect` Menu Item Descriptions

Sub-Command	Description
`primary`	Reads the manufacturer's defect list from the disk drive and updates the in-memory defect list.
`grown`	Reads the grown defect list, which are defects that have been detected during analysis, and then updates the in-memory defect list.

TABLE 14–5 The `defect` Menu Item Descriptions *(Continued)*

Sub-Command	Description
`both`	Reads both the manufacturer's defect list and the grown defect list, and then updates the in-memory defect list.
`print`	Displays the in-memory defect list.
`dump`	Saves the in-memory defect list to a file.
`quit`	Exits the `defect` menu.

The `format.dat` File

The `format.dat` file that is shipped with the Solaris operating system supports many standard disks. If your disk drive is not listed in the `format.dat` file, you can choose to add an entry for it or adding entries with the `format` utility by selecting the `type` command and choosing the `other` option.

Adding an entry to the `format.dat` file can save time if the disk drive will be used throughout your site. To use the `format.dat` file on other systems, copy the file to each system that will use the specific disk drive that you added to the `format.dat` file.

You should modify the `/etc/format.dat` file for your system if you have one of the following:

- A disk that is not supported by the Solaris operating system
- A disk with a slice table that is different from the Solaris operating system's default configuration

Note – Do not alter default entries in the `/etc/format.dat` file. If you want to alter the default entries, copy the entry, give it a different name, and make the appropriate changes to avoid confusion.

The `/etc/format.dat` is not applicable for disks with EFI labels.

Contents of the `format.dat` File

The `format.dat` contains specific disk drive information that is used by the `format` utility. Three items are defined in the `format.dat` file:

- Search paths

- Disk types
- Slice tables

Syntax of the `format.dat` File

The following syntax rules apply to the `/etc/format.dat` file:

- The pound sign (`#`) is the comment character. Any text on a line after a pound sign is not interpreted by the `format` utility.
- Each definition in the `format.dat` file appears on a single logical line. If the definition is longer than one line long, all but the last line of the definition must end with a backslash (`\`).
- A definition consists of a series of assignments that have an identifier on the left side and one or more values on the right side. The assignment operator is the equal sign (`=`). The assignments within a definition must be separated by a colon (`:`).
- White space is ignored by the `format` utility. If you want an assigned value to contain white space, enclose the entire value in double quotation marks (`"`). This syntax will cause the white space within the quotes to be preserved as part of the assignment value.
- Some assignments can have multiple values on the right hand side. Separate values by a comma.

Keywords in the `format.dat` File

The `format.dat` file contains disk definitions that are read by the `format` utility when it is started. Each definition starts with one of the following keywords: `disk_type` or `partition`. These keywords are described in the following table.

TABLE 14–6 Keyword Descriptions for the `format.dat` File

Keyword	Use
`disk_type`	Defines the controller and disk model. Each `disk_type` definition contains information that concerns the physical geometry of the disk. The default data file contains definitions for the controllers and disks that the Solaris operating system supports. You need to add a new `disk_type` only if you have an unsupported disk. You can add as many `disk_type` definitions to the data file as you want.

TABLE 14–6 Keyword Descriptions for the `format.dat` File *(Continued)*

Keyword	Use
`partition`	Defines a slice table for a specific disk type. The slice table contains the slice information, plus a name that lets you refer to it in the `format` utility. The default `format.dat` file contains default slice definitions for several kinds of disk drives. Add a slice definition if you recreated slices on any of the disks on your system. Add as many slice definitions to the data file as you need.

Disk Type (`format.dat`)

The `disk_type` keyword in the `format.dat` file defines the controller and disk model. Each `disk_type` definition contains information about the physical geometry of the disk. The default `format.dat` file contains definitions for the controllers and disks that the Solaris operating system supports. You need to add a new `disk_type` only if you have an unsupported disk. You can add as many `disk_type` definitions to the data file as you want.

The keyword itself is assigned the name of the disk type. This name appears in the disk's label, and is used to identify the disk type whenever the `format` utility is run. Enclose the name in double quotation marks to preserve any white space in the name. The following table describes the identifiers that must also be assigned values in all `disk_type` definitions.

TABLE 14–7 Required `disk_type` Identifiers

Identifier	Description
`ctlr`	Identifies the controller type for the disk type. Currently, the supported values are SCSI and ATA.
`ncyl`	Specifies the number of data cylinders in the disk type. This determines how many logical cylinders of the disk the system will be allowed to access.
`acyl`	Specifies the number of alternate cylinders in the disk type. These cylinders are used by the `format` utility to store information such as the defect list for the drive. You should always leave at least two cylinders for alternates.
`pcyl`	Specifies the number of physical cylinders in the disk type. This number is used to calculate the boundaries of the disk media. This number is usually equal to `ncyl` plus `acyl`.
`nhead`	Specifies the number of heads in the disk type. This number is used to calculate the boundaries of the disk media.

TABLE 14–7 Required `disk_type` Identifiers *(Continued)*

Identifier	Description
`nsect`	Specifies the number of data sectors per track in the disk type. This number is used to calculate the boundaries of the disk media. Note that this is only the data sectors. Any spares are not reflected in the number of data sections per track.
`rpm`	The rotations per minute of the disk type. This information is put in the label and later used by the file system to calculate the optimal placement of file data.

Other identifiers might be necessary, depending on the controller. The following table describes the identifiers that are required for SCSI controllers.

TABLE 14–8 `disk_type` Identifiers for SCSI Controllers

Identifier	Description
`fmt_time`	A number that Indicates how long it takes to format a given drive. See the controller manual for more information.
`cache`	A number that controls the operation of the on-board cache while the `format` utility is operating. See the controller manual for more information.
`trks_zone`	A number that specifies how many tracks you have per defect zone, to be used in alternate sector mapping. See the controller manual for more information.
`asect`	A number that specifies how many sectors are available for alternate mapping within a given defect zone. See the controller manual for more information.

The following are examples of `disk_type` definitions:

```
disk_type = "SUN1.3G" \
        : ctlr = SCSI : fmt_time = 4 \
        : trks_zone = 17 : asect = 6 : atrks = 17 \
        : ncyl = 1965 : acyl = 2 : pcyl = 3500 : nhead = 17 : nsect = 80 \
        : rpm = 5400 : bpt = 44823

disk_type = "SUN2.1G" \
        : ctlr = SCSI : fmt_time = 4 \
        : ncyl = 2733 : acyl = 2 : pcyl = 3500 : nhead = 19 : nsect = 80 \
        : rpm = 5400 : bpt = 44823

disk_type = "SUN2.9G" \
        : ctlr = SCSI : fmt_time = 4 \
        : ncyl = 2734 : acyl = 2 : pcyl = 3500 : nhead = 21 : nsect = 99 \
        : rpm = 5400
```

Partition or Slice Tables (`format.dat`)

A partition table in the `format.dat` file defines a slice table for a specific disk type.

The `partition` keyword in the `format.dat` file is assigned the name of the slice table. Enclose the name in double quotation marks to preserve any white space in the name. The following table describes the identifiers that must be assigned values in all slice tables.

TABLE 14–9 Required Identifiers for Slice Tables

Identifier	Description
`disk`	The name of the `disk_type` that this slice table is defined for. This name must appear exactly as it does in the `disk_type` definition.
`ctlr`	The disk controller type that this slice table can be attached to. Currently, the supported values are ATA for ATA controllers and SCSI for SCSI controllers. The controller type that is specified here must also be defined for the `disk_type` that you specified in the `disk_type` definition.

The other identifiers in a slice definition describe the actual slice information. The identifiers are the numbers 0 through 7. These identifiers are optional. Any slice that is not explicitly assigned is set to 0 length. The value of each of these identifiers is a pair of numbers separated by a comma. The first number is the starting cylinder for the slice, and the second is the number of sectors in the slice. The following are some examples of slice definitions:

```
partition = "SUN1.3G" \
        : disk = "SUN1.3G" : ctlr = SCSI \
        : 0 = 0, 34000 : 1 = 25, 133280 : 2 = 0, 2672400 : 6 = 123, 2505120

partition = "SUN2.1G" \
        : disk = "SUN2.1G" : ctlr = SCSI \
        : 0 = 0, 62320 : 1 = 41, 197600 : 2 = 0, 4154160 : 6 = 171, 3894240

partition = "SUN2.9G" \
        : disk = "SUN2.9G" : ctlr = SCSI \
        : 0 = 0, 195426 : 1 = 94, 390852 : 2 = 0, 5683986 : 6 = 282, 5097708
```

Specifying an Alternate Data File for the `format` utility

The `format` utility learns of the location of an alternate file by the following methods.

1. If a file name is given with the `format -x` option, that file is always used as the data file.
2. If the `-x` option is not specified, then the `format` utility looks in the current directory for a file named `format.dat`. If the file exists, it is used as the data file.
3. If neither of these methods yields a data file, the `format` utility uses the `/etc/format.dat` file as the data file. This file is shipped with the Solaris operating system and should always be present.

Rules for Input to format Commands

When you use the format utility, you need to provide various kinds of information. This section describes the rules for this information. For information on using format's help facility when you enter data, see "Getting Help on the format Utility" on page 239.

Specifying Numbers to format Commands

Several places in the format utility require an number as input. You must either specify the data or select a number from a list of choices. In either case, the help facility causes format to print the upper and lower limits of the number expected. Simply enter the number desired. The number is assumed to be in decimal format unless a base is explicitly specified as part of the number (for example, 0x for hexadecimal).

The following are examples of integer input:

```
Enter number of passes [2]: 34
Enter number of passes [34] Oxf
```

Specifying Block Numbers to format Commands

Whenever you are required to specify a disk block number, there are two ways to enter the information:

- Block number as an integer
- Block number in the cylinder/head/sector format

You can specify the information as an integer that represents the logical block number. You can specify the number in any base, but the default is decimal. The maximum operator (a dollar sign, $) can also be used here to let the format utility select the appropriate value. Logical block format is used by the SunOS disk drivers in error messages.

The other way to specify a block number is by the cylinder/head/sector designation. In this method, you must specify explicitly the three logical components of the block number: the cylinder, head, and sector values. These values are still logical, but they allow you to define regions of the disk that are related to the layout of the media.

If any of the cylinder/head/sector numbers are not specified, the value is assumed to be zero. You can also use the maximum operator in place of any of the numbers and let the format utility select the appropriate value. The following are some examples of cylinder, head, and sector entries:

```
Enter defective block number: 34/2/3
Enter defective block number: 23/1/
Enter defective block number: 457//
Enter defective block number: 12345
Enter defective block number: Oxabcd
Enter defective block number: 334/$/2
Enter defective block number: 892//$
```

The `format` utility always prints block numbers, in both formats. Also, the `help` facility shows you the upper and lower bounds of the block number expected, in both formats.

Specifying `format` Command Names

Command names are needed as input whenever the `format` utility displays a menu prompt. You can *abbreviate* the command names, as long as what you enter is sufficient to uniquely identify the command desired.

For example, use `p` to enter the `partition` menu from the `format` menu. Then, enter `p` to display the current slice table.

```
format> p
PARTITION MENU:
        0      - change ‘0’ partition
        1      - change ‘1’ partition
        2      - change ‘2’ partition
        3      - change ‘3’ partition
        4      - change ‘4’ partition
        5      - change ‘5’ partition
        6      - change ‘6’ partition
        7      - change ‘7’ partition
        select - select a predefined table
        modify - modify a predefined partition table
        name   - name the current table
        print  - display the current table
        label  - write partition map and label to the disk
        quit
partition> p
```

Specifying Disk Names to `format` Commands

There are certain times in the `format` utility when you must name something. In these cases, you are free to specify any string you want for the name. If the name has white space in it, the entire name must be enclosed in double quotation marks ("). Otherwise, only the first word of the name is used.

For example, if you want to identify a specific partition table for a disk, you can use the name sub-command available from the partition menu:

```
partition> name
Enter table name (remember quotes): "new disk3"
```

Getting Help on the `format` Utility

The `format` utility provides a help facility that you can use whenever the `format` utility is expecting input. You can request help about what input is expected by entering a question mark (`?`). The `format` utility displays a brief description of what type of input is needed.

If you enter a `?` at a menu prompt, a list of available commands is displayed.

The man pages associated with the `format` utility include the following:

- `format`(1M) - Describes the basic `format` utility capabilities and provides descriptions of all command-line variables.
- `format.dat`(4) - Describes disk drive configuration information for the `format` utility.

CHAPTER 15

Managing File Systems (Overview)

The management of file systems is one of your most important system administration tasks.

This is a list of the overview information in this chapter.

What's New in File Systems in the Solaris 9 Update Releases?

This section describes a new file system feature in this Solaris release.

UFS Logging Is Enabled by Default

Solaris 9 9/04 – Logging is enabled by default for all UFS file systems except under the following conditions:

- When logging is explicitly disabled.

- If there is insufficient file system space for the log.

In previous Solaris releases, you had to manually enable UFS logging. For more information about UFS logging, see "UFS Logging" on page 259.

Keep the following issues in mind when using UFS logging in this release:

- Ensure that you have enough disk space for your general system needs, such as for users and applications, and for UFS logging.
- If you don't have enough disk space for logging data, a message similar to the following is displayed:

  ```
  # mount /dev/dsk/c0t4d0s0 /mnt
  /mnt: No space left on device
  Could not enable logging for /mnt on /dev/dsk/c0t4d0s0.
  #
  ```

  However, the file system is still mounted. For example:

  ```
  # df -h /mnt
  Filesystem             size   used  avail capacity  Mounted on
  /dev/dsk/c0t4d0s0      142M   142M     0K   100%    /mnt
  #
  ```
- A UFS file system with logging enabled that is generally empty will have some disk space consumed for the log.
- If you upgrade to this Solaris release from a previous Solaris release, your UFS file systems will have logging enabled, even if the `logging` option was not specified in the `/etc/vfstab` file. To disable logging, add the `nologging` option to the UFS file system entries in the `/etc/vfstab` file.

Default Logging and Standards Conformance

UFS file system transactions that free blocks from files might not immediately add the freed blocks to the file system's free list. This behavior occurs on a system that has a UFS file system mounted with logging enabled.

This behavior improves file system performance, but does not conform to the following standards:

- POSIX, Single UNIX® Specification
- SPARC® Conformance Definition
- SPARC Conformance Definition System V Application Binary Interface
- System V Interface Definition
- X/Open® Portability Guide

These standards require that freed space be available immediately.

Consider disabling UFS logging under the following conditions:

- You want to enable standards conformance regarding file deletions.

- You encounter problems creating or growing files immediately after the files have been deleted on a relatively full file system.

For more information, see mount_ufs(1M).

SPARC: Support of Multiterabyte UFS File Systems

Solaris 9 8/03 – This Solaris release provides support for multiterabyte UFS file systems on systems that run a 64-bit Solaris kernel.

Previously, UFS file systems were limited to approximately 1 terabyte on both 64-bit and 32-bit systems. All UFS file system commands and utilities have been updated to support multiterabyte UFS file systems.

For example, the ufsdump command has been updated with a larger block size for dumping large UFS file systems:

```
# ufsdump 0f /dev/md/rdsk/d97 /dev/md/rdsk/d98
  DUMP: Date of this level 0 dump: Tue Jan 07 14:23:36 2003
  DUMP: Date of last level 0 dump: the epoch
  DUMP: Dumping /dev/md/rdsk/d98 to /dev/md/rdsk/d97.
  DUMP: Mapping (Pass I) [regular files]
  DUMP: Mapping (Pass II) [directories]
  DUMP: Forcing larger tape block size (2048).
  DUMP: Writing 32 Kilobyte records
  DUMP: Estimated 4390629500 blocks (2143862.06MB).
  DUMP: Dumping (Pass III) [directories]
  DUMP: Dumping (Pass IV) [regular files]
```

Administering UFS file systems that are less than 1 terabyte remains the same. No administration differences exist between UFS file systems that are less than one terabyte and file systems that are greater than 1 terabyte.

You can initially create a UFS file system that is less than 1 terabyte and specify that it can eventually be expanded into a multiterabyte file system by using the newfs -T option. This option sets the inode and fragment density to scale appropriately for a multiterabyte file system.

Using the newfs -T option when you create a UFS file system less than 1 terabyte on a system running a 32-bit kernel enables you to eventually expand this file system with the growfs command when you boot this system under a 64-bit kernel. For more information, see newfs(1M).

You can use the growfs command to expand a UFS file system to the size of the slice or the volume without loss of service or data. For more information, see growfs(1M).

Two new related features are multiterabyte volume support with the EFI disk label and multiterabyte volume support with Solaris Volume Manager. For more information, see "SPARC: Multiterabyte Disk Support With EFI Disk Label" on page 159 and the *Solaris Volume Manager Administration Guide*

Features of Multiterabyte UFS File Systems

Multiterabyte UFS file systems include the following features:

- The ability to create a UFS file system up to 16 terabytes in size.
- The ability to create a file system less than 16 terabytes that can later be increased in size up to 16 terabytes.
- Multiterabyte file systems can be created on physical disks, Solaris Volume Manager's logical volumes, and Veritas' VxVM logical volumes.
- Multiterabyte file systems benefit from the performance improvements of having UFS logging enabled. Multiterabyte file systems also benefit from the availability of logging because the `fsck` command might not have to be run when logging is enabled.
- When you create a partition for your multiterabyte UFS file system, the disk will be labeled automatically with an EFI disk label. For more information on EFI disk labels, see "SPARC: Multiterabyte Disk Support With EFI Disk Label" on page 159.

Limitations of Multiterabyte UFS File Systems

Limitations of multiterabyte UFS file systems are as follows:

- This feature is not supported on Solaris x86 systems.
- You cannot mount a file system greater than 1 terabyte on a system that is running a 32-bit Solaris kernel.
- You cannot boot from a file system greater than 1 terabyte on a system that is running a 64-bit Solaris kernel. This limitation means that you cannot put a root (/) file system on a multiterabyte file system.
- There is no support for individual files greater than 1 terabyte.
- The maximum number of files is 1 million files per terabyte of UFS file system. For example, a 4–terabyte file system can contain 4 million files.

  This limit is intended to reduce the time it takes to check the file system with the `fsck` command.
- The maximum quota that you can set on a multiterabyte UFS file system is 2 terabytes of 1024-byte blocks.
- Using the `fssnap` command to create a snapshot of a multiterabyte UFS file system is not currently supported.

▼ How to Create a Multiterabyte UFS File System

Support for a multiterabyte UFS file system assumes the availability of multiterabyte LUNs, provided as Solaris Volume Manager or VxVM volumes, or as physical disks greater than 1 terabyte.

Before you can create a multiterabyte UFS file system, verify that you have done either of the following:

- Created a multiterabyte disk partition with the `format` utility or the Solaris installation utilities.
- Set up a multiterabyte volume with Solaris Volume Manager.

Steps

1. **Become superuser.**
2. **Create a multiterabyte UFS file system on a logical volume.**

   For example, this command creates a UFS file system for a 1.8 terabyte volume.

```
# newfs /dev/md/rdsk/d99
newfs: construct a new file system /dev/md/rdsk/d99: (y/n)? y
/dev/md/rdsk/d99:    3859402752 sectors in 628158 cylinders of 48 tracks,
128 sectors
        1884474.0MB in 4393 cyl groups (143 c/g, 429.00MB/g, 448 i/g)
super-block backups (for fsck -F ufs -o b=#) at:
32, 878752, 1757472, 2636192, 3514912, 4393632, 5272352, 6151072, 702...
Initializing cylinder groups:
........................................................................
super-block backups for last 10 cylinder groups at:
 3850872736, 3851751456, 3852630176, 3853508896, 3854387616, 3855266336,
 3856145056, 3857023776, 3857902496, 3858781216,
#
```

3. **Verify the integrity of the newly created file system.**

   For example:

```
# fsck /dev/md/rdsk/d99
** /dev/md/rdsk/d99
** Last Mounted on
** Phase 1 - Check Blocks and Sizes
** Phase 2 - Check Pathnames
** Phase 3 - Check Connectivity
** Phase 4 - Check Reference Counts
** Phase 5 - Check Cyl groups
2 files, 2 used, 241173122 free (0 frags, 241173122 blocks, 0.0%
fragmentation)
#
```

4. **Mount and verify the newly created file system.**

   For example:

```
# mount /dev/md/dsk/d99 /bigdir
# df -h /bigdir
Filesystem             size   used  avail capacity  Mounted on
/dev/md/dsk/d99        1.8T    64M   1.8T     1%    /bigdir
```

▼ How to Expand a Multiterabyte UFS File System

After a multiterabyte UFS file system is created, you can use the `growfs` command to expand the file system. For example, using the file system that was created for the volume in the preceding procedure, you can add another disk to this volume. Then, expand the file system.

Steps

1. **Become superuser.**
2. **Add another disk to the volume.**

   For example:

```
# metattach d99 c4t5d0s4
d99: component is attached
# metastat
d99: Concat/Stripe
    Size: 5145882624 blocks (2.4 TB)
    Stripe 0:
        Device     Start Block  Dbase   Reloc
        c0t1d0s4      36864     Yes     Yes
    Stripe 1:
        Device     Start Block  Dbase   Reloc
        c3t7d0s4          0     No      Yes
    Stripe 2:
        Device     Start Block  Dbase   Reloc
        c1t1d0s4          0     No      Yes
    Stripe 3:
        Device     Start Block  Dbase   Reloc
        c4t5d0s4          0     No      Yes
```

3. **Expand the file system.**

   For example:

```
# growfs -v /dev/md/rdsk/d99
/usr/lib/fs/ufs/mkfs -G /dev/md/rdsk/d99 5145882624
/dev/md/rdsk/d99:    5145882624 sectors in 837546 cylinders of 48 tracks,
128 sectors
        2512638.0MB in 5857 cyl groups (143 c/g, 429.00MB/g, 448 i/g)
super-block backups (for fsck -F ufs -o b=#) at:
 32, 878752, 1757472, 2636192, 3514912, 4393632, 5272352, 6151072, 702...
Initializing cylinder groups:
.......................................................................
super-block backups for last 10 cylinder groups at:
 5137130400, 5138009120, 5138887840, 5139766560, 5140645280, 5141524000,
 5142402720, 5143281440, 5144160160, 5145038880,
#
```

4. **Mount and verify the expanded file system.**

   For example:

```
# mount /dev/md/dsk/d99 /bigdir
# df -h /bigdir
Filesystem             size   used  avail capacity  Mounted on
/dev/md/dsk/d99        2.4T    64M   2.4T     1%    /bigdir
```

▼ How to Expand a UFS File System to a Multiterabyte UFS File System

Use the following procedure to expand a UFS file system to greater than 1 terabyte in size. This procedure assumes that the `newfs -T` option was used initially to create the UFS file system.

Steps

1. **Become superuser.**
2. **Identify the size of the current disk or volume.**

   For example, the following volume is 800 gigabytes.

```
# metastat d98
d98: Concat/Stripe
    Size: 1677754368 blocks (800 GB)
    Stripe 0:
        Device     Start Block  Dbase   Reloc
        c0t1d0s4          0     No      Yes
    Stripe 1:
        Device     Start Block  Dbase   Reloc
        c3t7d0s4          0     No      Yes
```

3. **Increase the volume to greater than 1 terabyte.**

   For example:

```
# metattach d98 c1t1d0s4
d98: component is attached
# metastat d98
d98: Concat/Stripe
    Size: 2516631552 blocks (1.2 TB)
    Stripe 0:
        Device     Start Block  Dbase   Reloc
        c0t1d0s4          0     No      Yes
    Stripe 1:
        Device     Start Block  Dbase   Reloc
        c3t7d0s4          0     No      Yes
    Stripe 2:
        Device     Start Block  Dbase   Reloc
        c1t1d0s4          0     No      Yes
```

4. **Expand the UFS file system for the disk or volume to greater than 1 terabyte.**

   For example:

```
growfs -v /dev/md/rdsk/d98
/usr/lib/fs/ufs/mkfs -G /dev/md/rdsk/d98 2516631552
/dev/md/rdsk/d98:    2516631552 sectors in 68268 cylinders of 144 tracks,
256 sectors
        1228824.0MB in 2731 cyl groups (25 c/g, 450.00MB/g, 448 i/g)
super-block backups (for fsck -F ufs -o b=#) at:
 32, 921888, 1843744, 2765600, 3687456, 4609312, 5531168, 6453024, 737...
 8296736,
Initializing cylinder groups:
```

```
.....................................................
super-block backups for last 10 cylinder groups at:
 2507714848, 2508636704, 2509558560, 2510480416, 2511402272, 2512324128,
 2513245984, 2514167840, 2515089696, 2516011552,
```

5. **Mount and verify the expanded file system.**

   For example:

   ```
   # mount /dev/md/dsk/d98 /datadir
   # df -h /datadir
   Filesystem             size   used  avail capacity  Mounted on
   /dev/md/dsk/d98        1.2T    64M   1.2T     1%    /datadir
   ```

Troubleshooting Multiterabyte UFS File System Problems

Use the following error messages and solutions to troubleshoot problems with multiterabyte UFS file systems.

Error Message (similar to the following):

```
mount: /dev/rdsk/c0t0d0s0 is not this fstype.
```

Cause

You attempted to mount a UFS file system that is greater than 1 terabyte on a system running a Solaris release prior to the Solaris 9 8/03 release.

Solution

Mount a UFS file system that is greater than 1 terabyte on a system running the Solaris 9 8/03 or later release.

Error Message

```
"File system was not set up with the multi-terabyte format."  "Its size
cannot be increased to a terabyte or more."
```

Cause

You attempted to expand a file system that was not created with the `newfs -T` command.

Solution

1. Back up the data for the file system that you want to expand to greater than one terabyte.
2. Re-create the file system with the `newfs` command to create a multiterabyte file system.
3. Restore the backup data into the newly created file system.

Where to Find File System Management Tasks

Use these references to find step-by-step instructions for the management of file systems.

File System Management Task	For More Information
Create new file systems	Chapter 16 and Chapter 18
Make local and remote files available to users	Chapter 17
Connect and configure new disk devices	Chapter 10
Design and implement a backup schedule and restoring files and file systems, as needed	Chapter 22
Check for and correct file system inconsistencies	Chapter 20

Overview of File Systems

A file system is a structure of directories that is used to organize and store files. The term *file system* is used to describe the following:

- A particular type of file system: disk-based, network-based, or virtual
- The entire file tree, beginning with the root directory
- The data structure of a disk slice or other media storage device
- A portion of a file tree structure that is attached to a mount point on the main file tree so that the files are accessible

Usually, you can tell from the context which meaning is intended.

The Solaris operating system uses the *virtual file system* (VFS) architecture, which provides a standard interface for different file system types. The VFS architecture enables the kernel to handle basic operations, such as reading, writing, and listing files, and makes it easier to add new file systems.

Types of File Systems

The Solaris operating system supports three types of file systems:

- Disk-based
- Network-based
- Virtual

To identify the file system type, see "Determining a File System's Type" on page 265.

Disk-Based File Systems

Disk-based file systems are stored on physical media such as hard disks, CD-ROMs, and diskettes. Disk-based file systems can be written in different formats. The available formats are the following:

Disk-Based File System	Format Description
UFS	UNIX file system (based on the BSD Fast File system that was provided in the 4.3 Tahoe release). UFS is the default disk-based file system for the Solaris operating system. Before you can create a UFS file system on a disk, you must format the disk and divide it into slices. For information on formatting disks and dividing disks into slices, see Chapter 10.
HSFS	High Sierra, Rock Ridge, and ISO 9660 file system. High Sierra is the first CD-ROM file system. ISO 9660 is the official standard version of the High Sierra File System. The HSFS file system is used on CD-ROMs, and is a read-only file system. Solaris HSFS supports Rock Ridge extensions to ISO 9660, which, when present on a CD-ROM, provide all UFS file system features and file types, except for writability and hard links.
PCFS	PC file system, which allows read and write access to data and programs on DOS-formatted disks that are written for DOS-based personal computers.
UDF	The Universal Disk Format (UDF) file system, the industry-standard format for storing information on the optical media technology called DVD (Digital Versatile Disc or Digital Video Disc).

Each type of disk-based file system is customarily associated with a particular media device, as follows:

- UFS with hard disk

- HSFS with CD-ROM
- PCFS with diskette
- UDF with DVD

These associations are not, however, restrictive. For example, CD-ROMs and diskettes can have UFS file systems created on them.

Network-Based File Systems

Network-based file systems can be accessed from the network. Typically, network-based file systems reside on one system, typically a server, and are accessed by other systems across the network.

With NFS, you can administer distributed *resources* (files or directories) by exporting them from a server and mounting them on individual clients. For more information, see "The NFS Environment" on page 264.

Virtual File Systems

Virtual file systems are memory-based file systems that provide access to special kernel information and facilities. Most virtual file systems do not use file system disk space. However, the CacheFS file system uses a file system on the disk to contain the cache. Also, some virtual file systems, such as the temporary file system (TMPFS), use the swap space on a disk.

The CacheFS File System

The CacheFS™ file system can be used to improve performance of remote file systems or slow devices such as CD-ROM drives. When a file system is cached, the data that is read from the remote file system or CD-ROM is stored in a cache on the local system.

If you want to improve the performance and scalability of an NFS or CD-ROM file system, you should use the CacheFS file system. The CacheFS software is a general purpose caching mechanism for file systems that improves NFS server performance and scalability by reducing server and network load.

Designed as a layered file system, the CacheFS software provides the ability to cache one file system on another. In an NFS environment, CacheFS software increases the client per server ratio, reduces server and network loads, and improves performance for clients on slow links, such as Point-to-Point Protocol (PPP). You can also combine a CacheFS file system with the AutoFS service to help boost performance and scalability.

For detailed information about the CacheFS file system, see Chapter 18.

The Universal Disk Format (UDF) File System

The UDF file system is the industry-standard format for storing information on the *DVD* (Digital Versatile Disc or Digital Video Disc) optical media.

The UDF file system is provided as dynamically loadable, 32-bit and 64-bit modules, with system administration utilities for creating, mounting, and checking the file system on both SPARC and x86 platforms. The Solaris UDF file system works with supported ATAPI and SCSI DVD drives, CD-ROM devices, and disk and diskette drives. In addition, the Solaris UDF file system is fully compliant with the UDF 1.50 specification.

The UDF file system provides the following features:

- Ability to access the industry standard CD-ROM and DVD-ROM media when they contain a UDF file system
- Flexibility in exchanging information across platforms and operating systems
- A mechanism for implementing new applications rich in broadcast-quality video, high-quality sound along with the richness in interactivity using the DVD video specification based on UDF format

The following features are not included in the UDF file system:

- Support for write-once media, CD-RW, and DVD-RAM, with either the sequential disk-at-once and incremental recording
- UFS components such as quotas, ACLs, transaction logging, file system locking, and file system threads, which are not part of the UDF 1.50 specification

The UDF file system requires the following:

- At least the Solaris 7 11/99 release
- Supported SPARC or x86 platforms
- Supported CD-ROM or DVD-ROM device

The Solaris UDF file system implementation provides:

- Support for industry-standard read/write UDF version 1.50
- Fully internationalized file system utilities

Temporary File System

The temporary file system (TMPFS) uses local memory for file system reads and writes, which is typically much faster than a UFS file system. Using TMPFS can improve system performance by saving the cost of reading and writing temporary files to a local disk or across the network. For example, temporary files are created when you compile a program, and the operating system generates a lot of disk activity or network activity while manipulating these files. Using TMPFS to hold these temporary files can significantly speed up their creation, manipulation, and deletion.

Files in TMPFS file systems are not permanent. They are deleted when the file system is unmounted and when the system is shut down or rebooted.

TMPFS is the default file system type for the `/tmp` directory in the Solaris operating system. You can copy or move files into or out of the `/tmp` directory, just as you would in a UFS file system.

The TMPFS file system uses swap space as a temporary backing store. If a system with a TMPFS file system does not have adequate swap space, two problems can occur:

- The TMPFS file system can run out of space, just as regular file systems do.
- Because TMPFS allocates swap space to save file data (if necessary), some programs might not execute because of insufficient swap space.

For information about creating TMPFS file systems, see Chapter 16. For information about increasing swap space, see Chapter 19.

The Loopback File System

The loopback file system (LOFS) lets you create a new virtual file system so that you can access files by using an alternative path name. For example, you can create a loopback mount of root `(/)` on `/tmp/newroot`, which will make the entire file system hierarchy look like it is duplicated under `/tmp/newroot`, including any file systems mounted from NFS servers. All files will be accessible either with a path name starting from root (`/`), or with a path name that starts from `/tmp/newroot`.

For information on how to create LOFS file systems, see Chapter 16.

Process File System

The process file system (PROCFS) resides in memory and contains a list of active processes, by process number, in the `/proc` directory. Information in the `/proc` directory is used by commands like `ps`. Debuggers and other development tools can also access the address space of the processes by using file system calls.

Caution – Do not delete the files in the `/proc` directory. The deletion of processes from the `/proc` directory does not kill them. Remember, `/proc` files do not use disk space, so there is little reason to delete files from this directory.

The `/proc` directory does not require administration.

Additional Virtual File Systems

These additional types of virtual file systems are listed for your information. They do not require administration.

Virtual File System	Description
FIFOFS (first-in first-out)	Named pipe files that give processes common access to data
FDFS (file descriptors)	Provides explicit names for opening files using file descriptors
MNTFS	Provides read-only access to the table of mounted file systems for the local system
NAMEFS	Used mostly by STREAMS for dynamic mounts of file descriptors on top of files
SPECFS (special)	Provides access to character special devices and block devices
SWAPFS	Used by the kernel for swapping

Extended File Attributes

The UFS, NFS, and TMPFS file systems have been enhanced to include extended file attributes, which enable application developers to associate specific attributes to a file. For example, a developer of a windowing system file management application might choose to associate a display icon with a file. Extended file attributes are logically represented as files within a hidden directory that is associated with the target file.

You can use the `runat` command to add attributes and execute shell commands in the extended attribute name space, which is a hidden attribute directory that is associated with the specified file.

To use the `runat` command to add attributes to a file, you first have to create the attributes file.

```
$ runat filea cp /tmp/attrdata attr.1
```

Then, use the `runat` command to list the attributes of a file.

```
$ runat filea ls -l
```

For more information, see the `runat`(1) man page.

Many Solaris file system commands have been modified to support file system attributes by providing an attribute-aware option that you can use to query, copy, or find file attributes. For more information, see the specific man page for each file system command.

Commands for File System Administration

Most commands for file system administration have both a generic component and a file system–specific component. Whenever possible, you should use the generic commands, which call the file system–specific component. The following table lists the generic commands for file system administration, which are located in the `/usr/sbin` directory.

TABLE 15–1 Generic Commands for File System Administration

Command	Man Page	Description
`clri`	`clri`(1M)	Clears inodes
`df`	`df`(1M)	Reports the number of free disk blocks and files
`ff`	`ff`(1M)	Lists file names and statistics for a file system
`fsck`	`fsck`(1M)	Checks the integrity of a file system and repairs any damage found
`fsdb`	`fsdb`(1M)	Debugs the file system
`fstyp`	`fstyp`(1M)	Determines the file system type
`labelit`	`labelit`(1M)	Lists or provides labels for file systems when they are copied to tape (for use by the `volcopy` command only)
`mkfs`	`mkfs`(1M)	Creates a new file system
`mount`	`mount`(1M)	Mounts local and remote file systems
`mountall`	`mountall`(1M)	Mounts all file systems that are specified in the virtual file system table (`/etc/vfstab`)
`ncheck`	`ncheck`(1M)	Generates a list of path names with their inode numbers
`umount`	`mount`(1M)	Unmounts local and remote file systems
`umountall`	`mountall`(1M)	Unmounts all file systems that are specified in a virtual file system table (`/etc/vfstab`)
`volcopy`	`volcopy`(1M)	Creates an image copy of a file system

How File System Commands Determine the File System Type

The generic file system commands determine the file system type by following this sequence:

1. From the `-F` option, if supplied.
2. By matching a special device with an entry in the `/etc/vfstab` file (if *special* is supplied). For example, `fsck` first looks for a match against the `fsck device` field. If no match is found, it then checks the *special* device field.
3. By using the default specified in the `/etc/default/fs` file for local file systems and in the `/etc/dfs/fstypes` file for remote file systems.

Manual Pages for Generic and Specific Commands

Both the generic commands and specific commands have manual pages in the *man pages section 1M: System Administration Commands*. The manual page for the generic file system commands provide information about generic command options only. The manual page for a specific file system command has specific information about options for that file system. To look at a specific manual page, append an underscore and the abbreviation for the file system type to the generic command name. For example, to see the specific manual page for mounting a UFS file system, type the following:

```
$ man mount_ufs
```

The Default Solaris File Systems

The Solaris UFS file system is hierarchical, starting with the root directory (`/`) and continuing downwards through a number of directories. The Solaris installation process enables you to install a default set of directories and uses a set of conventions to group similar types of files together. The following table provides a summary of the default Solaris file systems.

TABLE 15–2 The Default Solaris File Systems

File System or Directory	File System Type	Description
root (/)	UFS	The top of the hierarchical file tree. The root directory contains the directories and files that are critical for system operation, such as the kernel, the device drivers, and the programs used to boot the system. The root directory also contains the mount point directories where local and remote file systems can be attached to the file tree.
`/usr`	UFS	System files and directories that can be shared with other users. Files that run only on certain types of systems are in the `/usr` file system (for example, SPARC executables). Files that can be used on all types of systems, such as the man pages, are in the `/usr/share` directory.
`/export/home` or `/home`	NFS, UFS	The mount point for users' home directories, which store user work files. By default the `/home` directory is an automounted file system. On standalone systems, the `/home` directory might be a UFS file system on a local disk slice.
`/var`	UFS	System files and directories that are likely to change or grow over the life of the local system. These include system logs, `vi` and `ex` backup files, and `uucp` files.
`/opt`	NFS, UFS	Optional mount point for third-party software. On some systems, the `/opt` directory might be a UFS file system on a local disk slice.
`/tmp`	TMPFS	Temporary files, which are cleared each time the system is booted or the `/tmp` file system is unmounted.
`/proc`	PROCFS	A list of active processes, by number.
`/etc/mnttab`	MNTFS	A file system that provides read-only access to the table of mounted file systems for the local system.
`/var/run`	TMPFS	A file system for storing temporary files that are not needed after the system is booted.

The root (/) and `/usr` file systems are needed to run a system. Some of the most basic commands in the `/usr` file system (like `mount`) are included in the root (/) file system so that they are available when the system boots or is in single-user mode and `/usr` is not mounted. For more detailed information on the default directories for the root (/) and `/usr` file systems, see Chapter 21.

Swap Space

The Solaris operating system uses some disk slices for temporary storage rather than for file systems. These slices are called *swap* slices, or *swap space*. Swap space is used as virtual memory storage areas when the system does not have enough physical memory to handle current processes.

Since many applications rely on swap space, you should know how to plan for, monitor, and add more swap space when needed. For an overview about swap space and instructions for adding swap space, see Chapter 19.

The UFS File System

UFS is the default disk-based file system in Solaris operating system. Most often, when you administer a disk-based file system, you will be administering UFS file systems. UFS provides the following features:

UFS Feature	Description
State flags	Show the state of the file system: clean, stable, active, logging, or unknown. These flags eliminate unnecessary file system checks. If the file system is “clean,” “stable,” or “logging,” file system checks are not run.
Extended fundamental types (EFT)	Provides 32-bit user ID (UID), group ID (GID), and device numbers.
Large file systems	Allows files of approximately 1 terabyte in size in a file system that can be up to 16 terabytes in size. You can create a multiterabyte UFS file system on a disk with an EFI disk label.

For detailed information about the UFS file system structure, see Chapter 21.

Planning UFS File Systems

When laying out file systems, you need to consider possible conflicting demands. Here are some suggestions:

- Distribute the work load as evenly as possible among different I/O systems and disk drives. Distribute the `/export/home` file system and swap space evenly across disks.
- Keep pieces of projects or members of groups within the same file system.
- Use as few file systems per disk as possible. On the system (or boot) disk, you should have three file systems: root (`/`), `/usr`, and swap space. On other disks, create one or, at most, two file systems; one being additional swap space, preferably. Fewer, roomier file systems cause less file fragmentation than many small, over-crowded file systems. Higher-capacity tape drives and the ability of the `ufsdump` command to handle multiple volumes make it easier to back up larger file systems.
- If you have some users who consistently create very small files, consider creating a separate file system with more inodes. However, most sites do not need to keep similar types of user files in the same file system.

For information on default file system parameters as well as procedures for creating new UFS file systems, see Chapter 16.

UFS Logging

UFS logging bundles the multiple metadata changes that make up a complete UFS operation into a transaction. Sets of transactions are recorded in an on-disk log, and then applied to the actual UFS file system's metadata.

At reboot, the system discards incomplete transactions, but applies the transactions for completed operations. The file system remains consistent because only completed transactions are ever applied. This consistency remains even when a system crashes, which normally interrupts system calls and introduces inconsistencies into a UFS file system.

UFS logging provides two advantages:

- If the file system is already consistent due to the transaction log, you might not have to run the `fsck` command after a system crash or an unclean shutdown. For more information on unclean shutdowns, see "What the `fsck` Command Checks and Tries to Repair" on page 333.
- Starting in the Solaris 9 12/02 release, the performance of UFS logging improves or exceeds the level of performance of non-logging file systems. This improvement can occur because a file system with logging enabled converts multiple updates to the same data into single updates, and so reduces the number of overhead disk operations required.

The log is allocated from free blocks on the file system, and it is sized at approximately 1 Mbyte per 1 Gbyte of file system, up to a maximum of 64 Mbytes. The log is continually flushed as it fills up. The log is also flushed when the file system is unmounted or as a result of any `lockfs` command.

UFS logging is enabled by default for all UFS file systems.

If you need to disable UFS logging, add the nologging option to the file system's entry in the `/etc/vfstab` file or when you mount the file system manually.

If you need to enable UFS logging, specify the `-o logging` option with the `mount` command in the `/etc/vfstab` file or when you mount the file system manually. Logging can be enabled on any UFS file system, including the root (`/`) file system. Also, the `fsdb` command now has new debugging commands to support UFS logging.

In some operating systems, a file system with logging enabled is known as a *journaling* file system.

UFS Snapshots

You can use the `fssnap` command to create a read-only snapshot of a file system. A snapshot is a file system's temporary image that is intended for backup operations.

See Chapter 24 for more information.

UFS Direct Input/Output (I/O)

Direct I/O is intended to boost bulk I/O operations. Bulk I/O operations use large buffer sizes to transfer large files (larger than 256 Kbytes).

Using UFS direct I/O might benefit applications, such as database engines, that do their own internal buffering. Starting with the Solaris 8 1/01 release, UFS direct I/O has been enhanced to allow the same kind of I/O concurrency seen when accessing raw devices. Now you can get the benefit of file system naming and flexibility with very little performance penalty. Check with your database vendor to see if they can enable UFS direct I/O in their product configuration options.

Direct I/O can also be enabled on a file system by using the `forcedirectio` option to the `mount` command. Enabling direct I/O is a performance benefit only when a file system is transferring large amounts of sequential data.

When a file system is mounted with this option, data is transferred directly between a user's address space and the disk. When forced direct I/O is not enabled for a file system, data transferred between a user's address space and the disk is first buffered in the kernel address space.

The default behavior is no forced direct I/O on a UFS file system. For more information, see `mount_ufs`(1M).

Mounting and Unmounting File Systems

Before you can access the files on a file system, you need to mount the file system. When you mount a file system, you attach that file system to a directory (*mount point*) and make it available to the system. The root (/) file system is always mounted. Any other file system can be connected or disconnected from the root (/) file system.

When you mount a file system, any files or directories in the underlying mount point directory are unavailable as long as the file system is mounted. These files are not permanently affected by the mounting process, and they become available again when the file system is unmounted. However, mount directories are typically empty, because you usually do not want to obscure existing files.

For example, the following figure shows a local file system, starting with a root (/) file system and the `sbin`, `etc`, and `opt` subdirectories.

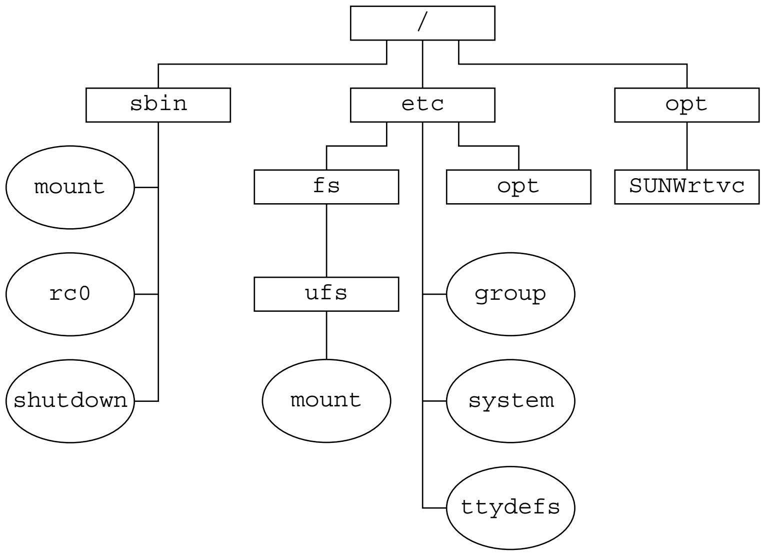

FIGURE 15–1 Sample root (/) File System

To access a local file system from the `/opt` file system that contains a set of unbundled products, you must do the following:

- First, you must create a directory to use as a mount point for the file system you want to mount, for example, `/opt/unbundled`.
- Once the mount point is created, you can mount the file system (by using the `mount` command), which makes all of the files and directories in `/opt/unbundled` available, as shown in the following figure.

For step-by-step instructions on how to mount file systems, see Chapter 17.

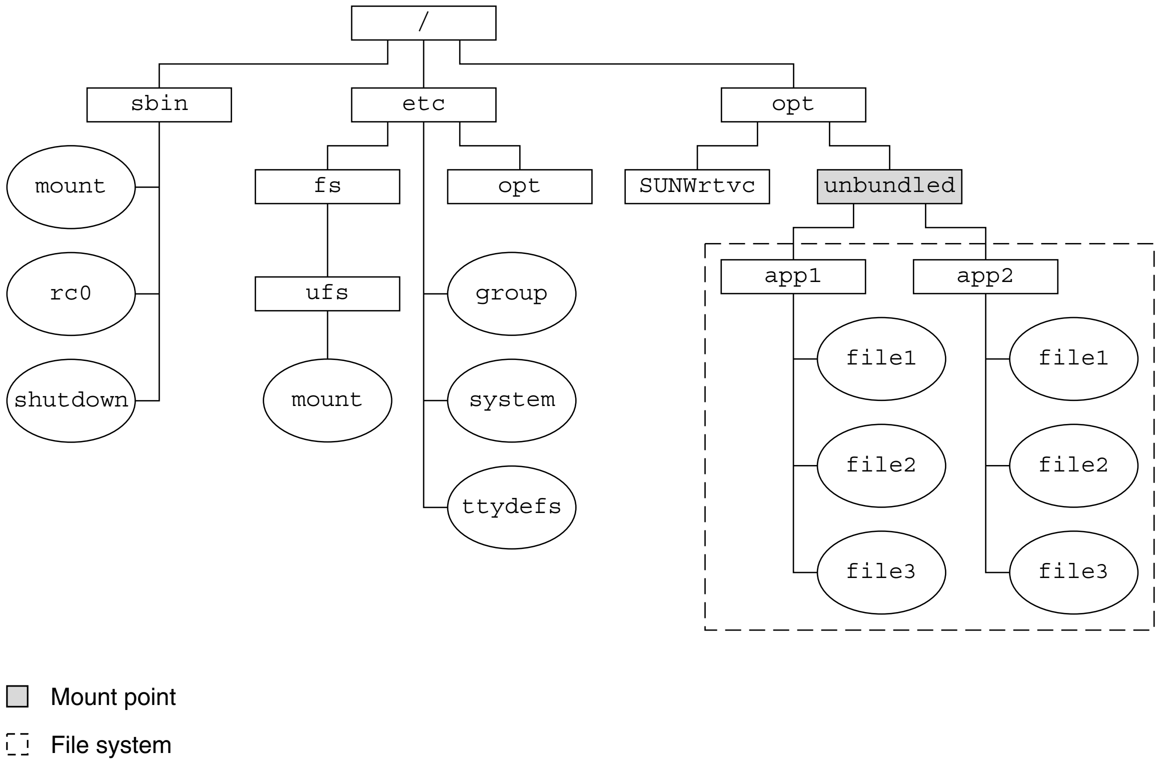

FIGURE 15–2 Mounting a File System

The Mounted File System Table

Whenever you mount or unmount a file system, the `/etc/mnttab` (mount table) file is modified with the list of currently mounted file systems. You can display the contents of this file with the `cat` or `more` commands, but you cannot edit it. Here is an example of an `/etc/mnttab` file:

```
$ more /etc/mnttab
/dev/dsk/c0t0d0s0  /  ufs rw,intr,largefiles,onerror=panic,suid,dev=2200000 938557523
/proc   /proc   proc    dev=3180000     938557522
fd      /dev/fd fd      rw,suid,dev=3240000     938557524
mnttab  /etc/mnttab     mntfs   dev=3340000     938557526
swap    /var/run        tmpfs   dev=1   938557526
swap    /tmp    tmpfs   dev=2   938557529
/dev/dsk/c0t0d0s7 /export/home ufs rw,intr,largefiles,onerror=panic,suid,dev=2200007 ...
$
```

The Virtual File System Table

It would be a very time-consuming and error-prone task to manually mount file systems every time you wanted to access them. To avoid this problem, the virtual file system table (the `/etc/vfstab` file) provides a list of file systems and how to mount them.

The `/etc/vfstab` file provides two important features:

- You can specify file systems to automatically mount when the system boots.
- You can mount file systems by using only the mount point name, because the `/etc/vfstab` file contains the mapping between the mount point and the actual device slice name.

A default `/etc/vfstab` file is created when you install a system, depending on the selections you make when installing system software. However, you can edit the `/etc/vfstab` file on a system whenever you want. To add an entry, the main information you need to specify is the device where the file system resides, the name of the mount point, the type of the file system, whether you want the file system to mount automatically when the system boots (by using the `mountall` command), and any mount options.

The following is an example of an `/etc/vfstab` file. Comment lines begin with `#`. This example shows an `/etc/vfstab` file for a system with two disks (`c0t0d0` and `c0t3d0`).

```
$ more /etc/vfstab
#device         device          mount           FS      fsck   mount  mount
#to mount       to fsck         point           type    pass   at boot options
/dev/dsk/c0t0d0s0 /dev/rdsk/c0t0d0s0 /          ufs     1       no      -
/proc           -               /proc           proc    -       no      -
/dev/dsk/c0t0d0s1 -               -             swap    -       no      -
swap            -               /tmp            tmpfs   -       yes     -
/dev/dsk/c0t0d0s6 /dev/rdsk/c0t0d0s6 /usr       ufs     2       no      -
/dev/dsk/c0t3d0s7 /dev/rdsk/c0t3d0s7 /test      ufs     2       yes     -
$
```

In the preceding example, the last entry specifies that a UFS file system on the `/dev/dsk/c0t3d0s7` slice will be automatically mounted on the `/test` mount point when the system boots. Note that, for root (`/`) and `/usr`, the `mount at boot` field value is specified as `no`, because these file systems are mounted by the kernel as part of the boot sequence before the `mountall` command is run.

For descriptions of each of the `/etc/vfstab` fields and information on how to edit and use the file, see Chapter 17.

The NFS Environment

NFS is a distributed file system service that can be used to share *resources* (files or directories) from one system, typically a server, with other systems on the network. For example, you might want to share third-party applications or source files with users on other systems.

NFS makes the actual physical location of the resource irrelevant to the user. Instead of placing copies of commonly used files on every system, NFS allows you to place one copy on one system's disk and let all other systems access it from the network. Under NFS, remote files are virtually indistinguishable from local ones.

A system becomes an NFS server if it has resources to share on the network. A server keeps a list of currently shared resources and their access restrictions (such as read/write or read-only access).

When you share a resource, you make it available for mounting by remote systems.

You can share a resource in these ways:

- By using the `share` or `shareall` command
- By adding an entry to the `/etc/dfs/dfstab` (distributed file system table) file and rebooting the system

For information on how to share resources, see Chapter 17. For a complete description of NFS, see Chapter 14, "Managing Network File Systems (Overview)," in *System Administration Guide: Resource Management and Network Services*.

Automounting or AutoFS

You can mount NFS file system resources by using a client-side service called automounting (or AutoFS), which enables a system to automatically mount and unmount NFS resources whenever you access them. The resource remains mounted as long as you remain in the directory and are using a file. If the resource is not accessed for a certain period of time, it is automatically unmounted.

AutoFS provides the following features:

- NFS resources don't need to be mounted when the system boots, which saves booting time.
- Users don't need to know the root password to mount and unmount NFS resources.
- Network traffic might be reduced, since NFS resources are only mounted when they are in use.

The AutoFS service is initialized by the `automount` utility, which runs automatically when a system is booted. The `automountd` daemon runs continuously and is responsible for the mounting and unmounting of the NFS file systems on an as-needed basis. By default, the `/home` file system is is mounted by the `automount` daemon.

With AutoFS, you can specify multiple servers to provide the same file system. This way, if one of the servers is down, AutoFS can try to mount from another machine.

For complete information on how to set up and administer AutoFS, see *System Administration Guide: IP Services*.

Determining a File System's Type

You can determine a file system's type by using the following:

- The `FS type` field in the virtual file system table (`/etc/vfstab` file)
- The `/etc/default/fs` file for local file systems
- The `/etc/dfs/fstypes` file for NFS file systems

How to Determine a File System's Type

This procedure works whether the file system is mounted or not.

Determine a file system's type by using the `grep` command.

```
$ grep mount-point fs-table
```

mount-point	Specifies the mount point name of the file system for which you want to know the file system type. For example, the `/var` directory.
fs-table	Specifies the absolute path to the file system table in which to search for the file system's type. If the file system is mounted, *fs-table* should be `/etc/mnttab`. If the file system isn't mounted, *fs-table* should be `/etc/vfstab`.

Information for the mount point is displayed.

Note – If you have the raw device name of a disk slice, you can use the `fstyp` command to determine a file system's type (if the disk slice contains a file system). For more information, see `fstyp`(1M).

Examples—Determining a File System's Type

The following example uses the `/etc/vfstab` file to determine the type of the `/export` file system.

```
$ grep /export /etc/vfstab
/dev/dsk/c0t3d0s6   /dev/rdsk/c0t3d0s6  /export ufs   2       yes    -
$
```

The following example uses the `/etc/mnttab` file to determine the file system type of the currently mounted diskette (which was mounted by `vold`).

```
$ grep /floppy /etc/mnttab
/vol/dev/diskette0/unnamed_floppy   /floppy/unnamed_floppy  pcfs rw,
nohidden,nofoldcase,dev=16c0009      89103376
$
```

CHAPTER 16

Creating UFS, TMPFS, and LOFS File Systems (Tasks)

This chapter describes how to create UFS, temporary (TMPFS), and loopback (LOFS) file systems. For UFS file systems, this chapter shows you how to create a file system on a hard disk by using the `newfs` command. Because TMPFS and LOFS are virtual file systems, you actually "access" them by mounting them.

This is a list of the step-by-step instructions in this chapter.

Note – For instructions on how to create UFS and DOS file systems on removable media, see Chapter 1.

Creating a UFS File System

Before you can create a UFS file system on a disk, the disk must be formatted and divided into slices. A disk slice is a physical subset of a disk that is composed of a single range of contiguous blocks. A slice can be used either as a raw device that provides, for example, swap space, or to hold a disk-based file system. See Chapter 10 for complete information on formatting disks and dividing disks into slices.

Volume management products, like Solaris Volume Manager, create more sophisticated *volumes*, that expand beyond single slice or single disk boundaries. For more information about using volumes, see *Solaris Volume Manager Administration Guide*.

Note – Solaris device names use the term slice (and the letter `s` in the device name) to refer to the slice number. Slices are also called "partitions."

You need to create UFS file systems only occasionally, because the Solaris operating system automatically creates them as part of the installation process. You need to create (or re-create) a UFS file system when you want to do the following:

- Add or replace disks
- Change the existing partitioning structure
- Do a full restoration of a file system

The `newfs` command is the standard way to create UFS file systems. The `newfs` command is a convenient front-end to the `mkfs` command, which actually creates the new file system. The `newfs` command reads parameter defaults, such as tracks per cylinder and sectors per track, from the disk label that will contain the new file system. The options you choose are passed to the `mkfs` command to build the file system.

Default Parameters for the `newfs` Command

To make a new file system on a disk slice, you almost always use the `newfs` command. The following table shows the default parameters that are used by the `newfs` command.

Parameter	Default Value
Block size	8 Kbytes
Fragment size	1 Kbyte
Minimum free space	((64 Mbytes/partition size) * 100), rounded down to the nearest integer and limited to between 1% and 10%, inclusively
Rotational delay	Zero
Optimization type	Time
Number of inodes	1 inode for each 2 Kbytes of data space

▼ How to Create a UFS File System

Steps

1. **Make sure you have met the following prerequisites:**

   a. **The disk must be formatted and divided into slices.**

For information on formatting disks and dividing disks into slices, see Chapter 10.

b. **You need to know the device name of the slice that will contain the file system.**

   For information on finding disks and disk slice numbers, see Chapter 11.

c. **If you are re-creating an existing UFS file system, unmount it.**

d. **You must be superuser or assume an equivalent role.**

2. **Create the UFS file system.**

```
# newfs [-N] [-b size] [-i bytes] /dev/rdsk/device-name
```

`-N`	Displays what parameters the `newfs` command would pass to the `mkfs` command without actually creating the file system. This option is a good way to test the `newfs` command.
`-b` *size*	Specifies the block size for the file system, either 4096 or 8192 bytes per block. The default is 8192.
`-i` *bytes*	Specifies the number of bytes per inode. The default varies depending on the disk size. For more information, see `newfs`(1M).
device-name	Specifies the disk device name on which to create the new file system.

The system asks for confirmation.

Caution – Be sure you have specified the correct device name for the slice before performing this step. If you specify the wrong slice, you will erase its contents when the new file system is created. This error might cause the system to panic.

3. **To verify the creation of the UFS file system, check the new file system.**

```
# fsck /dev/rdsk/device-name
```

The *device-name* argument specifies the name of the disk device that contains the new file system.

The `fsck` command checks the consistency of the new file system, reports any problems, and prompts you before it repairs the problems. For more information on the `fsck` command, see Chapter 20 or `fsck`(1M).

Example 16–1 Creating a UFS File System

The following example shows how to create a UFS file system on `/dev/rdsk/c0t1d0s7`.

```
# newfs /dev/rdsk/c0t1d0s7
/dev/rdsk/c0t1d0s7:  725760 sectors in 720 cylinders of 14 tracks, 72 sectors
        354.4MB in 45 cyl groups (16 c/g, 7.88MB/g, 3776 i/g)
super-block backups (for fsck -F ufs -o b=#) at:
 32, 16240, 32448, 48656, 64864, 81072, 97280, 113488, 129696, 145904, 162112,
 178320, 194528, 210736, 226944, 243152, 258080, 274288, 290496, 306704,
 322912, 339120, 355328, 371536, 387744, 403952, 420160, 436368, 452576,
 468784, 484992, 501200, 516128, 532336, 548544, 564752, 580960, 597168,
 613376, 629584, 645792, 662000, 678208, 694416, 710624,
#
```

See Also To mount the UFS file system and make it available, go to Chapter 17.

Creating a Temporary File System (TMPFS)

A temporary file system (TMPFS) uses local memory for file system reads and writes, which is typically much faster than reads and writes in a UFS file system. TMPFS file systems can improve system performance by saving the cost of reading and writing temporary files to a local disk or across the network. Files in TMPFS file systems do not survive across reboots or unmounts.

If you create multiple TMPFS file systems, be aware that they all use the same system resources. Files created under one TMPFS file system use up the space available for any other TMPFS file system, unless you limit TMPFS sizes by using the `-o size` option of the `mount` command.

For more information, see the `tmpfs`(7FS).

▼ How to Create a TMPFS File System

Steps

1. **Become superuser or assume an equivalent role.**
2. **Create the directory that you want to mount as the TMPF file system, if necessary.**

   ```
   # mkdir /mount-point
   ```

   mount-point is the directory on which the TMPFS file system is mounted.
3. **Mount the TMPFS file system.**

   ```
   # mount -F tmpfs [-o size=number]  swap mount-point
   ```

`-o size=`*number*	Specifies the size limit of the TMPFS file system in Mbytes.
mount-point	Specifies the directory on which the TMPFS file system is mounted.

To set up the system to automatically mount a TMPFS file system when it boots, see Example 16–3.

4. **Verify that the TMPFS file system has been created.**

```
# mount -v
```

Example 16–2 Creating a TMPFS File System

The following example shows how to create, mount, and limit the size of the TMPFS file system, `/export/reports`, to 50 Mbytes.

```
# mkdir /export/reports
# chmod 777 /export/reports
# mount -F tmpfs -o size=50m swap /export/reports
```

Example 16–3 Mounting a TMPFS File System at Boot Time

You can set up the system to automatically mount a TMPFS file system when it boots by adding an `/etc/vfstab` entry. The following example shows an entry in the `/etc/vfstab` file that mounts `/export/test` as a TMPFS file system when the system boots. Since the `size=`*number* option is not specified, the size of the TMPFS file system on `/export/test` is limited only by the available system resources.

```
swap - /export/test  tmpfs   -  yes  -
```

For more information on the `/etc/vfstab` file, see "Field Descriptions for the `/etc/vfstab` File" on page 278.

Creating a Loopback File System (LOFS)

A LOFS file system is a virtual file system that provides an alternate path to an existing file system. When other file systems are mounted onto an LOFS file system, the original file system does not change.

For more information, see the `lofs`(7FS).

Caution – Be careful when creating LOFS file systems. Because LOFS file systems are virtual file systems, the potential for confusing both users and applications is enormous.

▼ How to Create an LOFS File System

Steps

1. **Become superuser or assume an equivalent role.**
2. **Create the directory you want to mount as an LOFS file system, if necessary.**

   ```
   # mkdir loopback-directory
   ```

3. **Grant the appropriate permissions and ownership on the newly created directory.**
4. **Create the mount point where you want to mount the LOFS file system, if necessary.**

   ```
   # mkdir /mount-point
   ```

5. **Mount the LOFS file system.**

   ```
   # mount -F lofs loopback-directory /mount-point
   ```

loopback-directory	Specifies the file system to be mounted on the loopback mount point.
/mount-point	Specifies the directory on which to mount the LOFS file system.

6. **Verify that the LOFS file system has been mounted.**

   ```
   # mount -v
   ```

Example 16–4 Creating and Mounting an LOFS File System

The following example illustrates how to create, mount, and test new software in the `/new/dist` directory as a loopback file system without actually having to install it.

```
# mkdir /tmp/newroot
# mount -F lofs /new/dist /tmp/newroot
# chroot /tmp/newroot newcommand
```

Example 16–5 Mounting an LOFS File System at Boot Time

You can set up the system to automatically mount an LOFS file system when it boots by adding an entry to the end of the `/etc/vfstab` file. The following example shows an entry in the `/etc/vfstab` file that mounts an LOFS file system for the root (/) file system on `/tmp/newroot`.

```
/ - /tmp/newroot  lofs   -  yes  -
```

Make sure the loopback entries are the last entries in the `/etc/vfstab` file. Otherwise, if the `/etc/vfstab` entry for a loopback file system precedes the file systems to be included in it, the loopback file system cannot be mounted.

See Also For more information on the `/etc/vfstab` file, see "Field Descriptions for the `/etc/vfstab` File" on page 278.

CHAPTER 17

Mounting and Unmounting File Systems (Tasks)

This chapter describes how to mount and unmount file systems.

This is a list of the step-by-step instructions in this chapter.

Overview of Mounting File Systems

After you create a file system, you need to make it available to the system so you can use it. You make a file system available by mounting it, which attaches the file system to the system directory tree at the specified mount point. The root (`/`) file system is always mounted.

The following table provides guidelines on mounting file systems based on how you use them.

Mount Type Needed	Suggested Mount Method
Local or remote file systems that need to be mounted infrequently	The mount command that you enter manually from the command line.
Local file systems that need to be mounted frequently	The /etc/vfstab file, which mounts the file system automatically when the system is booted in multi-user state.
Remote file systems that need to be mounted frequently, such as home directories	■ The /etc/vfstab file, which automatically mounts the file system when the system is booted in multi-user state. ■ AutoFS, which automatically mounts or unmounts the file system when you access or change out of the directory. To enhance performance, you can also cache the remote file systems by using the CacheFS file system.

You can mount media that contains a file system by inserting the media into the drive and running the `volcheck` command if necessary. For more information on mounting removable media, see Chapter 1.

Commands for Mounting and Unmounting File Systems

The following table lists the commands in the `/usr/sbin` directory that you use to mount and unmount file systems.

TABLE 17–1 Commands for Mounting and Unmounting File Systems

Command	Man Page	Description
`mount`	`mount`(1M)	Mounts file systems and remote resources.
`mountall`	`mountall`(1M)	Mounts all file systems that are specified in the `/etc/vfstab` file. The `mountall` command runs automatically when the system enters multiuser mode.
`umount`	`mount`(1M)	Unmounts file systems and remote resources.
`umountall`	`mountall`(1M)	Unmounts all file systems that are specified in the `/etc/vfstab` file.

The `mount` and `mountall` commands will not mount a read/write file system that has known inconsistencies. If you receive an error message from the `mount` or `mountall` command, you might need to check the file system. See Chapter 20 for information on how to check the file system.

The `umount` and `umountall` commands will not unmount a file system that is busy. A file system is considered busy if one of the following is true:

- A user is accessing a file or directory in the file system.
- If a program has a file open in that file system.
- If the file system is shared.

Commonly Used Mount Options

The following table describes the commonly used options that you can specify with the `mount -o` option. If you specify multiple options, separate them with commas (no spaces). For example, `-o ro,nosuid`.

For a complete list of mount options for each file system type, refer to the specific mount man pages (for example, `mount_ufs`(1M)).

TABLE 17–2 Commonly Used `-o` Mount Options

Option	File System	Description
`bg | fg`	NFS	If the first mount attempt fails, retries in the background (`bg`) or in the foreground (`fg`). This option is safe for non-critical `vfstab` entries. The default is `fg`.
`hard | soft`	NFS	Specifies the procedure if the server does not respond. The `soft` option indicates that an error is returned. The `hard` option indicates that the retry request is continued until the server responds. The default is `hard`.
`intr | nointr`	NFS	Specifies whether keyboard interrupts are delivered to a process that is hung while waiting for a response on a hard-mounted file system. The default is `intr` (interrupts allowed).
`largefiles | nolargefiles`	UFS	Enables you to create files larger than 2 Gbytes. The `largefiles` option means that a file system mounted with this option *might* contain files larger than 2 Gbytes, but it is not required. If the `nolargefiles` option is specified, the file system cannot be mounted on a system that is running Solaris 2.6 or compatible versions. The default is `largefiles`.

TABLE 17–2 Commonly Used `-o` Mount Options *(Continued)*

Option	File System	Description
`logging | nologging`	UFS	Enables or disables logging for the file system. UFS logging is the process of storing transactions (changes that make up a complete UFS operation) into a log before the transactions are applied to the UFS file system. Logging helps prevent UFS file systems from becoming inconsistent, which means `fsck` can be bypassed. Bypassing `fsck` reduces the time to reboot a system if it crashes, or after a system is shutdown uncleanly. The log is allocated from free blocks on the file system, and is sized at approximately 1 Mbyte per 1 Gbyte of file system, up to a maximum of 64 Mbytes. The default is `logging`.
`atime | noatime`	UFS	Suppresses access time updates on files, except when they coincide with updates to the time of the last file status change or the time of the last file modification. For more information, see `stat`(2). This option reduces disk activity on file systems where access times are unimportant (for example, a Usenet news spool). The default is normal access time (`atime`) recording.
`remount`	All	Changes the mount options associated with an already-mounted file system. This option can generally be used with any option except `ro`, but what can be changed with this option is dependent on the file system type.
`retry=`*n*	NFS	Retries the mount operation when it fails. *n* is the number of times to retry.
`ro | rw`	CacheFS, NFS, PCFS, UFS, HSFS	Specifies read/write (rw) or read-only (ro). If you do not specify this option, the default is rw. The default option for HSFS is `ro`.
`suid | nosuid`	CacheFS, HSFS, NFS, UFS	Allows or disallows `setuid` execution. The default is to allow `setuid` execution.

Field Descriptions for the `/etc/vfstab` File

An entry in the `/etc/vfstab` file has seven fields, which are described in the following table.

TABLE 17–3 Field Descriptions for the `/etc/vfstab` File

Field Name	Description
`device to mount`	This field identifies one of the following: ■ The block device name for a local UFS file system (for example, `/dev/dsk/c0t0d0s0`). ■ The resource name for a remote file system (for example, `myserver:/export/home`). For more information about NFS, see *System Administration Guide: IP Services*. ■ The block device name of the slice on which to swap (for example, `/dev/dsk/c0t3d0s1).` ■ A directory for a virtual file system type.
`device to fsck`	The raw (character) device name that corresponds to the UFS file system identified by the `device to mount` field (for example, `/dev/rdsk/c0t0d0s0`). This field determines the raw interface that is used by the `fsck` command. Use a dash (`-`) when there is no applicable device, such as for a read-only file system or a remote file system.
`mount point`	Identifies where to mount the file system (for example, `/usr`).
`FS type`	Identifies the type of file system.
`fsck pass`	The pass number used by the `fsck` command to decide whether to check a file system. When the field contains a dash (`-`), the file system is not checked. When the field contains a zero, UFS file systems are not checked but non-UFS file systems are checked. When the field contains a value greater than zero, the file system is always checked. All file systems with a value of `1` in this field are checked one at a time in the order they appear in the `vfstab` file. When the `fsck` command is run on multiple UFS file systems that have `fsck pass` values greater than one and the preen option (`-o p`) is used, the `fsck` command automatically checks the file systems on different disks in parallel to maximize efficiency. Otherwise, the value of the pass number does not have any effect.
`mount at boot`	Set to `yes` or `no` for whether the file system should be automatically mounted by the `mountall` command when the system is booted. Note that this field has nothing to do with AutoFS. The root (`/`), `/usr` and `/var` file systems are not mounted from the `vfstab` file initially. This field should always be set to `no` for these file systems and for virtual file systems such as `/proc` and `/dev/fd`.
`mount options`	A list of comma-separated options (with no spaces) that are used for mounting the file system. Use a dash (`-`) to indicate no options. For a list of commonly used mount options, see Table 17–2.

Note – You must have an entry in each field in the `/etc/vfstab` file. If there is no value for the field, be sure to enter a dash (`-`). Otherwise, the system might not boot successfully. Similarly, white space should not be used in a field value.

Mounting File Systems

The following sections describe how to mount a file system by adding an entry in the `/etc/vfstab` file or by using the `mount` command from the command line.

How to Determine Which File Systems Are Mounted

You can determine which file systems are already mounted by using the `mount` command.

```
$ mount [ -v ]
```

`-v` Displays the list of mounted file systems in verbose mode.

EXAMPLE 17–1 Determining Which File Systems Are Mounted

This example shows how to use the `mount` command to display information about the file systems that are currently mounted.

```
$ mount
/ on /dev/dsk/c0t0d0s0 read/write/setuid/intr/largefiles/xattr/onerror=...
/devices on /devices read/write/setuid/dev=46c0000 on Thu Jan  ...
/usr on /dev/dsk/c0t0d0s6 read/write/setuid/intr/largefiles/xattr/...
/proc on /proc read/write/setuid/dev=4700000 on Thu Jan  8 ...
/etc/mnttab on mnttab read/write/setuid/dev=47c0000 on Thu Jan  8 ...
/dev/fd on fd read/write/setuid/dev=4800000 on Thu Jan  8 ...
/var/run on swap read/write/setuid/xattr/dev=1 on Thu Jan  8 ...
/tmp on swap read/write/setuid/xattr/dev=2 on Thu Jan  8 ...
/stuff on /dev/dsk/c0t0d0s5 read/write/setuid/intr/largefiles/xattr...
/export/home on /dev/dsk/c0t0d0s7 read/write/setuid/intr/largefiles/...
/home/rimmer on pluto:/export/home/rimmer remote/read/write/setuid/xattr/...
$
```

▼ How to Add an Entry to the `/etc/vfstab` File

Steps

1. **Become superuser or assume an equivalent role.**
2. **Create a mount point for the file system to be mounted, if necessary.**

   ```
   # mkdir /mount-point
   ```

   There must be a mount point on the local system to mount a file system. A mount point is a directory to which the mounted file system is attached.

3. **Edit the `/etc/vfstab` file and add an entry. Make sure that you do the following:**

   a. **Separate each field with white space (a space or a tab).**

   b. **Enter a dash (`-`) if a field has no contents.**

   c. **Save the changes.**

   For detailed information about the `/etc/vfstab` field entries, see Table 17–3.

   Note – Since the root (`/`) file system is mounted read-only by the kernel during the boot process, only the `remount` option (and options that can be used in conjunction with `remount`) affect the root (`/`) entry in the `/etc/vfstab` file.

Example 17–2 Adding an Entry to the `/etc/vfstab` File

The following example shows how to mount the disk slice `/dev/dsk/c0t3d0s7` as a UFS file system to the mount point directory `/files1`. The raw character device `/dev/rdsk/c0t3d0s7` is specified as the `device to fsck`. The `fsck pass` value of `2` means that the file system will be checked, but not sequentially.

```
#device           device             mount    FS       fsck   mount    mount
#to mount         to fsck            point    type     pass   at boot  options
#
/dev/dsk/c0t3d0s7 /dev/rdsk/c0t3d0s7 /files1  ufs      2      yes       -
```

The following example shows how to mount the `/export/man` directory from the system `pluto` as an NFS file system on mount point `/usr/man`. A `device to fsck` nor a `fsck pass` is specified because it's an NFS file system. In this example, `mount options` are `ro` (read-only) and `soft`.

```
#device           device             mount    FS       fsck   mount    mount
#to mount         to fsck            point    type     pass   at boot  options
pluto:/export/man   -                /usr/man nfs      -      yes       ro,soft
```

The following example shows how to mount the root (/) file system on a loopback mount point, /tmp/newroot. LOFS file systems must always be mounted after the file systems that are in the LOFS file system.

```
#device             device              mount    FS       fsck   mount    mount
#to mount           to fsck             point    type     pass   at boot  options
#
/                     -                 /tmp/newroot lofs -      yes       -
```

▼ How to Mount a File System (/etc/vfstab File)

Steps

1. **Become superuser or assume an equivalent role.**

2. **Mount a file system listed in the /etc/vfstab file.**

   ```
   # mount /mount-point
   ```

   /mount-point specifies an entry in the mount point or device to mount field in the /etc/vfstab file. It is usually easier to specify the mount point.

Example 17–3 Mounting a File System (/etc/vfstab File)

The following example shows how to mount the /usr/dist file system that is listed in the /etc/vfstab file.

```
# mount /usr/dist
```

Example 17–4 Mounting All File Systems (/etc/vfstab File)

The following example shows the messages that are displayed if file systems are already mounted when you use the mountall command.

```
# mountall
/dev/rdsk/c0t0d0s7 already mounted
mount: /tmp already mounted
mount: /dev/dsk/c0t0d0s7 is already mounted, /export/home is busy,
        or the allowable number of mount points has been exceeded
```

All the file systems with a device to fsck entry are checked and fixed, if necessary, before they are mounted.

The following example shows how to mount all the local systems that are listed in the /etc/vfstab file.

```
# mountall -l
# mount
/ on /dev/dsk/c0t0d0s0 read/write/setuid/intr/largefiles/xattr/onerror=...
/devices on /devices read/write/setuid/dev=46c0000 on Thu Jan  8 ...
/usr on /dev/dsk/c0t0d0s6 read/write/setuid/intr/largefiles/xattr/...
```

```
/proc on /proc read/write/setuid/dev=4700000 on Thu Jan  8 ...
/etc/mnttab on mnttab read/write/setuid/dev=47c0000 on Thu Jan  8 ...
/dev/fd on fd read/write/setuid/dev=4800000 on Thu Jan  8 09:38:30 2004
/var/run on swap read/write/setuid/xattr/dev=1 on Thu Jan  8 ...
/tmp on swap read/write/setuid/xattr/dev=2 on Thu Jan  8 09:38:30 2004
/stuff on /dev/dsk/c0t0d0s5 read/write/setuid/intr/largefiles/xattr...
/export/home on /dev/dsk/c0t0d0s7 read/write/setuid/intr/largefiles/...
```

The following example shows how to mount all of the remote file systems that are listed in the `/etc/vfstab` file.

```
# mountall -r
# mount
/ on /dev/dsk/c0t0d0s0 read/write/setuid/intr/largefiles/xattr/onerror=...
/devices on /devices read/write/setuid/dev=46c0000 on Thu Jan  8 ...
/usr on /dev/dsk/c0t0d0s6 read/write/setuid/intr/largefiles/xattr/...
/proc on /proc read/write/setuid/dev=4700000 on Thu Jan  8 ...
/etc/mnttab on mnttab read/write/setuid/dev=47c0000 on Thu Jan  8 ...
/dev/fd on fd read/write/setuid/dev=4800000 on Thu Jan  8 ...
/var/run on swap read/write/setuid/xattr/dev=1 on Thu Jan  8 ...
/tmp on swap read/write/setuid/xattr/dev=2 on Thu Jan  8 09:38:30 2004
/stuff on /dev/dsk/c0t0d0s5 read/write/setuid/intr/largefiles/xattr...
/export/home on /dev/dsk/c0t0d0s7 read/write/setuid/intr/largefiles/...
/home/rimmer on pluto:/export/home/rimmer remote/read/write/setuid/xattr/...
```

▼ How to Mount a UFS File System (`mount` Command)

Steps

1. **Become superuser or assume an equivalent role.**
2. **Create a mount point for the file system to be mounted, if necessary.**

   ```
   # mkdir /mount-point
   ```

   There must be a mount point on the local system to mount a file system. A mount point is a directory to which the mounted file system is attached.
3. **Mount the UFS file system.**

   ```
   # mount [-o mount-options] /dev/dsk/device-name /mount-point
   ```

`-o` *mount-options*	Specifies mount options that you can use to mount a UFS file system. For a list of options, see Table 17–2 or `mount_ufs`(1M).
`/dev/dsk/`*device-name*	Specifies the disk device name for the slice that contains the file system (for example, `/dev/dsk/c0t3d0s7`). To get slice information for a disk, see "How to Display Disk Slice Information" on page 185.

/mount-point	Specifies the directory on which to mount the file system.

Example 17–5 Mounting a UFS File System (`mount` Command)

The following example shows how to mount `/dev/dsk/c0t3d0s7` on the `/files1` directory.

```
# mount /dev/dsk/c0t3d0s7 /files1
```

▼ How to Mount a UFS File System Without Large Files (`mount` Command)

When you mount a file system, the `largefiles` option is selected by default, which enables you to create files larger than 2 Gbytes. Once a file system contains large files, you cannot remount the file system with the `nolargefiles` option or mount it on a system that is running Solaris 2.6 or compatible versions, until you remove any large files and run the `fsck` command to reset the state to `nolargefiles`.

This procedure assumes that the file system is in the `/etc/vfstab` file.

Steps

1. **Become superuser or assume an equivalent role.**
2. **Create a mount point for the file system to be mounted, if necessary.**

   ```
   # mkdir /mount-point
   ```

   There must be a mount point on the local system to mount a file system. A mount point is a directory to which the mounted file system is attached.
3. **Make sure there are no large files in the file system.**

   ```
   # cd /mount-point
   # find . -xdev -size +20000000 -exec ls -l {} \;
   ```

   /mount-point identifies the mount point of the file system you want to check for large files.
4. **Remove or move any large files in this file system to another file system, if necessary.**
5. **Unmount the file system.**

   ```
   # umount /mount-point
   ```
6. **Reset the file system state.**

   ```
   # fsck /mount-point
   ```

7. **Remount the file system with the `nolargefiles` option.**

   ```
   # mount -o nolargefiles /mount-point
   ```

Example 17–6 Mounting a File System Without Large Files (`mount` Command)

The following example shows how to check the `/datab` file system and remount it with the `nolargefiles` option.

```
# cd /datab
# find . -xdev -size +20000000 -exec ls -l {} \;
# umount /datab
# fsck /datab
# mount -o nolargefiles /datab
```

▼ How to Mount an NFS File System (`mount` Command)

Steps

1. **Become superuser or assume an equivalent role.**
2. **Create a mount point for the file system to be mounted, if necessary.**

   ```
   # mkdir /mount-point
   ```

   There must be a mount point on the local system to mount a file system. A mount point is a directory to which the mounted file system is attached.
3. **Make sure the resource (file or directory) is available from a server.**

   To mount an NFS file system, the resource must be made available on the server by using the `share` command. For information on how to share resources, see "About the NFS Service" in *System Administration Guide: Resource Management and Network Services*.
4. **Mount the NFS file system.**

   ```
   # mount -F nfs [-o mount-options] server:/directory /mount-point
   ```

`-o` *mount-options*	Specifies `mount` options that you can use to mount an NFS file system. See Table 17–2 for the list of commonly used `mount` options or `mount_nfs`(1M) for a complete list of options.
server:/directory	Specifies the server's host name that contains the shared resource, and the path to the file or directory to mount.
/mount-point	Specifies the directory on which to mount the file system.

Example 17–7 Mounting an NFS File System (`mount` Command)

The following example shows how to mount the `/export/packages` directory on `/mnt` from the server `pluto`.

```
# mount -F nfs pluto:/export/packages /mnt
```

▼ x86: How to Mount a PCFS (DOS) File System From a Hard Disk (`mount` Command)

Use the following procedure to mount a PCFS (DOS) file system from a hard disk.

Steps

1. **Become superuser or assume an equivalent role.**
2. **Create a mount point for the file system to be mounted, if necessary.**

   ```
   # mkdir /mount-point
   ```

   There must be a mount point on the local system to mount a file system. A mount point is a directory to which the mounted file system is attached.
3. **Mount the PCFS file system.**

   ```
   # mount -F pcfs [-o rw | ro] /dev/dsk/device-name:logical-drive /mount-point
   ```

`-o rw | ro`	Specifies that you can mount a PCFS file system read/write (rw) or read-only (ro). If you do not specify this option, the default is rw.
`/dev/dsk/`*device-name*	Specifies the device name of the whole disk (for example, `/dev/dsk/c0t0d0p0`).
logical-drive	Specifies either the DOS logical drive letter (c through z) or a drive number (1 through 24). Drive c is equivalent to drive 1 and represents the Primary DOS slice on the drive. All other letters or numbers represent DOS logical drives within the Extended DOS slice.
/mount-point	Specifies the directory on which to mount the file system.

   Note that the *device-name* and *logical-drive* must be separated by a colon.

Example 17–8 x86: Mounting a PCFS (DOS) File System From a Hard Disk (`mount` Command)

The following example shows how to mount the logical drive in the primary DOS slice on the `/pcfs/c` directory.

```
# mount -F pcfs /dev/dsk/c0t0d0p0:c /pcfs/c
```

The following example shows how to mount the first logical drive in the extended DOS slice read-only on `/mnt`.

```
# mount -F pcfs -o ro /dev/dsk/c0t0d0p0:2 /mnt
```

Unmounting File Systems

The unmounting of a file system removes it from the file system mount point, and deletes the entry from the `/etc/mnttab` file. Some file system administration tasks cannot be performed on mounted file systems. You should unmount a file system when the following occurs:

- The file system is no longer needed or has been replaced by a file system that contains more current software.
- You need to check and repair the file system by using the `fsck` command. For more information about the `fsck` command, see Chapter 20.

  It is a good idea to unmount a file system before doing a complete backup. For more information about doing backups, see Chapter 23.

Note – File systems are automatically unmounted as part of the system shutdown procedure.

You can use the `umount -f` option to forcibly unmount a file system that is busy in an emergency situation. This practice is not recommended under normal circumstances because the unmounting of a file system with open files could cause a loss of data. This option is only available for UFS and NFS file systems.

Prerequisites for Unmounting File Systems

The prerequisites for unmounting file systems include the following:

- You must be superuser or assume an equivalent role.

- A file system must be available for unmounting. You cannot unmount a file system that is busy. A file system is considered busy if a user is accessing a directory in the file system, if a program has a file open in that file system, or if it is being shared. You can make a file system available for unmounting by doing the following:
  - Changing to a directory in a different file system.
  - Logging out of the system.
  - Using the `fuser` command to list all processes that are accessing the file system and to stop them if necessary. For more details, see "How to Stop All Processes Accessing a File System" on page 288.

    Notify users if you need to unmount a file system that they are using.
  - Unsharing the file system. For information about unsharing a file system, see `unshare`(1M).

How to Verify a File System is Unmounted

To verify that you unmounted a file system or a number of file systems, examine the output from the `mount` command.

```
$ mount | grep unmounted-file-system
$
```

▼ How to Stop All Processes Accessing a File System

Steps

1. **Become superuser or assume an equivalent role.**
2. **List all the processes that are accessing the file system so that you know which processes you are going to stop.**

   ```
   # fuser -c [ -u ] /mount-point
   ```

`-c`	Reports on files that are mount points for file systems and any files within those mounted file systems.
`-u`	Displays the user login name for each process ID.
/mount-point	Specifies the name of the file system for which you want to stop processes.

3. **Stop all processes that are accessing the file system.**

   ```
   # fuser -c -k /mount-point
   ```

   A `SIGKILL` is sent to each process that is using the file system.

Note – You should not stop a user's processes without first warning the user.

4. **Verify that there are no processes that are accessing the file system.**

   ```
   # fuser -c /mount-point
   ```

Example 17–9 Stopping All Processes Accessing a File System

The following example shows how to stop process `4006c` that is using the `/export/home` file system.

```
# fuser -c /export/home
/export/home:     4006c
# fuser -c -k /export/home
/export/home:     4006c
# fuser -c /export/home
/export/home:
```

▼ How to Unmount a File System

Use the following procedure to unmount a file system, except for the root (`/`), `/usr`, or `/var` file systems.

Note – The root (`/`), `/usr`, and `/var` file systems can be unmounted only during a shutdown, since the system needs these file systems to function.

Steps

1. **Make sure that you have met the prerequisites listed in "Prerequisites for Unmounting File Systems" on page 287.**
2. **Unmount the file system.**

   ```
   # umount /mount-point
   ```

   /mount-point is the name of the file system that you want to unmount. This can be one of the following:

   - The directory name where the file system is mounted
   - The device name path of the file system
   - The resource for an NFS file system
   - The loopback directory for a LOFS file system

Example 17–10 Unmounting a File System

The following example shows how to unmount a local home file system.

```
# umount /export/home
```

The following example shows how to unmount the file system on slice 7.

```
# umount /dev/dsk/c0t0d0s7
```

The following example shows how to forcibly unmount the `/export` file system.

```
# umount -f /export
#
```

The following example shows how to unmount all file systems in the `/etc/vfstab` file, except for the root (`/`), `/proc`, `/var`, and `/usr` file systems.

```
# umountall
```

All file systems are unmounted, except for those file systems that are busy.

CHAPTER 18

Using The CacheFS File System (Tasks)

This chapter describes how to set up and maintain CacheFS™ file systems.

This is a list of task maps in this chapter.

For information on troubleshooting CacheFS errors, see "Troubleshooting `cachefspack` Errors" on page 310.

High-Level View of Using the CacheFS File System (Task Map)

Use this task map to identify all the tasks for using CacheFS file systems. Each task in this map points to a series of additional tasks such as creating and mounting the CacheFS file systems, and packing and maintaining the cache.

Task	Description	For Instructions
1. Create and mount a CacheFS file system	Create the cache and mount the file system in the cache.	"Creating and Mounting a CacheFS File System (Task Map)" on page 294

Task	Description	For Instructions
2. Maintain a CacheFS file system	Display and modify a CacheFS file system by unmounting, removing, or re-creating the cache.	"Maintaining a CacheFS File System (Task Map)" on page 299
3. (Optional) Pack and unpack a CacheFS file system	Determine whether you want to pack the cache and use packing lists. Packing the cache ensures that certain files and directories are always updated in the cache.	"Packing a Cached File System (Task Map)" on page 305
4. Collect CacheFS statistics	Determine cache performance and appropriate cache size.	"Collecting CacheFS Statistics (Task Map)" on page 314

Overview of the CacheFS File System

The CacheFS file system is a general purpose caching mechanism that improves NFS server performance and scalability by reducing server and network load. Designed as a layered file system, the CacheFS file system provides the ability to cache one file system on another. In an NFS environment, the CacheFS file system increases the client per server ratio, reduces server and network loads, and improves performance for clients on slow links, such as Point-to-Point Protocol (PPP).

How a CacheFS File System Works

You create a CacheFS file system on a client system so that file systems you cache can be accessed by the client locally instead of across the network. The following figure shows the relationship of the components that are involved in using CacheFS file system.

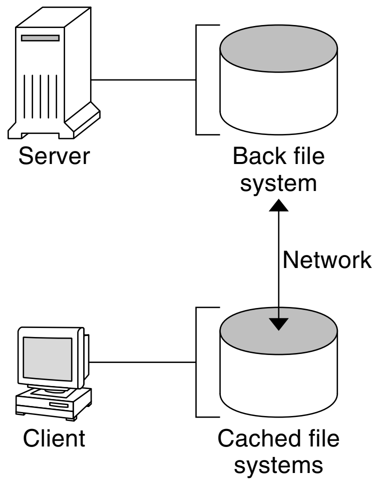

FIGURE 18–1 How a CacheFS File System Works

The *back* file system is the file system that you specify to be mounted in the cache, which can be either NFS or HSFS (High Sierra File System). When the user attempts to access files that are part of the back file system, those files are placed in the cache. The *front* file system is the file system that is mounted in the cache and is accessed from the local mount point. The front file system type must be UFS.

To the user, the initial request to access a file in a CacheFS file system might seem slow, but subsequent uses of the same file are faster.

CacheFS File System Structure and Behavior

Each cache has a set of parameters that determines the cache structure and how it behaves. The parameters are set to default values which are listed in the following table. The default values specify that the entire front file system is used for caching, which is the recommended method of caching file systems.

TABLE 18–1 CacheFS File System Parameters and Their Default Values

CacheFS File System Parameter	Default Value	Definition
`maxblocks`	90%	Sets the maximum number of blocks that a CacheFS file system is allowed to claim within the front file system.

TABLE 18–1 CacheFS File System Parameters and Their Default Values *(Continued)*

CacheFS File System Parameter	Default Value	Definition
`minblocks`	0%	Sets the minimum number of blocks that a CacheFS file system is allowed to claim within the front file system.
`threshblocks`	85%	Sets the number of blocks that must be available in the front file system before a CacheFS file system can claim more than the blocks specified by `minblocks`.
`maxfiles`	90%	Sets the maximum number of available inodes (number of files) that a CacheFS file system is allowed to claim within the front file system.
`minfiles`	0%	Sets the minimum number of available inodes that a CacheFS file system is allowed to claim within the front file system.
`threshfiles`	85%	Sets the number of inodes that must be available in the front file system before a CacheFS file system can claim more than the files specified in `minfiles`.

Typically, you should not change any of these parameter values. They are set to default values to achieve optimal cache behavior. However, you might want to modify the `maxblocks` and `maxfiles` values if you have some room in the front file system that is not used by the cache, and you want to use it for some other file system. You do so by using the `cfsadmin` command. For example:

```
$ cfsadmin -o maxblocks=60
```

Creating and Mounting a CacheFS File System (Task Map)

Use the procedures in this table to create and mount a CacheFS file system.

Task	Description	For Instructions
1. Share the file system to be cached	Verify that the file system you want to cache is shared.	`share`(1M)

Task	Description	For Instructions
2. Create the cache	Use the `cfsadmin` command to create the cache.	"How to Create the Cache" on page 295
3. Mount a file system in the cache	Mount a file system in a cache by using one of the following methods:	
	Mount a CacheFS file system by using the `mount` command.	"How to Mount a CacheFS File System (`mount`)" on page 296
	Mount a CacheFS file system by editing the `/etc/vfstab` file.	"How to Mount a CacheFS File System (`/etc/vfstab`)" on page 298
	Mount a cached a file system by using AutoFS.	"How to Mount a CacheFS File System (AutoFS)" on page 299

▼ How to Create the Cache

Steps

1. **Become superuser on the client system.**

2. **Create the cache.**

   ```
   # cfsadmin -c /cache-directory
   ```

   cache-directory indicates the name of the directory where the cache resides.

   For more information, see `cfsadmin`(1M).

   Note – After you have created the cache, do not perform any operations within the cache directory itself. Doing so could cause conflicts within the CacheFS software.

Example 18–1 Creating the Cache

The following example shows how to create a cache in the `/local/mycache` directory by using the default cache parameter values.

```
# mkdir /local
# cfsadmin -c /local/mycache
```

Mounting a File System in the Cache

You specify a file system to be mounted in the cache so that users can locally access files in that file system. The files do not actually get placed in the cache until the user accesses the files.

The following table describes three ways to mount a CacheFS file system.

Mount Type for CacheFS File System	Frequency of CacheFS Mount Type
Using the `mount` command	Every time the system reboots in order to access the same file system.
Editing the `/etc/vfstab` file	Only once. The `/etc/vfstab` file remains unchanged after the system reboots.
Using AutoFS	Only once. AutoFS maps remain unchanged after the system reboots.

Choose the method of mounting file systems that best suits your environment.

You can mount only file systems that are shared. For information on sharing file systems, see `share`(1M).

Note – The caching of the root (`/`) and `/usr` file systems is not supported in a CacheFS file system.

▼ How to Mount a CacheFS File System (`mount`)

Steps

1. **Become superuser on the client system.**

2. **Create the mount point, if necessary.**

   ```
   # mkdir /mount-point
   ```

   You can create the mount point from anywhere but it must be a UFS file system. The CacheFS options used with the `mount` command, as shown in the next step, determine that the mount point you create is cached in the cache directory you specify.

3. **Mount a file system in the cache.**

   ```
   # mount -F cachefs -o backfstype=fstype,cachedir=/cache-directory [,options]
   /back-filesystem  /mount-point
   ```

fstype	Indicates the file system type of the back file system, which can be either `NFS` or `HSFS`.
/cache-directory	Indicates the name of the UFS directory where the cache resides. This name is the same name you specified when you created the cache in "How to Create the Cache" on page 295.

options	Specifies other mount options that you can include when you mount a file system in a cache. For a list of CacheFS `mount` options, see `mount_cachefs`(1M).
/back-filesystem	Specifies the mount point of the back file system to cache. If the back file system is an NFS file system, you must specify the host name of the server from which you are mounting the file system and the name of the file system to cache, separated by a colon. For example, *merlin: /data/abc*.
/mount-point	Indicates the directory where the file system is mounted.

4. **Verify that the cache you created was actually mounted.**

   ```
   # cachefsstat /mount-point
   ```

   The */mount-point* is the CacheFS file system that you created.

   For example:

   ```
   # cachefsstat /docs
   /docs
                cache hit rate:   100% (0 hits, 0 misses)
           consistency checks:      1 (1 pass, 0 fail)
                     modifies:      0
           garbage collection:      0
   ```

   If the file system was not mounted in the cache, you see an error message similar to the following:

   ```
   # cachefsstat /mount-point
   cachefsstat: mount-point: not a cachefs mountpoint
   ```

   For more information about the `cachefsstat` command, see "Collecting CacheFS Statistics" on page 314.

Example 18–2 Mounting a CacheFS File System (`mount`)

The following example shows how to mount the NFS file system `merlin:/docs` as a CacheFS file system named `/docs` in the cache named `/local/mycache`.

```
# mkdir /docs
# mount -F cachefs -o backfstype=nfs,cachedir=/local/mycache merlin:/docs /docs
```

The following example shows how to make a Solaris 9 SPARC CD (HSFS file system) available as a CacheFS file system named `/cfssrc`. Because you cannot write to the CD, the `ro` argument is specified to make the CacheFS file system read-only. This example assumes that `vold` is not running.

```
# mount -F hsfs -o ro /dev/dsk/c0t6d0s0 /sol9
# mount -F cachefs -o backfstype=hsfs,cachedir=/cfs/cache,ro,noconst,
backpath=/sol9 /dev/dsk/c0t6d0s0 /cfssrc
# ls /cfssrc
Copyright  Solaris_9
```

The following example shows how to mount a Solaris 9 SPARC CD as a CacheFS file system with `vold` running.

```
# mount -F cachefs -o backfstype=hsfs,cachedir=/cfs/cache,ro,noconst,
backpath=/cdrom/sol_9_sparc/s0 /vol/dev/dsk/c0t2d0/sol_9_sparc/s0 /cfssrc
```

The following example shows how to mount a CD as a CacheFS file system with `vold` running.

```
# mount -F cachefs -o backfstype=hsfs,cachedir=/cfs/cache,ro,noconst,
backpath=/cdrom/epson /vol/dev/dsk/c0t2d0/epson /drvrs
```

The following example uses the `demandconst` option to specify consistency checking on demand for the NFS CacheFS file system `/docs`, whose back file system is `merlin:/docs`. For more information, see "Consistency Checking of a CacheFS File System" on page 302.

```
# mount -F cachefs -o backfstype=nfs,cachedir=/local/mycache,demandconst merlin:/docs /docs
```

▼ How to Mount a CacheFS File System (`/etc/vfstab`)

Steps

1. **Become superuser on the client system.**
2. **Using an editor, specify the file systems to be mounted in the `/etc/vfstab` file.**

   See the example that follows.

   For more information on the `/etc/vfstab` file, see "Field Descriptions for the `/etc/vfstab` File" on page 278.
3. **Mount the CacheFS file system.**

   ```
   # mount /mount-point
   ```

   Or, reboot the system.

Example 18–3 Mounting a CacheFS File System (`/etc/vfstab`)

The following example shows the `/etc/vfstab` entry for the `/data/abc` directory from remote system `starbug` that is mounted in the cached directory, `/opt/cache`.

```
#device           device             mount      FS      fsck  mount   mount
#to mount         to fsck            point      type    pass  at boot options
#
starbug:/data/abc /local/abc         /opt/cache cachefs 7     yes     local-access,bg,
nosuid,demandconst,backfstype=nfs,cachedir=/opt/cache
```

▼ How to Mount a CacheFS File System (AutoFS)

You can mount a file system in a cache with AutoFS by specifying the `-fstype=cachefs` mount option in your automount map. Note that the CacheFS mount options, for example, `backfstype` and `cachedir`, are also specified in the automount map.

For details on automount maps, see "Task Overview for Autofs Administration" in *System Administration Guide: Resource Management and Network Services* or`automount`(1M).

Steps

1. **Become superuser on the client system.**
2. **Using an editor, add the following line to the `auto_direct` map:**

   ```
   /mount-point -fstype=cachefs,cachedir=/directory,backfstype=nfs
   server:/file-system
   ```

3. **Using an editor, add the following line to the `auto_master` map:**

   ```
   /-
   ```

   The `/-` entry is a pointer to check the `auto_direct` map.

4. **Reboot the system.**
5. **Verify that the entry was made correctly by changing to the file system you mounted in the cache, and then list the contents, as follows:**

   ```
   # cd /filesystem
   # ls
   ```

Example 18–4 Mounting a CacheFS File System (AutoFS)

The following `auto_direct` entry automatically mounts the CacheFS file system in the `/docs` directory.

```
/docs -fstype=cachefs,cachedir=/local/mycache,backfstype=nfs merlin:/docs
```

Maintaining a CacheFS File System (Task Map)

After a CacheFS file system is set up, it requires little maintenance. Use the optional procedures in this table if you need to perform maintenance tasks on your CacheFS file systems.

Task	Description	For Instructions
1. Modify a CacheFS file system	Modify CacheFS file system behavior by unmounting, deleting, or re-creating the cache.	"Modifying a CacheFS File System" on page 300
2. Display CacheFS file system information	Display information about CacheFS file systems by using the `cfsadmin` command.	"How to Display Information About a CacheFS File System" on page 301
3. Perform consistency checking	Perform consistency checking on demand by using the `cfsadmin` command.	"How to Specify Cache Consistency Checking on Demand" on page 302
4. Delete a CacheFS file system	Delete a CacheFS file system by using the `umount` command and the `cfsadmin` command.	"How to Delete a CacheFS File System" on page 302
5. Check the integrity of a CacheFS file system	Check the integrity of a CacheFS file system by using the `fsck_cachefs` command.	"How to Check the Integrity of a CacheFS File System" on page 304

Maintaining a CacheFS File System

This section describes how to maintain a CacheFS file system.

If you are using the `/etc/vfstab` file to mount file systems, you modify the cache by editing the file system options in the `/etc/vfstab` file. If you are using AutoFS, you modify the cache by editing the file system options in the AutoFS maps.

Modifying a CacheFS File System

When you modify a file system in the cache, you need to delete the cache and then re-create it. You might also need to reboot your machine in single-user mode, depending on how your file systems are shared and accessed.

In the following example, the cache is deleted, re-created, and then mounted again with the `demandconst` option specified for the `/docs` file system.

```
# shutdown -g30 -y
.
.
.
```

```
Type Cntrl-d to proceed with normal startup,
(or give root password for system maintenance):
# enter password:
.
.
.
Here is where you might be prompted to run fsck on the
file system where the cache is located.

# fsck /local
# mount /local
# cfsadmin -d all /local/mycache
# cfsadmin -c /local/mycache
# init 6
.
.
.
console login:
password:
# mount -F cachefs -o backfstype=nfs,cachedir=/local/cache1,demandconst
merlin:/docs /docs
#
```

▼ How to Display Information About a CacheFS File System

Steps

1. **Become superuser on the client system.**
2. **Display information about all file systems cached under a specified cache.**

   ```
   # cfsadmin -l /cache-directory
   ```

   /cache-directory is the name of the directory where the cache resides.

Example 18–5 Displaying Information About CacheFS File Systems

The following example shows information about the `/local/mycache` cache directory. In this example, the `/docs` file system is cached in `/local/mycache`. The last line displays the name of the CacheFS file system.

```
# cfsadmin -l /local/mycache
cfsadmin: list cache FS information
   maxblocks     90%
   minblocks      0%
   threshblocks  85%
   maxfiles      90%
   minfiles       0%
   threshfiles   85%
   maxfilesize    3MB
merlin:_docs:_docs
```

```
#
```

Consistency Checking of a CacheFS File System

To ensure that the cached directories and files remain current, the CacheFS software periodically checks the consistency of files stored in the cache. To check consistency, the CacheFS software compares the current modification time to the previous modification time. If the modification times are different, all data and attributes for the directory or file are purged from the cache. And, new data and attributes are retrieved from the back file system.

Consistency Checking on Demand

Consistency checks can be performed only when you explicitly request checks for file systems that are mounted with `-o demandconst` option. If you mount a file system in a cache with this option, then use the `cfsadmin` command with the `-s` option to request a consistency check. By default, consistency checking is performed file by file as the files are accessed. If no files are accessed, no checks are performed. Using the `-o demandconst` option avoids the situation where the network is flooded with consistency checks.

For more information, see `mount_cachefs`(1M).

▼ How to Specify Cache Consistency Checking on Demand

Steps

1. **Become superuser on the client system.**
2. **Mount the file system in the cache and specify cache consistency checking.**

   ```
   # mount -F cachefs -o backfstype=nfs,cachedir=/directory,demandconst
   server:/file-system  /mount-point
   ```

3. **Initiate consistency checking on a specific CacheFS file system.**

   ```
   # cfsadmin -s /mount-point
   ```

▼ How to Delete a CacheFS File System

Steps

1. **Become superuser on the client system.**
2. **Unmount the CacheFS file system.**

   ```
   # umount /mount-point
   ```

/mount-point specifies the CacheFS file system that you want to delete.

3. **Determine the name of the CacheFS file system (cache ID).**

   ```
   # cfsadmin -l /cache-directory
   cfsadmin: list cache FS information
      maxblocks     90%
      minblocks      0%
      threshblocks  85%
      maxfiles      90%
      minfiles       0%
      threshfiles   85%
      maxfilesize    3MB
   cache-ID
   #
   ```

4. **Delete the CacheFS file system from the specified cache.**

   ```
   # cfsadmin -d cache-ID /cache-directory
   ```

cache-ID	Indicates the name of the CacheFS file system, which is the last line of the `cfsadmin -l` output. For more information, see "How to Display Information About a CacheFS File System" on page 301. You can delete all the CacheFS file systems in a particular cache by specifying `all` for *cache-ID*.
/cache-directory	Specifies the directory where the cache resides.

5. **Verify that the file system has been deleted.**

   The cache ID of the file system you just deleted should be missing from the `cfsadmin -l` output.

   ```
   # cfsadmin -l /cache-directory
   cfsadmin: list cache FS information
      maxblocks     90%
      minblocks      0%
      threshblocks  85%
      maxfiles      90%
      minfiles       0%
      threshfiles   85%
      maxfilesize    3MB
   #
   ```

   For more information about the fields that are specified in the command output, refer to `cfsadmin`(1M).

6. **Update the resource counts for the cache by running the `fsck -F cachefs` command.**

   For more information, see "How to Check the Integrity of a CacheFS File System" on page 304.

Example 18–6 Deleting a CacheFS File System

The following example shows how to delete the file systems from the cache.

```
# umount /cfssrc
# cfsadmin -l /cfssrc
# cfsadmin -d _dev_dsk_c0t6d0s0:_cfssrc
# cfsadmin -l
```

▼ How to Check the Integrity of a CacheFS File System

Use the `fsck` command to check the integrity of CacheFS file systems. The CacheFS version of the `fsck` command automatically corrects problems without requiring user interaction. You should not need to run the `fsck` command manually for CacheFS file systems because the `fsck` command is run automatically at boot time or when the file system is mounted. If you want to manually check the integrity, you can use the following procedure.

For more information, see `fsck_cachefs`(1M).

Steps

1. **Become superuser on the client system.**
2. **Check the file systems in the specified cache.**

```
# fsck -F cachefs [-m -o noclean] /cache-directory
```

`-m`	Causes the `fsck` command to check a CacheFS file system without making any repairs.
`-o noclean`	Forces a check on the CacheFS file systems only. Does not make any repairs.
/cache-directory	Indicates the name of the directory where the cache resides.

Example 18–7 Checking the Integrity of CacheFS File Systems

The following example shows how to check the file systems cached in the `/local/mycache` cache.

```
# fsck -F cachefs /local/mycache
#
```

Packing a Cached File System (Task Map)

The following task map describes the procedures that are associated with packing a CacheFS file system. All of these procedures are optional.

Task	Description	For Instructions
Pack files in the cache	Identify files and directories to be loaded in the cache and pack them. Packing ensures that current copies of these files are available in the cache.	"How to Pack Files in the Cache" on page 306
Create a packing list	Create a packing list if you do not want to specify each individual file that you want packed in the cache.	"How to Create a Packing List" on page 308
Pack files in the cache with a packing list	Specify the name of the packing list of the files to be packed in the cache.	"How to Pack Files in the Cache With a Packing List" on page 309
Unpack files or packing lists from the cache	Remove a file from the cache that is no longer needed.	"How to Unpack Files or Packing Lists From the Cache" on page 309
Display packed files information	View information about the files that you've packed, including their packing status.	"How to Display Packed Files Information" on page 307

Packing a CacheFS File System

For general use, the CacheFS software operates automatically after it is set up, without requiring any action from the user. Files are cached on a most recently used basis. With the *packing* feature, you can take a more active role in managing your cache by ensuring that certain files or directories are always updated in the cache.

You can specify files and directories to be loaded in the cache with the `cachefspack` command. This command ensures that current copies of these files are available in the cache.

The *packing list* contains the names of specific files and directories. The packing list can also contain other packing lists. This feature saves you having to specify individual files and directories when you have many items to pack in your cache.

You can print out a brief help summary of all the `cachefspack` options by using the `-h` option as follows:

```
$ cachefspack -h
Must select 1 and only 1 of the following 5 options
-d Display selected filenames
-i Display selected filenames packing status
-p Pack selected filenames
-u Unpack selected filenames
-U Unpack all files in directory 'dir'
-f Specify input file containing rules
-h Print usage information
-r Interpret strings in LIST rules as regular expressions
-s Strip './' from the beginning of a pattern name
-v Verbose option
files - a list of filenames to be packed/unpacked
```

How to Pack Files in the Cache

Pack files in the cache by using the `cachefspack` command.

```
$ cachefspack -p filename
```

`-p`	Specifies that you want the file or files to be packed. This option is also the default.
filename	Specifies the name of the files or directory you want packed in the cache. When you specify a directory, all of its subdirectories are also packed. For more information, see `cachefspack`(1M).

Examples—Packing Files in the Cache

The following example shows the `projects` file being packed in the cache.

```
$ cachefspack -p projects
```

The following example shows three files being packed in the cache.

```
$ cachefspack -p projects updates master_plan
```

The following example shows a directory being packed in the cache.

```
$ cachefspack -p /data/abc/bin
```

How to Display Packed Files Information

Display packed files information by using the `cachefspack -i` command.

```
$ cachefspack -i[v] filename
```

`-i`	Specifies that you want to view information about your packed files.
`-v`	Is the verbose option.
cached-filename-or-directory	Specifies the name of the file or directory for which to display information.

EXAMPLE 18–8 Displaying Packed Files Information

The following example shows that the `doc_file` file is successfully packed.

```
$ cachefspack -i doc_file
cachefspack: file doc_file marked packed YES, packed YES
```

In the following example, the `/data/abc` directory contains the `bin` subdirectory. The `bin` subdirectory has three files: `big`, `medium`, and `small`. Although the `big` and `small` files are specified to be packed, they are not. The `medium` file is successfully packed.

```
$ cd /data/abc
$ cachefspack -i bin
.
.
.
cachefspack: file /bin/big marked packed YES, packed NO
cachefspack: file /bin/medium marked packed YES,
packed YES
cachefspack: file /bin/small marked packed YES,
packed NO
.
.
.
```

If you use the `-iv` options together, you get additional information as to whether the file or directory specified has been flushed from the cache. For example:

```
$ cd /data/bin
FSCACHEPACK-4$ cachefspack -iv bin
.
.
.
cachefspack: file /bin/big marked packed YES, packed NO,
nocache YES
cachefspack: file /bin/medium marked packed YES,
packed YES, nocache NO
cachefspack: file /bin/small marked packed YES,
packed NO
```

EXAMPLE 18–8 Displaying Packed Files Information *(Continued)*

```
nocache NO
.
.
.
```

The last line of the preceding example shows that the directory contents have not been flushed from the cache.

Using Packing Lists

One feature of the `cachefspack` command is the ability to create packing lists.

A packing list contains files or directories to be packed in the cache. If a directory is in the packing list, all of its subdirectories and files will also be packed.

This feature saves the time of having to specify each individual file that you want packed in the cache.

How to Create a Packing List

To create a packing list, open a file by using `vi` or the editor of your choice. The packing list file format uses the same format as the `filesync` command. For more information, see `filesync`(1).

Two packing list features are the following:

- You can identify files in the packing list as regular expressions rather than literal file names so that you don't have to specify each individual file name.
- You can pack files from a shared directory by ensuring that you pack only those files that you own.

For more information on using these features, see `cachefspack`(1M).

EXAMPLE 18–9 Creating a Packing List

The following example shows the contents of a packing list file.

```
BASE /home/ignatz
LIST plans
LIST docs
IGNORE *.ps
```

- The path identified with the `BASE` statement is the directory where you have items you want to pack.

EXAMPLE 18–9 Creating a Packing List *(Continued)*

- The two `LIST` statements identify specific files within that directory to pack.
- The `IGNORE` statement identifies the file type of `.ps`, which you do not want to pack.

How to Pack Files in the Cache With a Packing List

Pack files in the packing list by using the `cachefspack -f` command, as follows:

```
$ cachefspack -f packing-list
```

`-f`	Specifies that you want to use a packing list.
packing-list	Specifies the name of the packing list.

EXAMPLE 18–10 Packing Files in the Cache With a Packing List

This example uses the `list.pkg` file as the packing list for the `cachefspack` command.

```
$ cachefspack -f list.pkg
```

Unpacking Files or Packing Lists From the Cache

You might need to remove, or unpack, a file from the cache. Perhaps you have some files or directories that have a higher priority than others, so you need to unpack the less critical files. For example, you finished up a project and have archived the files that are associated with that project. You are now working on a new project, and therefore, a new set of files.

How to Unpack Files or Packing Lists From the Cache

Unpack files or packing lists from the cache by using the `-u` or `-U` option of the `cachefspack` command.

```
$ cachefspack -u filename | -U cache-directory
```

`-u`	Specifies that you want the file or files unpacked. You must specify a filename with this option.
filename	Specifies the name of the file or packing list that you want unpacked in the cache.

`-U` Specifies that you want to unpack all files in the cache.

For more information about the `cachefspack` command, see the man page.

EXAMPLE 18–11 Unpacking Files or Packing Lists From the Cache

The following example shows the file `/data/abc/bin/big` being unpacked from the cache.

```
$ cachefspack -u /data/abc/bin/big
```

The following example shows several files being unpacked from the cache.

```
$ cd /data/abc/bin/big
$ cachefspack -u big small medium
```

The following example shows how to unpack a packing list, which is a file that contains the path to a directory of files, as follows:

```
$ cachefspack -uf list.pkg
```

The following example uses the `-U` option to specify that all files in a cache directory being unpacked.

```
$ cachefspack -U /local/mycache
```

You cannot unpack a cache that does not have at least one file system mounted. With the `-U` option, if you specify a cache that does not contain mounted file systems, you see output similar to the following:

```
$ cachefspack -U /local/mycache
cachefspack: Could not unpack cache /local/mycache, no mounted
filesystems in the cache.
```

Troubleshooting `cachefspack` Errors

You might see the following error messages when you use the `cachefspack` command.

`cachefspack:` *pathname* `- can't open directory: permission denied`

Cause
: You might not have the correct permissions to access the file or directory.

Action
: Set the correct permissions.

`cachefspack:` *pathname* `- can't open directory: no such file or directory`

Cause
: You might not have the correct file or directory.

Action
: Check for a possible typo.

`cachefspack:` *pathname* `- can't open directory: stale NFS file handle`

Cause
: The file or directory might have been moved or deleted from the server at the time you attempted to access it.

Action
: Verify that the file or directory on the server is still accessible.

`cachefspack:` *pathname* `- can't open directory: interrupted system call`

Cause
: You might have inadvertently pressed Control-C while issuing the command.

Action
: Reissue the command.

`cachefspack:` *pathname* `- can't open directory: I/O error`

Cause
: You might have a hardware problem.

Action
: Check your hardware connections.

`cachefspack: error opening dir`

Cause
: You might not have the correct file or directory. The path identified after the `BASE` command in the file format could be a file and not a directory. The path specified must be a directory.

Action
: Check for a possible typo. Check the path identified after the `BASE` command in your file format. Make sure the path identifies a directory, not a file.

`cachefspack: unable to get shared objects`

Cause
: The executable might be corrupt or in a format that is not recognizable.

Action
: Replace the executable.

`cachefspack:` *filename* `- can't pack file: permission denied`

Cause
: You might not have the correct permissions to access the file or directory.

Action
: Set the correct permissions.

`cachefspack:` *filename* `- can't pack file: no such file or directory`

Cause
: You might not have the correct file or directory.

Action
: Check for a possible typo.

`cachefspack:` *filename*`- can't pack file: stale NFS file handle`

Cause
: The file or directory might have been moved or deleted from the server at the time you attempted to access it.

Action
: Verify that the file or directory on the server is still accessible.

`cachefspack:` *filename*`- can't pack file: interrupted system call`

Cause
: You might have inadvertently pressed Control-C while issuing the command.

Action
: Reissue the command.

`cachefspack:` *filename*`- can't pack file: I/O error`

Cause
: You might have a hardware problem.

Action
: Check your hardware connections.

`cachefspack:` *filename*`- can't pack file: no space left on device.`

Cause
: The cache is out of disk space.

Action
: You need to increase the size of the cache by increasing disk space.

`cachefspack:` *filename* `- can't unpack file: permission denied`

Cause
: You might not have the correct permissions to access the file or directory.

Action
: Set the correct permissions.

`cachefspack:` *filename* `- can't unpack file: no such file or directory`

Cause
: You might not have the correct file or directory.

Action
: Check for a possible typo.

`cachefspack:` *filename*`- can't unpack file: stale NFS file handle`

Cause
: The file or directory might have been moved or deleted from the server at the time you attempted to access it.

Action
: Verify that the file or directory on the server is still accessible.

`cachefspack:` *filename* `- can't unpack file: interrupted system call`

Cause
: You might have pressed Control-C inadvertently while issuing the command.

Action
: Reissue the command.

`cachefspack:` *filename*`- can't unpack file I/O error`

Cause
: You might have a hardware problem.

Action
: Check your hardware connections.

`cachefspack: only one 'd', 'i', 'p', or 'u' option allowed`

Cause
: You entered more than one of these options in a command session.

Action
: Select one option for the command session.

`cachefspack: can't find environment variable.`

Cause
: You forgot to set a corresponding environment variable to match the `$` in your configuration file.

Action
: Define the environment variable in the proper location.

`cachefspack: skipping LIST command - no active base`

Cause
: A `LIST` command is present in your configuration file that has no corresponding `BASE` command.

Action
: Define the `BASE` command.

Collecting CacheFS Statistics (Task Map)

The following task map shows the steps involved in collecting CacheFS statistics. All the procedures in this table are optional.

Task	Description	For Instructions
Set up logging	Set up logging on a CacheFS file system using the `cachefslog` command.	"How to Set Up CacheFS Logging" on page 316
Locate the log file	Locate the log file with the `cachefslog` command.	"How to Locate the CacheFS Log File" on page 316
Stop logging	Stop logging with the `cachefslog` command.	"How to Stop CacheFS Logging" on page 317
View the cache size	View the cache size by using the `cachefswssize` command.	"How to View the Working Set (Cache) Size" on page 317
View the cache statistics	View the statistics by using the `cachefsstat` command.	"How to View CacheFS Statistics" on page 318

Collecting CacheFS Statistics

Collecting CacheFS statistics enable you to do the following:

- Determine an appropriate cache size
- Observe the performance of the cache

These statistics will help you determine the trade-off between your cache size and the desired performance of the cache.

The CacheFS statistics commands consist of the following:

Command	Man Page	Description
`cachefslog`	`cachefslog`(1M)	Specifies the location of the log file. This command also displays where the statistics are currently being logged, and enables you to stop logging.
`cachefswssize`	`cachefswssize`(1M)	Interprets the log file to give a recommended cache size.
`cachefsstat`	`cachefsstat`(1M)	Displays statistical information about a specific file system or all CacheFS file systems. The information provided in the output of this command is taken directly from the cache.

Note – You can issue the CacheFS statistics commands from any directory. You must be superuser to issue the `cachefswssize` command.

The CacheFS statistics begin accumulating when you create the log file. When the work session is over, stop the logging by using the `cachefslog -h` command, as described in "How to Stop CacheFS Logging" on page 317.

Before using the CacheFS statistics commands, you must do the following:

- Set up your cache by using the `cfsadmin` command.
- Decide on an appropriate length of time to allow statistical information to collect in the log file you create. The length of time should equal a typical work session. For example, a day, a week, or a month.
- Select a location or path for the log file. Make sure that there is enough space to allow for the growth of the log file. The longer you intend to allow statistical information to collect in the log file, the more space you need.

Note – The following procedures are presented in a recommended order. The order is not required.

▼ How to Set Up CacheFS Logging

Steps

1. **Set up logging.**

   ```
   $ cachefslog -f log-file-path /mount-point
   ```

`-f`	Sets up logging.
log-file-path	Specifies the location of the log file. The log file is a standard file you create with an editor, such as vi.
/mount-point	Designates the mount point (CacheFS file system) for which statistics are being collected.

2. **Verify that you correctly set up the log file.**

   ```
   $ cachefslog /mount-point
   ```

Example 18–12 Setting Up CacheFS Logging

The following example shows how to set up the `/var/tmp/samlog` log file to collect statistics about the `/home/sam` directory.

```
$ cachefslog -f /var/tmp/samlog /home/sam
  /var/tmp/samlog: /home/sam
```

How to Locate the CacheFS Log File

You can also use the `cachefslog` command with no options to locate a log file for a particular mount point.

```
$ cachefslog /mount-point
```

/mount-point specifies the CacheFS file system for which you want to view the statistics.

The following example shows what you would see if a log file has been set up. The location of the log file is `/var/tmp/stufflog`.

```
$ cachefslog /home/stuff
     /var/tmp/stufflog: /home/stuff
```

The following example shows that no log file has been set up for the specified file system.

```
$ cachefslog /home/zap
    not logged: /home/zap
```

How to Stop CacheFS Logging

Use the `cachefslog -h` option to stop logging.

```
$ cachefslog -h /mount-point
```

The following example shows how to stop logging on `/home/stuff`.

```
$ cachefslog -h /home/stuff
not logged: /home/stuff
```

If you get a system response other than the one specified here, you did not successfully stop logging. Check to see if you are using the correct log file name and mount point.

▼ How to View the Working Set (Cache) Size

You might want to check if you need to increase the size of the cache. Or, you might want to determine what the ideal cache size is based on your activity since you last used the `cachefslog` command for a particular mount point.

Steps

1. **Become superuser on the client system.**
2. **View the current cache size and highest logged cache size.**

   ```
   # cachefswssize log-file-path
   ```

   For more information, see `cachefswssize`(1M).

Example 18–13 Viewing the Working Set (Cache) Size

In the following example, the `end size` is the size of the cache at the time you issued the `cachefswssize` command. The `high water size` is the largest size of the cache during the time frame in which logging occurred.

```
# cachefswssize /var/tmp/samlog

    /home/sam
           end size:  10688k
    high water size:  10704k

    /
           end size:   1736k
    high water size:   1736k

    /opt
           end size:    128k
    high water size:    128k
```

```
/nfs/saturn.dist
       end size:   1472k
high water size:   1472k

/data/abc
       end size:   7168k
high water size:   7168k

/nfs/venus.svr4
       end size:   4688k
high water size:   5000k

/data
       end size:   4992k
high water size:   4992k

total for cache
   initial size: 110960k
       end size:  30872k
high water size:  30872k
```

Viewing CacheFS Statistics

You might want to view certain information about a specific CacheFS file system. The following table explains the terminology that is displayed in the statistics output.

TABLE 18–2 CacheFS Statistics Terminology

Output Term	Description
`cache hit rate`	The rate of cache hits versus cache misses, followed by the actual number of hits and misses. A cache hit occurs when the user wants to perform an operation on a file or files, and the file or files are actually in the cache. A cache miss occurs when the file is not in the cache. The load on the server is the sum of cache misses, consistency checks, and modifications (modifies).
`consistency checks`	The number of consistency checks performed, followed by the number that passed, and the number that failed.
`modifies`	The number of modify operations. For example, writes or creates.

How to View CacheFS Statistics

View the statistics with the `cachefsstat` command. You can view the statistics at any time. For example, you do not have to set up logging in order to view the statistics.

$ **`cachefsstat`** */mount-point*

/mount-point specifies the CacheFS file system for which you want to view the statistics.

If you do not specify the mount point, statistics for all mounted CacheFS file systems will be displayed.

For more information, see `cachefsstat`(1M).

EXAMPLE 18–14 Viewing CacheFS Statistics

This example shows how to view statistics on the cached file system, `/home/sam`.

```
$ cachefsstat /home/sam
        cache hit rate: 73% (1234 hits, 450 misses)
    consistency checks: 700 (650 pass, 50 fail)
              modifies: 321
garbage collection:  0
```

CHAPTER 19

Configuring Additional Swap Space (Tasks)

This chapter provides guidelines and step-by-step instructions for configuring additional swap space after the Solaris release is installed.

This is a list of step-by-step instructions in this chapter.

- "How to Create a Swap File and Make It Available" on page 328
- "How to Remove Unneeded Swap Space" on page 329

This is a list of the overview information in this chapter.

- "About Swap Space" on page 321
- "How Do I Know If I Need More Swap Space?" on page 324
- "How Swap Space Is Allocated" on page 325
- "Planning for Swap Space" on page 325
- "Monitoring Swap Resources" on page 326
- "Adding More Swap Space" on page 327

About Swap Space

System administrators should understand the features of the SunOS swap mechanism to determine the following:

- Swap space requirements
- The relationship between swap space and the TMPFS file system
- Recovery from error messages related to swap space

Swap Space and Virtual Memory

The Solaris software uses some disk slices for temporary storage rather than for file systems. These slices are called *swap* slices. Swap slices are used as virtual memory storage areas when the system does not have enough physical memory to handle current processes.

The virtual memory system maps physical copies of files on disk to virtual addresses in memory. Physical memory pages that contain the data for these mappings can be backed by regular files in the file system, or by swap space. If the memory is backed by swap space it is referred to as *anonymous* memory because there is no identity assigned to the disk space that is backing the memory.

The Solaris environment uses the concept of *virtual swap space*, a layer between anonymous memory pages and the physical storage (or disk-backed swap space) that actually back these pages. A system's virtual swap space is equal to the sum of all its physical (disk-backed) swap space plus a portion of the currently available physical memory.

Virtual swap space has these advantages:

- The need for large amounts of physical swap space is reduced because virtual swap space does not necessarily correspond to physical (disk) storage.
- A pseudo file system called SWAPFS provides addresses for anonymous memory pages. Because SWAPFS controls the allocation of memory pages, it has greater flexibility in deciding what happens to a page. For example, SWAPFS might change the page's requirements for disk-backed swap storage.

Swap Space and the TMPFS File System

The TMPFS file system is activated automatically in the Solaris environment by an entry in the `/etc/vfstab` file. The TMPFS file system stores files and their associated information in memory (in the `/tmp` directory) rather than on disk, which speeds access to those files. This feature results in a major performance enhancement for applications such as compilers and DBMS products that use `/tmp` heavily.

The TMPFS file system allocates space in the `/tmp` directory from the system's swap resources. This feature means that as you use up space in the `/tmp` directory, you are also using up swap space. So if your applications use the `/tmp` directory heavily and you do not monitor swap space usage, your system could run out of swap space.

Use the following if you want to use TMPFS but your swap resources are limited:

- Mount the TMPFS file system with the size option (`-o` *size*) to control how much swap resources TMPFS can use.
- Use your compiler's `TMPDIR` environment variable to point to another larger directory.

Using your compiler's `TMPDIR` variable only controls whether the compiler is using the `/tmp` directory. This variable has no effect on other programs' use of the `/tmp` directory.

Swap Space as a Dump Device

A dump device is usually disk space that is reserved to store system crash dump information. By default, a system's dump device is configured to be an appropriate swap partition. If possible, you should configure a alternate disk partition as a *dedicated dump device* instead to provide increased reliability for crash dumps and faster reboot time after a system failure. You can configure a dedicated dump device by using the `dumpadm` command. For more information, see Chapter 28, "Managing System Crash Information (Tasks)," in *System Administration Guide: Advanced Administration*.

If you are using a volume manager to manage your disks, such as Solaris Volume Manager, do not configure your dedicated dump device to be under the control of Solaris Volume Manager. You can keep your swap areas under Solaris Volume Manager's control, which is a recommended practice. However, for accessibility and performance reasons, configure another disk as a dedicated dump device outside of Solaris Volume Manager's control.

Swap Space and Dynamic Reconfiguration

A good practice is to allocate enough swap space to support a failing CPU or system board during dynamic reconfiguration. Otherwise, a CPU or system board failure might result in your host or domain rebooting with less memory.

Without having this additional swap space available, one or more of your applications might fail to start due to insufficient memory, which would require manual intervention to either add additional swap space or to reconfigure the memory usage of these applications.

If you have allocated additional swap to handle a potential loss of memory on reboot, all of your intensive applications might start as usual. This means the system will be available to the users, even if possibly slower due to some additional swapping.

For more information, see your specific hardware dynamic reconfiguration guide.

How Do I Know If I Need More Swap Space?

Use the `swap -l` command to determine if your system needs more swap space.

For example, the following `swap -l` output shows that this system's swap space is almost entirely consumed or at 100% allocation.

```
% swap -l
swapfile             dev   swaplo blocks   free
/dev/dsk/c0t0d0s1   136,1      16 1638608    88
```

When a system's swap space is at 100% allocation, an application's memory pages become temporarily locked. Application errors might not occur, but system performance will likely suffer.

For information on adding more swap space to your system, see "How to Create a Swap File and Make It Available" on page 328.

Swap-Related Error Messages

These messages indicate that an application was trying to get more anonymous memory, and there was no swap space left to back it.

```
application is out of memory
```

```
malloc error O
```

```
messages.1:Sep 21 20:52:11 mars genunix: [ID 470503 kern.warning]
WARNING: Sorry, no swap space to grow stack for pid 100295 (myprog)
```

TMPFS-Related Error Messages

The following message is displayed if a page could not be allocated when writing a file. This problem can occur when TMPFS tries to write more than it is allowed or if currently executed programs are using a lot of memory.

```
directory: File system full, swap space limit exceeded
```

The following message means TMPFS ran out of physical memory while attempting to create a new file or directory.

```
directory: File system full, memory allocation failed
```

For information on recovering from the TMPFS-related error messages, see `tmpfs`(7FS).

How Swap Space Is Allocated

Initially, swap space is allocated as part of the Solaris installation process. If you use the installation program's automatic layout of disk slices and do not manually change the size of the swap slice, the Solaris installation program allocates a default swap area of 512 Mbytes.

Starting in the Solaris 9 release, the installation program allocates swap space starting at the first available disk cylinder (typically cylinder 0). This placement provides maximum space for the root (/) file system during the default disk layout and enables the growth of the root (/) file system during an upgrade.

For general guidelines on allocating swap space, see "Planning for Swap Space" on page 325.

You can allocate additional swap space to the system by creating a swap file. For information about creating a swap file, see "Adding More Swap Space" on page 327.

The `/etc/vfstab` File

After the system is installed, swap slices and swap files are listed in the `/etc/vfstab` file. They are activated by the `/sbin/swapadd` script when the system is booted.

An entry for a swap device in the `/etc/vfstab` file contains the following:

- The full path name of the swap slice or swap file
- File system type of swap

The file system that contains a swap file must be mounted before the swap file is activated. So, in the `/etc/vfstab` file, make sure that the entry that mounts the file system comes before the entry that activates the swap file.

Planning for Swap Space

The most important factors in determining swap space size are the requirements of the system's software applications. For example, large applications such as computer-aided-design simulators, database-management products, transaction monitors, and geologic analysis systems can consume as much as 200-1000 Mbytes of swap space.

Consult your application vendor for swap space requirements for their applications.

If you are unable to determine swap space requirements from your application vendor, use the following general guidelines based on your system type to allocate swap space:

System Type	Swap Space Size	Dedicated Dump Device Size
Workstation with approximately 4 Gbytes of physical memory	1 Gbyte	1 Gbyte
Mid-range server with approximately 8 Gbytes of physical memory	2 Gbytes	2 Gbytes
High-end server with approximately 16 to 128 Gbytes of physical memory	4 Gbytes	4 Gbytes

In addition to the general guidelines, consider allocating swap or disk space for the following:

- A dedicated dump device.
- Determine whether large applications (like compilers) will be using the `/tmp` directory. Then allocate additional swap space to be used by TMPFS. For information about TMPFS, see "Swap Space and the TMPFS File System" on page 322.

Monitoring Swap Resources

The `/usr/sbin/swap` command is used to manage swap areas. Two options, `-l` and `-s`, display information about swap resources.

Use the `swap -l` command to identify a system's swap areas. Activated swap devices or files are listed under the `swapfile` column.

```
# swap -l
swapfile             dev  swaplo blocks   free
/dev/dsk/c0t0d0s1   136,1      16 1638608 1600528
```

Use the `swap -s` command to monitor swap resources.

```
# swap -s
total: 57416k bytes allocated + 10480k reserved = 67896k used,
833128k available
```

The `used` value plus the `available` value equals the total swap space on the system, which includes a portion of physical memory and swap devices (or files).

You can use the amount of available and used swap space (in the `swap -s` output) as a way to monitor swap space usage over time. If a system's performance is good, use `swap -s` to see how much swap space is available. When the performance of a system slows down, check the amount of available swap space to see if it has decreased. Then you can identify what changes to the system might have caused swap space usage to increase.

When using this command, keep in mind that the amount of physical memory available for swap usage changes dynamically as the kernel and user processes lock down and release physical memory.

Note – The `swap -l` command displays swap space in 512-byte blocks and the `swap -s` command displays swap space in 1024-byte blocks. If you add up the blocks from `swap -l` and convert them to Kbytes, the result will be less than `used` + `available` (in the `swap -s` output) because `swap -l` does not include physical memory in its calculation of swap space.

The output from the `swap -s` command is summarized in the following table.

TABLE 19–1 Output of the `swap -s` Command

Keyword	Description
`bytes allocated`	The total amount of swap space in 1024-byte blocks that is currently allocated as backing store (disk-backed swap space).
`reserved`	The total amount of swap space in 1024-byte blocks that is not currently allocated, but claimed by memory for possible future use.
`used`	The total amount of swap space in 1024-byte blocks that is either allocated or reserved.
`available`	The total amount of swap space in 1024-byte blocks that is currently available for future reservation and allocation.

Adding More Swap Space

As system configurations change and new software packages are installed, you might need to add more swap space. The easiest way to add more swap space is to use the `mkfile` and `swap` commands to designate a part of an existing UFS or NFS file system as a supplementary swap area. These commands, described in the following sections, enable you to add more swap space without repartitioning a disk.

Alternative ways to add more swap space are to repartition an existing disk or add another disk. For information on how to repartition a disk, see Chapter 10.

Creating a Swap File

The following general steps are involved in creating a swap file:

- Creating a swap file with the `mkfile` command
- Activating the swap file with the `swap` command
- Adding an entry for the swap file in the `/etc/vfstab` file so that the swap file is activated automatically when the system is booted.

The `mkfile` Command

The `mkfile` command creates a file that is suitable for use as either an NFS-mounted or a local swap area. The sticky bit is set, and the file is filled with zeros. You can specify the size of the swap file in bytes (the default) or in Kbytes, blocks, or Mbytes by using the `k`, `b`, or `m` suffixes, respectively.

The following table shows the `mkfile` command options.

TABLE 19–2 Options to the `mkfile` Command

Option	Description
`-n`	Creates an empty file. The size is noted, but the disk blocks are not allocated until data is written to them.
`-v`	Reports the names and sizes of created files.

Caution – Use the `-n` option only when you create an NFS swap file.

▼ How to Create a Swap File and Make It Available

Steps

1. **Become superuser.**

   You can create a swap file without root permissions. However, to avoid accidental overwriting, root should be the owner of the swap file.

2. **Create a directory for the swap file, if needed.**

3. **Create the swap file.**

   ```
   # mkfile nnn[k|b|m] filename
   ```

   The swap file of the size *nnn* (in Kbytes, bytes, or Mbytes) and filename you specify is created.

4. **Activate the swap file.**

   ```
   # /usr/sbin/swap -a /path/filename
   ```

   You must use the absolute path name to specify the swap file. The swap file is added and available until the file system is unmounted, the system is rebooted, or the swap file is removed. Keep in mind that you can't unmount a file system while some process or program is swapping to the swap file.

5. **Add an entry for the swap file to the `/etc/vfstab` file that specifies the full path name of the file, and designates `swap` as the file system type, as follows:**

   ```
   /path/filename   -      -        swap      -      no      -
   ```

6. **Verify that the swap file is added.**

   ```
   $ /usr/sbin/swap -l
   ```

Example 19–1 Creating a Swap File and Making It Available

The following examples shows how to create a 100–Mbyte swap file called `/files/swapfile`.

```
# mkdir /files
# mkfile 100m /files/swapfile
# swap -a /files/swapfile
# vi /etc/vfstab
(An entry is added for the swap file):
/files/swapfile   -      -       swap      -     no      -
# swap -l
swapfile             dev  swaplo blocks   free
/dev/dsk/c0t0d0s1   136,1      16 1638608 1600528
/files/swapfile        -       16 204784  204784
```

Removing a Swap File From Use

If you have unneeded swap space, you can remove it.

▼ How to Remove Unneeded Swap Space

Steps

1. **Become superuser.**
2. **Remove the swap space.**

   ```
   # /usr/sbin/swap -d /path/filename
   ```

The swap file name is removed so that it is no longer available for swapping. The file itself is not deleted.

3. **Edit the `/etc/vfstab` file and delete the entry for the swap file.**

4. **Recover the disk space so that you can use it for something else.**

   ```
   # rm /path/filename
   ```

   If the swap space is a file, remove it. Or, if the swap space is on a separate slice and you are sure you will not need it again, make a new file system and mount the file system.

   For information on mounting a file system, see Chapter 17.

5. **Verify that the swap file is no longer available.**

   ```
   # swap -l
   ```

Example 19–2 Removing Unneeded Swap Space

The following examples shows how to delete the `/files/swapfile` swap file.

```
# swap -d /files/swapfile
# (Remove the swap entry from the /etc/vfstab file)
# rm /files/swapfile
# swap -l
swapfile             dev  swaplo  blocks   free
/dev/dsk/c0t0d0s1   136,1      16 1638608 1600528
```

CHAPTER 20

Checking UFS File System Consistency (Tasks)

This chapter provides overview information and step-by-step instructions about checking UFS file system consistency.

This is a list of step-by-step instructions in this chapter.

- "How to Check the root (/) or /usr File Systems From an Alternate Boot Device" on page 341
- "How to Check Non-root (/) or Non-/usr File Systems" on page 343
- "How to Preen a UFS File System" on page 345
- "How to Restore a Bad Superblock" on page 346

This is a list of the overview information in this chapter.

- "File System Consistency" on page 332
- "How the File System State Is Recorded" on page 332
- "What the `fsck` Command Checks and Tries to Repair" on page 333
- "Interactively Checking and Repairing a UFS File System" on page 340
- "Restoring a Bad Superblock" on page 346
- "Syntax and Options for the `fsck` Command" on page 348

For information about `fsck` error messages, see Chapter 32, "Resolving UFS File System Inconsistencies (Tasks)," in *System Administration Guide: Advanced Administration*.

For background information on the UFS file system structures referred to in this chapter, see Chapter 21.

File System Consistency

The UFS file system relies on an internal set of tables to keep track of inodes used and available blocks. When these internal tables are not properly synchronized with data on a disk, inconsistencies result and file systems need to be repaired.

File systems can be inconsistent because of abrupt termination of the operating system in these ways:

- Power failure
- Accidental unplugging of the system
- Turning off the system without proper shutdown procedure
- A software error in the kernel

File system inconsistencies, while serious, are not common. When a system is booted, a check for file system consistency is automatically performed (with the `fsck` command). Most of the time, this file system check repairs problems it encounters.

The `fsck` command places files and directories that are allocated but unreferenced in the `lost+found` directory. A inode number is assigned as the name of unreferenced file and directory. If the `lost+found` directory does not exist, the `fsck` command creates it. If there is not enough space in the `lost+found` directory, the `fsck` command increases its size.

For a description of inodes, see "Inodes" on page 358.

How the File System State Is Recorded

The `fsck` command uses a state flag, which is stored in the superblock, to record the condition of the file system. This flag is used by the `fsck` command to determine whether a file system needs to be checked for consistency. The flag is used by the `/sbin/rcS` script during booting and by the `fsck -m` command. If you ignore the result from the `fsck -m` command, all file systems can be checked regardless of the setting of the state flag.

For a description of the superblock, see "The Superblock" on page 358.

The possible state flag values are described in the following table.

TABLE 20–1 Values of File System State Flags

State Flag Values	Description
FSACTIVE	Indicates a mounted file system that has modified data in memory. A mounted file system with this state flag indicates that user data or metadata would be lost if power to the system is interrupted.
FSBAD	Indicates that this file system contains inconsistent file system data.
FSCLEAN	Indicates an undamaged, cleanly unmounted file system.
FSLOG	Indicates that the file system has logging enabled. A file system with this flag set is either mounted or unmounted. If a file system has logging enabled, the only flags that it can have are `FSLOG` or `FSBAD`. A non-logging file system can have `FSACTIVE`, `FSSTABLE`, or `FSCLEAN`.
FSSTABLE	Indicates an idle mounted file system. A mounted file system with this state flag indicates that neither user data nor metadata would be lost if power to the system is interrupted.

The following table shows how the state flag is modified by the `fsck` command, based on its initial state.

TABLE 20–2 How the State Flag is Modified by `fsck`

Initial State: Before `fsck`	State After `fsck`		
	No Errors	All Errors Corrected	Uncorrected Errors
unknown	FSSTABLE	FSSTABLE	unknown
FSACTIVE	FSSTABLE	FSSTABLE	FSACTIVE
FSSTABLE	FSSTABLE	FSSTABLE	FSACTIVE
FSCLEAN	FSCLEAN	FSSTABLE	FSACTIVE
FSBAD	FSSTABLE	FSSTABLE	FSBAD
FSLOG	FSLOG	FSLOG	FSLOG

What the `fsck` Command Checks and Tries to Repair

This section describes what happens in the normal operation of a file system, what can go wrong, what problems the `fsck` command (the checking and repair utility) looks for, and how this command corrects the inconsistencies it finds.

Why Inconsistencies Might Occur

Every working day hundreds of files might be created, modified, and removed. Each time a file is modified, the operating system performs a series of file system updates. These updates, when written to the disk reliably, yield a consistent file system.

When a user program does an operation to change the file system, such as a write, the data to be written is first copied into an in-core buffer in the kernel. Normally, the disk update is handled asynchronously. The user process is allowed to proceed even though the data write might not happen until long after the write system call has returned. Thus, at any given time, the file system, as it resides on the disk, lags behind the state of the file system that is represented by the in-core information.

The disk information is updated to reflect the in-core information when the buffer is required for another use or when the kernel automatically runs the `fsflush` daemon (at 30-second intervals). If the system is halted without writing out the in-core information, the file system on the disk might be in an inconsistent state.

A file system can develop inconsistencies in several ways. The most common causes are operator error and hardware failures.

Problems might result from an *unclean shutdown*, if a system is shut down improperly, or when a mounted file system is taken offline improperly. To prevent unclean shutdowns, the current state of the file systems must be written to disk (that is, "synchronized") before you shut down the system, physically take a disk pack out of a drive, or take a disk offline.

Inconsistencies can also result from defective hardware or problems with the disk or controller firmware. Blocks can become damaged on a disk drive at any time, or a disk controller can stop functioning correctly.

The UFS Components That Are Checked for Consistency

This section describes the kinds of consistency checks that the `fsck` command applies to these UFS file system components: superblock, cylinder group blocks, inodes, indirect blocks, and data blocks.

For information about UFS file system structures, see "The Structure of Cylinder Groups for UFS File Systems" on page 357.

Superblock Checks

The superblock stores summary information, which is the most commonly corrupted component in a UFS file system. Each change to the file system inodes or data blocks also modifies the superblock. If the CPU is halted and the last command is not a `sync` command, the superblock almost certainly becomes corrupted.

The superblock is checked for inconsistencies in the following:

- File system size
- Number of inodes
- Free block count
- Free inode count

File System Size and Inode List Size Checks

The file system size must be larger than the number of blocks used by the superblock and the list of inodes. The number of inodes must be less than the maximum number allowed for the file system. An inode represents all the information about a file. The file system size and layout information are the most critical pieces of information for the `fsck` command. Although there is no way to actually check these sizes because they are statically determined when the file system is created. However, the `fsck` command can check that the sizes are within reasonable bounds. All other file system checks require that these sizes be correct. If the `fsck` command detects corruption in the static parameters of the primary superblock, it requests the operator to specify the location of an alternate superblock.

For more information about the structure of the UFS file system, see "The Structure of Cylinder Groups for UFS File Systems" on page 357.

Free Block Checks

Free blocks are stored in the cylinder group block maps. The `fsck` command checks that all the blocks marked as free are not claimed by any files. When all the blocks have been accounted for, the `fsck` command checks to see if the number of free blocks plus the number of blocks that are claimed by the inodes equal the total number of blocks in the file system. If anything is wrong with the block maps, the `fsck` command rebuilds them, leaving out blocks already allocated.

The summary information in the superblock includes a count of the total number of free blocks within the file system. The `fsck` command compares this count to the number of free blocks it finds within the file system. If the counts do not agree, the `fsck` command replaces the count in the superblock with the actual free-block count.

Free Inode Checks

The summary information in the superblock contains a count of the free inodes within the file system. The `fsck` command compares this count to the number of free inodes it finds within the file system. If the counts do not agree, `fsck` replaces the count in the superblock with the actual free inode count.

Inodes

The list of inodes is checked sequentially starting with inode 2 (inode 0 and inode 1 are reserved). Each inode is checked for inconsistencies in the following:

- Format and type
- Link count
- Duplicate block
- Bad block numbers
- Inode size

Format and Type of Inodes

Each inode contains a mode word, which describes the type and state of the inode. Inodes might be one of nine types:

- Regular
- Directory
- Block special
- Character special
- FIFO (named-pipe)
- Symbolic link
- Shadow (used for ACLs)
- Attribute directory
- Socket

Inodes might be in one of three states:

- Allocated
- Unallocated
- Partially allocated

When the file system is created, a fixed number of inodes are set aside, but they are not allocated until they are needed. An allocated inode is one that points to a file. An unallocated inode does not point to a file and, therefore, should be empty. The partially allocated state means that the inode is incorrectly formatted. An inode can get into this state if, for example, bad data is written into the inode list because of a hardware failure. The only corrective action the `fsck` command can take is to clear the inode.

Link Count Checks

Each inode contains a count of the number of directory entries linked to it. The `fsck` command verifies the link count of each inode by examining the entire directory structure, starting from the root directory, and calculating an actual link count for each inode.

Discrepancies between the link count stored in the inode and the actual link count as determined by the `fsck` command might be of three types:

- The stored count is *not* 0 and the actual count is 0.

  This condition can occur if no directory entry exists for the inode. In this case, the `fsck` command puts the disconnected file in the `lost+found` directory.

- The stored count is *not* 0 and the actual count is *not* 0, but the counts are *unequal*.

  This condition can occur if a directory entry has been added or removed, but the inode has not been updated. In this case, the `fsck` command replaces the stored link count with the actual link count.
- The stored count is 0 and the actual count is not 0.

  In this case, the `fsck` command changes the link count of the inode to the actual count.

Duplicate Block Checks

Each inode contains a list, or pointers to lists (indirect blocks), of all the blocks claimed by the inode. Because indirect blocks are owned by an inode, inconsistencies in indirect blocks directly affect the inode that owns the indirect block.

The `fsck` command compares each block number claimed by an inode to a list of allocated blocks. If another inode already claims a block number, the block number is put on a list of duplicate blocks. Otherwise, the list of allocated blocks is updated to include the block number.

If there are any duplicate blocks, the `fsck` command makes a second pass of the inode list to find the other inode that claims each duplicate block. (A large number of duplicate blocks in an inode might be caused by an indirect block not being written to the file system.) It is not possible to determine with certainty which inode is in error. The `fsck` command prompts you to choose which inode should be kept and which should be cleared.

Bad Block Number Checks

The `fsck` command checks each block number claimed by an inode to see that its value is higher than that of the first data block and lower than that of the last data block in the file system. If the block number is outside this range, it is considered a bad block number.

Bad block numbers in an inode might be caused by an indirect block not being written to the file system. The `fsck` command prompts you to clear the inode.

Inode Size Checks

Each inode contains a count of the number of data blocks that it references. The number of actual data blocks is the sum of the allocated data blocks and the indirect blocks. The `fsck` command computes the number of data blocks and compares that block count against the number of blocks that the inode claims. If an inode contains an incorrect count, the `fsck` command prompts you to fix it.

Each inode contains a 64-bit size field. This field shows the number of characters (data bytes) in the file associated with the inode. A rough check of the consistency of the size field of an inode is done by using the number of characters shown in the size field to calculate how many blocks should be associated with the inode, and then comparing that to the actual number of blocks claimed by the inode.

Indirect Blocks

Indirect blocks are owned by an inode. Therefore, inconsistencies in an indirect block affect the inode that owns it. Inconsistencies that can be checked are the following:

- Blocks already claimed by another inode
- Block numbers outside the range of the file system

These consistency checks listed are also performed for indirect blocks.

Data Blocks

An inode can directly or indirectly reference three kinds of data blocks. All referenced blocks must be of the same kind. The three types of data blocks are the following:

- Plain data blocks
- Symbolic-link data blocks
- Directory data blocks

Plain data blocks contain the information stored in a file. Symbolic-link data blocks contain the path name stored in a symbolic link. Directory data blocks contain directory entries. The `fsck` command can check only the validity of directory data blocks.

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

- Directory inode numbers that point to unallocated inodes
- Directory inode numbers that are greater than the number of inodes in the file system
- Incorrect directory inode numbers for "`.`" and "`..`" directories
- Directories that are disconnected from the file system

Directory Unallocated Checks

If the inode number in a directory data block points to an unallocated inode, the `fsck` command removes the directory entry. This condition can occur if the data blocks that contain a new directory entry are modified and written out, but the inode does not get written out. This condition can occur if the CPU is shutdown abruptly.

Bad Inode Number Checks

If a directory entry inode number points beyond the end of the inode list, the `fsck` command removes the directory entry. This condition can occur when bad data is written into a directory data block.

Incorrect "`.`" and "`..`" Entry Checks

The directory inode number entry for "`.`" must be the first entry in the directory data block. The directory inode number must reference itself; that is, its value must be equal to the inode number for the directory data block.

The directory inode number entry for "`..`" must be the second entry in the directory data block. The directory inode number value must be equal to the inode number of the parent directory (or the inode number of itself if the directory is the root directory).

If the directory inode numbers for "`.`" and "`..`" are incorrect, the `fsck` command replaces them with the correct values. If there are multiple hard links to a directory, the first hard link found is considered the real parent to which "`..`" should point. In this case, the `fsck` command recommends that you have it delete the other names.

Disconnected Directories

The `fsck` command checks the general connectivity of the file system. If a directory is found that is not linked to the file system, the `fsck` command links the directory to the `lost+found` directory of the file system. This condition can occur when inodes are written to the file system, but the corresponding directory data blocks are not.

Regular Data Blocks

Data blocks associated with a regular file hold the contents of the file. The `fsck` command does not attempt to check the validity of the contents of a regular file's data blocks.

The `fsck` Summary Message

When you run the `fsck` command interactively and it completes successfully, a message similar to the following is displayed:

```
# fsck /dev/rdsk/c0t0d0s7
** /dev/rdsk/c0t0d0s7
** Last Mounted on /export/home
** Phase 1 - Check Blocks and Sizes
** Phase 2 - Check Pathnames
** Phase 3 - Check Connectivity
** Phase 4 - Check Reference Counts
** Phase 5 - Check Cyl groups
2 files, 9 used, 2833540 free (20 frags, 354190 blocks, 0.0% fragmentation)
#
```

The last line of `fsck` output describes the following information about the file system:

# `files`	Number of inodes in use
# `used`	Number of fragments in use
# `free`	Number of unused fragments
# `frags`	Number of unused non-block fragments
# `blocks`	Number of unused full blocks
% `fragmentation`	Percentage of fragmentation, where: free fragments x 100 / total fragments in the file system

For information about fragments, see "Fragment Size" on page 362.

Interactively Checking and Repairing a UFS File System

You might need to interactively check file systems in the following instances:

- When they cannot be mounted
- When they develop inconsistences while in use

When an in-use file system develops inconsistencies, error messages might be displayed in the console window, the system messages file, or the system might crash. For example, the system messages file, `/var/adm/messages`, might include messages similar to the following:

```
Sep  5 13:42:40 hostname ufs: [ID 879645 kern.notice] NOTICE: /: unexpected
free inode 630916, run fsck(1M)
```

hostname is the system reporting the error.

Before using the `fsck` command, you might want to refer to "Syntax and Options for the `fsck` Command" on page 348 and Chapter 32, "Resolving UFS File System Inconsistencies (Tasks)," in *System Administration Guide: Advanced Administration* for information on resolving `fsck` error messages.

Keep the following points in mind when running the `fsck` command to check UFS file systems:

- A file system *should* be inactive when using `fsck` to check that file system. File system changes waiting to be flushed to disk or file system changes that occur during the `fsck` checking process can be interpreted as file system corruption and may not be a reliable indication of a problem.
- A file system *must* be inactive when using `fsck` to repair that file system. File system changes waiting to be flushed to disk or file system changes that occur during the `fsck` repairing process might cause the file system to become corrupted or might cause the system to crash.
- Unmount a file system before using `fsck` on that file system, to ensure that it is inactive and that all file system data structures are consistent as possible. The only exceptions are for the active root (`/`) and `/usr` file systems, because they must be mounted to run `fsck`.
- If you need to repair the root (`/`) or `/usr` file systems, boot the system from an alternate device, if possible, so that these file systems are unmounted and inactive.

  For step-by-step instructions on running `fsck` on the root (`/`) or `/usr` file system, see "How to Check the root (`/`) or `/usr` File Systems From an Alternate Boot Device" on page 341.

▼ How to Check the root (`/`) or `/usr` File Systems From an Alternate Boot Device

This procedure assumes that a local CD or network boot server is available so that you can boot the system from an alternate device.

For information on restoring a bad superblock, see "How to Restore a Bad Superblock" on page 346.

Steps

1. **Become superuser or assume an equivalent role.**
2. **For systems with mirrored root (/) file systems only: Detach the root (`/`) mirror before booting from the alternate device or you risk corrupting the file system.**

   For information on detaching the root (`/`) mirror, see "Working With Submirrors" in *Solaris Volume Manager Administration Guide*.
3. **Identify the device, such as `/dev/dsk/c0t0d0s0`, of the root (`/`) or `/usr` file system that needs to be checked.**

   You'll need to supply this device name when booted from an alternate device. It will more difficult to identify this device when you are already booted from the alternate device.
4. **Boot the system with the root (`/`) or `/usr` file system that needs to be checked from an alternate device, such as a local CD or the network, in single-user mode to ensure that there is no activity on these file systems.**

For example:

```
# init 0
ok boot net -s
.
.
.
#
```

5. **Check the device that contains the root (/) or `/usr` file system as identified in step #3.**

   If the hardware for the file system to be checked or repaired has changed, the device names might have changed. Be sure to check that the `fsck -n` message `Last Mounted on ...` indicates the expected device for the file system.

   For example, the root file system to be checked is `/dev/dsk/c0t0d0s0`.

   ```
   # fsck -n /dev/rdsk/c0t0d0s0
   ** /dev/rdsk/c0t0d0s0 (NO WRITE)
   ** Last Mounted on /
   .
   .
   .
   fsck /dev/rdsk/c0t0d0s0
   ** /dev/rdsk/c0t0d0s0
   ** Last Mounted on /
   ** Phase 1 - Check Blocks and Sizes
   ** Phase 2 - Check Pathnames
   .
   .
   .
   ```

6. **Correct any reported `fsck` errors.**

   For information about how to respond to the error message prompts while interactively checking one or more UFS file systems, see Chapter 32, "Resolving UFS File System Inconsistencies (Tasks)," in *System Administration Guide: Advanced Administration*.

7. **If necessary, run the `fsck` command again if you see messages similar to the following, `FILE SYSTEM STATE NOT SET TO OKAY` or `FILE SYSTEM MODIFIED`.**

   The `fsck` command might be unable to fix all errors in one execution.

   If `fsck` cannot repair all of the problems after running it several times, see "Fixing a UFS File System That the `fsck` Command Cannot Repair" on page 345.

8. **Mount the repaired file system to see if there are any files in the `lost+found` directory.**

   Individual files put in the `lost+found` directory by the `fsck` command are renamed with their inode numbers. If possible, rename the files and move them where they belong. You might be able to use the `grep` command to match phrases within individual files and the `file` command to identify file types.

Eventually, remove unidentifiable files or directories left in the `lost+found` directory so it doesn't fill it up unnecessarily.

9. **Bring the system back to multi-user mode.**

   ```
   # init 6
   ```

   If you press Control-D when booted in single-user mode from an alternate device, the system will start the Solaris installation process.

10. **For systems with mirrored root (/) file systems only: Reattach the root (/) mirror.**

▼ How to Check Non-root (/) or Non-`/usr` File Systems

This procedure assumes that the file system to be checked is unmounted.

For information on restoring a bad superblock, see "How to Restore a Bad Superblock" on page 346.

Steps

1. **Become superuser or assume an equivalent role.**

2. **Unmount the local file system first to ensure that there is no activity on the file system.**

   Specify the mount point directory or `/dev/dsk/`*device-name* as arguments to the `fsck` command. Any inconsistency messages are displayed.

   For example:

   ```
   # umount /export/home
   # fsck /dev/rdsk/c0t0d0s7
   ** /dev/dsk/c0t0d0s7
   ** Last Mounted on /export/home
   .
   .
   .
   ```

3. **Correct any reported `fsck` errors.**

   For information about how to respond to the error message prompts while interactively checking one or more UFS file systems, see Chapter 32, "Resolving UFS File System Inconsistencies (Tasks)," in *System Administration Guide: Advanced Administration*.

4. **If necessary, run the `fsck` command again if you see the following messages, `FILE SYSTEM STATE NOT SET TO OKAY` or `FILE SYSTEM MODIFIED`.**

   The `fsck` command might be unable to fix all errors in one execution.

If `fsck` cannot repair all of the problems after running it several times, see "Fixing a UFS File System That the `fsck` Command Cannot Repair" on page 345.

5. **Mount the repaired file system to see if there are any files in the `lost+found` directory.**

   Individual files put in the `lost+found` directory by the `fsck` command are renamed with their inode numbers. If possible, rename the files and move them where they belong. You might be able to use the `grep` command to match phrases within individual files and the `file` command to identify file types.

   Eventually, remove unidentifiable files or directories left in the `lost+found` directory so it doesn't fill it up unnecessarily.

6. **Rename and move any files put in the `lost+found` directory.**

Example 20–1 Checking Non-root (/) or Non-`/usr` File Systems Interactively

The following example shows how to check the `/dev/rdsk/c0t0d0s6` file system and corrects the incorrect block count. This example assumes that the file system is unmounted.

```
# fsck /dev/rdsk/c0t0d0s6
** Phase 1 - Check Block and Sizes
INCORRECT BLOCK COUNT I=2529 (6 should be 2)
CORRECT? y

** Phase 2 - Check Pathnames
** Phase 3 - Check Connectivity
** Phase 4 - Check Reference Counts
** Phase 5 - Cylinder Groups
929 files, 8928 used, 2851 free (75 frags, 347 blocks, 0.6%
fragmentation)
/dev/rdsk/c0t0d0s6 FILE SYSTEM STATE SET TO OKAY

***** FILE SYSTEM WAS MODIFIED *****
```

Preening UFS File Systems

The `fsck -o p` command (p is for preen) checks UFS file systems and automatically fixes the problems that normally result from an unexpected system shutdown. This command exits immediately if it encounters a problem that requires operator intervention. This command also permits parallel checking of file systems.

You can run the `fsck -o p` command to preen the file systems after an unclean shutdown. In this mode, the `fsck` command does not look at the clean flag and does a full check. These actions are a subset of the actions that the `fsck` command takes when it runs interactively.

▼ How to Preen a UFS File System

This procedure assumes that the file system is unmounted or inactive.

Steps

1. **Become superuser or assume an equivalent role.**
2. **Unmount the UFS file system.**

   ```
   # umount /mount-point
   ```
3. **Check the UFS file system with the preen option.**

   ```
   # fsck -o p /dev/rdsk/device-name
   ```

   You can preen individual file systems by using */mount-point* or `/dev/rdsk/`*device-name* as arguments to the `fsck` command.

Example 20–2 Preening a UFS File System

The following example shows how to preen the `/export/home` file system.

```
# fsck -o p /export/home
```

Fixing a UFS File System That the `fsck` Command Cannot Repair

The `fsck` command operates in several passes, and a problem corrected in a later pass can expose other problems that are only detected by earlier passes. Therefore, it is sometimes necessary to run `fsck` repeatedly until it no longer reports any problems, to ensure that all errors have been found and repaired. The `fsck` command does not keep running until it comes up clean, so you must rerun it manually.

Pay attention to the information displayed by the `fsck` command. This information might help you fix the problem. For example, the messages might point to a damaged directory. If you delete the directory, you might find that the `fsck` command runs cleanly.

If the `fsck` command still cannot repair the file system, you can try to use the `ff`, `clri`, and `ncheck` commands to figure out and fix what is wrong. For information about how to use these commands, see `fsdb`(1M), `ff`(1M), `clri`(1M), and `ncheck`(1M). You might, ultimately, need to re-create the file system and restore its contents from backup media.

For information about restoring complete file systems, see Chapter 25.

If you cannot fully repair a file system but you can mount it read-only, try using the `cp`, `tar`, or `cpio` commands to retrieve all or part of the data from the file system.

If hardware disk errors are causing the problem, you might need to reformat and divide the disk into slices again before re-creating and restoring file systems. Check that the device cables and connectors are functional before replacing the disk device. Hardware errors usually display the same error again and again across different commands. The `format` command tries to work around bad blocks on the disk. If the disk is too severely damaged, however, the problems might persist, even after reformatting. For information about using the `format` command, see `format`(1M). For information about installing a new disk, see Chapter 12 or Chapter 13.

Restoring a Bad Superblock

When the superblock of a file system becomes damaged, you must restore it. The `fsck` command tells you when a superblock is bad. Fortunately, copies of the superblock are stored within a file system. You can use the `fsck -o b` command to replace the superblock with one of the copies.

For more information about the superblock, see "The Superblock" on page 358.

If the superblock in the root (/) file system becomes damaged and you cannot restore it, you have two choices:

- Reinstall the system
- Boot from the network or local CD, and attempt the following steps. If these steps fail, recreate the root (/) file system with the `newfs` command and restore it from a backup copy.

▼ How to Restore a Bad Superblock

Steps

1. **Become superuser or assume an equivalent role.**
2. **Determine whether the bad superblock is in the root (/) or `/usr` file system and select one of the following:**
   a. **Stop the system and boot from the network or a locally-connected CD if the bad superblock is in the root (/) or `/usr` file system.**

      From a locally-connected CD, use the following command:

      ```
      ok boot cdrom -s
      ```

      From the network where a boot or install server is already setup, use the following command:

```
ok boot net -s
```

If you need help stopping the system, see Chapter 10, “Booting a System (Tasks),” in *System Administration Guide: Basic Administration* or Chapter 11, “Booting a System (Tasks),” in *System Administration Guide: Basic Administration*.

b. **Change to a directory outside the damaged file system and unmount the file system if the bad superblock is not in the root (/) or `/usr` file system.**

```
# umount /mount-point
```

Caution – Be sure to use the `newfs -N` in the next step. If you omit the `-N` option, you will destroy all of the data in the file system and replace it with an empty file system.

3. **Display the superblock values with the `newfs -N` command.**

```
# newfs -N /dev/rdsk/device-name
```

The output of this command displays the block numbers that were used for the superblock copies when the `newfs` command created the file system, unless the file system was created with special parameters. For information on creating a customized file system, see “Custom File System Parameters” on page 361.

4. **Provide an alternate superblock with the `fsck` command.**

```
# fsck -F ufs -o b=block-number /dev/rdsk/device-name
```

The `fsck` command uses the alternate superblock you specify to restore the primary superblock. You can always try `32` as an alternate block, or use any of the alternate blocks shown by the `newfs -N` command.

Example 20–3 Restoring a Bad Superblock

The following example shows how to restore the superblock copy `5264`.

```
# newfs -N /dev/rdsk/c0t3d0s7
/dev/rdsk/c0t3d0s7: 163944 sectors in 506 cylinders of 9 tracks, 36 sectors
 83.9MB in 32 cyl groups (16 c/g, 2.65MB/g, 1216 i/g)
super-block backups (for fsck -b #) at:
 32, 5264, 10496, 15728, 20960, 26192, 31424, 36656, 41888,
 47120, 52352, 57584, 62816, 68048, 73280, 78512, 82976, 88208,
 93440, 98672, 103904, 109136, 114368, 119600, 124832, 130064, 135296,
 140528, 145760, 150992, 156224, 161456,
# fsck -F ufs -o b=5264 /dev/rdsk/c0t3d0s7
Alternate superblock location: 5264.
** /dev/rdsk/c0t3d0s7
** Last Mounted on
** Phase 1 - Check Blocks and Sizes
** Phase 2 - Check Pathnames
** Phase 3 - Check Connectivity
** Phase 4 - Check Reference Counts
** Phase 5 - Check Cyl groups
36 files, 867 used, 75712 free (16 frags, 9462 blocks, 0.0% fragmentation)
/dev/rdsk/c0t3d0s7 FILE SYSTEM STATE SET TO OKAY

***** FILE SYSTEM WAS MODIFIED *****
#
```

Syntax and Options for the `fsck` Command

The `fsck` command checks and repairs inconsistencies in file systems. If you run the `fsck` command without any options, it interactively asks for confirmation before making repairs. This command has four options:

Command and Option	Description
`fsck -m`	Checks whether a file system can be mounted
`fsck -y`	Assumes a yes response for all repairs
`fsck -n`	Assumes a no response for all repairs
`fsck -o p`	Noninteractively preens the file system, fixing all expected (innocuous) inconsistencies, but exits when a serious problem is encountered

CHAPTER 21

UFS File System (Reference)

This is a list of the reference information in this chapter.

Default Directories for root (/) and `/usr` File Systems

The `/kernel` directory contains only platform-independent objects, including a platform-independent kernel, `genunix`. For a description of `/platform` and `/usr/platform`, the platform-dependent directories, see Table 21–3.

The following table describes the directories that are contained in the root (/) file system.

TABLE 21–1 Default Directories in the root (/) File System

Directory	Description
`/`	Root of the overall file system name space
`/dev`	Primary location for logical device files
`/dev/cfg`	Symbolic links to physical ap_ids
`/dev/cua`	Device files for `uucp`

TABLE 21–1 Default Directories in the root (/) File System *(Continued)*

Directory	Description
`/dev/dsk`	Block disk devices
`/dev/fbs`	Frame buffer device files
`/dev/fd`	File descriptors
`/dev/md`	Volume management device names
`/dev/printers`	USB printer device files
`/dev/pts`	`pty` slave devices
`/dev/rdsk`	Raw disk devices
`/dev/rmt`	Raw tape devices
`/dev/sad`	Entry points for the STREAMS Administrative Driver
`/dev/sound`	Audio device and audio device control files
`/dev/swap`	Default swap device
`/dev/term`	Serial devices
`/devices`	Physical device files
`/etc`	Host-specific system administration configuration files and databases
`/etc/acct`	Accounting configuration information
`/etc/apache`	Apache configuration files
`/etc/cron.d`	Configuration information for `cron`
`/etc/default`	Defaults information for various programs
`/etc/dfs`	Configuration information for shared file systems
`/etc/dhcp`	Dynamic Host Configuration Protocol (DHCP) configuration files
`/etc/dmi`	Solstice Enterprise Agents configuration files
`/etc/fn`	Federated Naming Service and x.500 support files
`/etc/fs`	Binaries organized by file system types
`/etc/ftpd`	`ftpd` configuration files
`/etc/gss`	Generic Security Service (GSS) Application Program Interface configuration files
`/etc/gtk`	GNOME (GNU Network Object Model Environment) configuration files

TABLE 21–1 Default Directories in the root (/) File System *(Continued)*

Directory	Description
`/etc/inet`	Configuration files for Internet services
`/etc/init.d`	Scripts for changing run levels
`/etc/iplanet`	iPlanet configuration files
`/etc/krb5`	Kerberos configuration files
`/etc/lib`	Dynamic linking libraries that are needed when `/usr` is not available
`/etc/llc2`	Logical link control (`llc2`) driver configuration files
`/etc/lp`	Configuration information for the printer subsystem
`/etc/lu`	Solaris Live Upgrade configuration files
`/etc/lvm`	Solaris Volume Manager configuration files
`/etc/mail`	Mail subsystem configuration information
`/etc/nca`	Solaris Network Cache and Accelerator (NCA) configuration files
`/etc/net`	Configuration information for TI (transport- independent) network services
`/etc/nfs`	NFS server logging configuration file
`/etc/openwin`	OpenWindows configuration files
`/etc/opt`	Configuration information for optional packages
`/etc/ppp`	Solaris PPP configuration files
`/etc/rc0.d`	Scripts for entering or leaving run level 0
`/etc/rc1.d`	Scripts for entering or leaving run level 1
`/etc/rc2.d`	Scripts for entering or leaving run level 2
`/etc/rc3.d`	Scripts for entering or leaving run level 3
`/etc/rcS.d`	Scripts for bringing the system to single-user mode
`/etc/rcm`	Directory for reconfiguration manager (RCM) custom scripts
`/etc/rpcsec`	Might contain an NIS+ authentication configuration file
`/etc/saf`	Service access facility files (including FIFOs)
`/etc/security`	Basic Security Module (BSM) configuration files
`/etc/sfw`	Samba configuration files
`/etc/skel`	Default profile scripts for new user accounts

TABLE 21–1 Default Directories in the root (/) File System *(Continued)*

Directory	Description
`/etc/smartcard`	Solaris SmartCards configuration files
`/etc/snmp`	Solstice Enterprise Agents configuration files
`/etc/ssh`	Secure shell configuration files
`/etc/sysevent`	`syseventd` configuration files
`/etc/tm`	Trademark files, whose contents are displayed at boot time
`/etc/usb`	USB configuration information
`/etc/uucp`	`uucp` configuration information
`/etc/wrsm`	WCI Remote Shared Memory (WRSM) configuration information
`/export`	Default directory for users' home directories, client file systems, or other shared file systems
`/home`	Default directory or mount point for a user's home directory on a standalone system. When AutoFS is running, you cannot create any new entries in this directory.
`/kernel`	Directory of platform-independent loadable kernel modules that are required as part of the boot process. Includes the generic part of the core kernel that is platform-independent, `/kernel/genunix`. See Table 21–3 for the `/platform` and `/usr/platform` directory structure.
`/mnt`	Convenient, temporary mount point for file systems
`/opt`	Default directory or mount point for add-on application packages
`/platform`	Supported platform files. For more information, see Table 21–3.
`/proc`	Process information
`/sbin`	Essential executables used in the booting process and in manual system failure recovery
`/tmp`	Temporary files, whose contents are cleared during boot sequence
`/usr`	Mount point for the `/usr` file system. For more information, see Table 21–2.
`/var`	Directory for varying files, which usually includes temporary files, logging files, or status files
`/var/adm`	System logging files and accounting files

TABLE 21–1 Default Directories in the root (/) File System *(Continued)*

Directory	Description
`/var/apache`	Scripts, icons, logs, and cache pages for Apache web server
`/var/audit`	Basic Security Module (BSM) audit files
`/var/crash`	Default depository for kernel crash dumps
`/var/cron`	`cron`’s log file
`/var/dmi`	Solstice Enterprise Agents Desktop Management Interface (DMI) run-time components
`/var/dt`	`dtlogin` configuration files
`/var/inet`	IPv6 router state files
`/var/krb5`	Database and log files for Kerberos
`/var/ld`	Configuration files for run-time linker
`/var/ldap`	LDAP client configuration files
`/var/log`	System log files
`/var/lp`	Line printer subsystem logging information
`/var/mail`	Directory where user mail is kept
`/var/news`	Community service messages. These messages are not the same as USENET-style news.
`/var/nfs`	NFS server log files
`/var/nis`	NIS+ databases
`/var/ntp`	Network Time Protocol (NTP) server state directory
`/var/opt`	Root of a subtree for varying files that are associated with software packages
`/var/preserve`	Backup files for `vi` and `ex`
`/var/run`	Temporary system files that are not needed across system reboots. A TMPFS-mounted directory.
`/var/sadm`	Databases that are maintained by the software package management utilities
`/var/saf`	`saf` (service access facility) logging files and accounting files
`/var/samba`	Log files and lock files for Samba
`/var/snmp`	SNMP status and configuration information
`/var/spool`	Directories for spooled temporary files

TABLE 21–1 Default Directories in the root (/) File System *(Continued)*

Directory	Description
`/var/spool/clientmqueue`	Sendmail client files
`/var/spool/cron`	`cron` and `at` spool files
`/var/spool/locks`	Spooling lock files
`/var/spool/lp`	Line printer spool files
`/var/spool/mqueue`	Mail queued for delivery
`/var/spool/pkg`	Spooled packages
`/var/spool/print`	LP print service client-side request staging area
`/var/spool/samba`	Samba print queue
`/var/spool/uucp`	Queued `uucp` jobs
`/var/spool/uucppublic`	Files deposited by `uucp`
`/var/statmon`	Network status monitor files
`/var/tmp`	Directory for temporary files that are not cleared during boot sequence
`/var/uucp`	`uucp` log files and status files
`/var/yp`	NIS databases

The following table describes the default directories in the `/usr` file system.

TABLE 21–2 Default Directories in the `/usr` File System

Directory	Description
`4lib`	SunOS 4.1 binary compatibility package libraries
`5bin`	Symbolic link to the `/usr/bin` directory
`X`	Symbolic link to the `/usr/openwin` directory
`adm`	Symbolic link to the `/var/adm` directory
`apache`	Apache executables, loadable modules, and documentation
`aset`	Directory for Automated Security Enhancement Tools (ASET) programs and files
`bin`	Location for standard system commands
`ccs`	C compilation programs and libraries
`demo`	Demo programs and data

TABLE 21–2 Default Directories in the `/usr` File System *(Continued)*

Directory	Description
`dict`	Symbolic link to the `/usr/share/lib/dict` directory, which contains the dictionary file used by the UNIX spell program
`dt`	Directory or mount point for CDE software
`games`	An empty directory, which is a remnant of the SunOS 4.0-4.1 software
`include`	Header files for C programs, and so on.
`iplanet`	Directory server executables, loadable modules, and documentation
`j2se`	Java 2 SDK executables, loadable modules, and documentation
`java*`	Directories that contain Java programs and libraries
`kernel`	Additional kernel modules
`kvm`	Obsolete
`lib`	Various program libraries, architecture-dependent databases, and binaries not invoked directly by the user
`local`	Commands local to a site
`mail`	Symbolic link to the `/var/mail` directory
`man`	Symbolic link to the `/usr/share/man` directory
`net`	Directory for network listener services
`news`	Symbolic link to the `/var/news` directory
`oasys`	Files for the Form and Menu Language Interpreter (FMLI) execution environment
`old`	Programs that are being phased out
`openwin`	Directory or mount point for OpenWindows software
`perl5`	Perl 5 programs and documentation
`platform`	Supported platform files. For more information, see Table 21–3.
`preserve`	Symbolic link to the `/var/preserve` directory
`proc`	Directory for the `proc` tools
`pub`	Files for online man page and character processing

TABLE 21–2 Default Directories in the `/usr` File System *(Continued)*

Directory	Description
`sadm`	Various files and directories related to system administration
`sbin`	Executables for system administration
`sbin/install.d`	Custom JumpStart scripts and executables
`sbin/static`	Statically linked version of selected programs from `/usr/bin` and `/usr/sbin`
`sbin/sparcv7` and `sparcv9`	32-bit and 64-bit versions of commands on SPARC systems
`sbin/i86`	x86 architecture specific commands
`sfw`	GNU and open source executables, libraries, and documentation
`share`	Architecture-independent sharable files
`share/admserv5.1`	iPlanet Console and Administration Server 5.0 documentation
`share/audio`	Sample audio files
`share/ds5`	iPlanet Directory Server 5.1 Documentation
`share/lib`	Architecture-independent databases
`share/man`	Solaris manual pages
`share/src`	Source code for kernel, libraries, and utilities
`snadm`	Programs and libraries related to system and network administration
`spool`	Symbolic link to the `/var/spool` directory
`src`	Symbolic link to the `share/src` directory
`tmp`	Symbolic link to the `var/tmp` directory
`ucb`	Berkeley compatibility package binaries
`ucbinclude`	Berkeley compatibility package header files
`ucblib`	Berkeley compatibility package libraries
`vmsys`	Directory for Framed Access Command Environment (FACE) programs
`xpg4`	Directory for POSIX-compliant utilities

The Platform-Dependent Directories

The following table describes the platform-dependent objects in the `/platform` and `/usr/platform` directories.

TABLE 21–3 The `/platform` and `/usr/platform` Directories

Directory	Description
`/platform`	Contains a series of directories, one directory per supported platform that needs to reside in the root (`/`) file system.
`/platform/*/kernel`	Contains platform-dependent kernel components, including the file `unix`, the core kernel that is platform–dependent. For more information, see `kernel`(1M).
`/usr/platform`	Contains platform-dependent objects that do not need to reside in the root (`/`) file system.
`/usr/platform/*/lib`	Contains platform-dependent objects similar to those objects found in the `/usr/lib` directory.
`/usr/platform/*/sbin`	Contains platform-dependent objects similar to those objects found in the `/usr/sbin` directory.

The Structure of Cylinder Groups for UFS File Systems

When you create a UFS file system, the disk slice is divided into *cylinder groups*, which is made up of one or more consecutive disk cylinders. The cylinder groups are then further divided into addressable blocks to control and organize the structure of the files within the cylinder group. Each type of block has a specific function in the file system. A UFS file system has these four types of blocks:

Block Type	Type of Information Stored
Boot block	Information used when booting the system
Superblock	Detailed information about the file system
Inode	All information about a file

Block Type	Type of Information Stored
Storage or data block	Data for each file

The following sections provide additional information about the organization and function of these blocks.

The Boot Block

The boot block stores objects that are used in booting the system. If a file system is not to be used for booting, the boot block is left blank. The boot block appears only in the first cylinder group (cylinder group 0) and is the first 8 Kbytes in a slice.

The Superblock

The superblock stores much of the information about the file system, which includes the following:

- Size and status of the file system
- Label, which includes file system name and volume name
- Size of the file system logical block
- Date and time of the last update
- Cylinder group size
- Number of data blocks in a cylinder group
- Summary data block
- File system state
- Path name of the last mount point

Because the superblock contains critical data, multiple superblocks are made when the file system is created.

A summary information block is kept within the superblock. The summary information block is not replicated, but is grouped with the primary superblock, usually in cylinder group 0. The summary block records changes that take place as the file system is used. In addition, the summary block lists the number of inodes, directories, fragments, and storage blocks within the file system.

Inodes

An inode contains all the information about a file except its name, which is kept in a directory. An inode is 128 bytes. The inode information is kept in the cylinder information block, and contains the following:

- The type of the file:

    - Regular
    - Directory
    - Block special
    - Character special
    - FIFO, also known as named pipe
    - Symbolic link
    - Socket
    - Other inodes – attribute directory and shadow (used for ACLs)
- The mode of the file (the set of read-write-execute permissions)
- The number of hard links to the file
- The user ID of the owner of the file
- The group ID to which the file belongs
- The number of bytes in the file
- An array of 15 disk-block addresses
- The date and time the file was last accessed
- The date and time the file was last modified
- The date and time the file was created

The array of 15 disk addresses (0 to 14) points to the data blocks that store the contents of the file. The first 12 are direct addresses. That is, they point directly to the first 12 logical storage blocks of the file contents. If the file is larger than 12 logical blocks, the 13th address points to an indirect block, which contains direct block addresses instead of file contents. The 14th address points to a double indirect block, which contains addresses of indirect blocks. The 15th address is for triple indirect addresses. The following figure shows this chaining of address blocks starting from the inode.

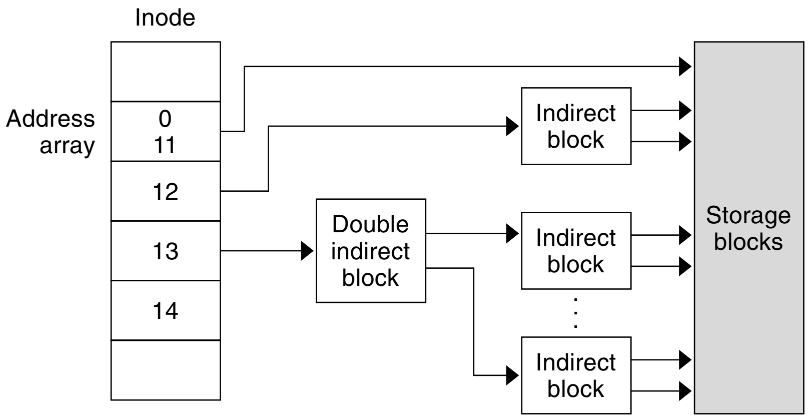

FIGURE 21–1 Address Chain for a UFS File System

Data Blocks

Data blocks, also called storage blocks, contain the rest of the space that is allocated to the file system. The size of these data blocks is determined at the time a file system is created. Data blocks are allocated, by default, in two sizes: an 8-Kbyte logical block size, and a 1-Kbyte fragment size.

For a regular file, the data blocks contain the contents of the file. For a directory, the data blocks contain entries that give the inode number and the file name of the files in the directory.

Free Blocks

Blocks that are not currently being used as inodes, as indirect address blocks, or as storage blocks are marked as free in the cylinder group map. This map also keeps track of fragments to prevent fragmentation from degrading disk performance.

To give you an idea of the appearance of a typical UFS file system, the following figure shows a series of cylinder groups in a generic UFS file system.

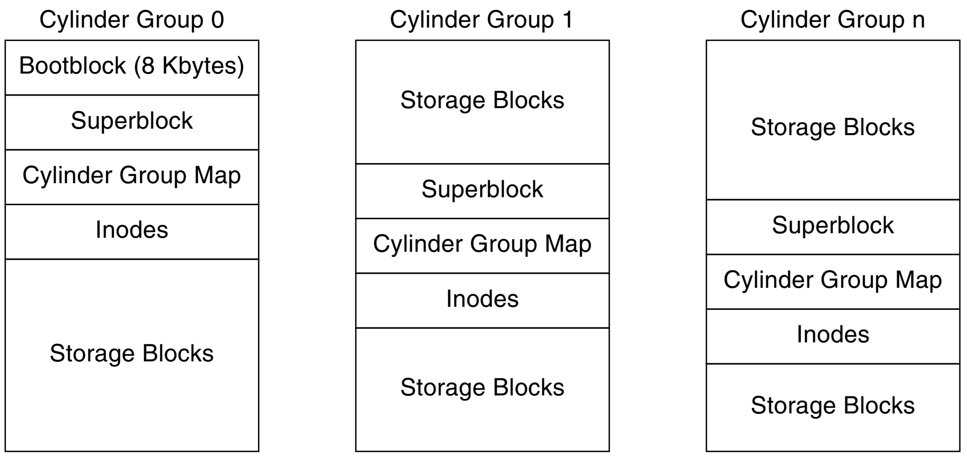

FIGURE 21–2 A Typical UFS File System

Custom File System Parameters

Before you choose to alter the default file system parameters that are assigned by the `newfs` command, you need to understand them. This section describes each of these parameters:

- Logical block size
- Fragment size
- Minimum free space
- Rotational delay
- Optimization type
- Number of files

Logical Block Size

The logical block size is the size of the blocks that the UNIX kernel uses to read or write files. The logical block size is usually different from the physical block size. The physical block size is usually 512 bytes, which is the size of the smallest block that the disk controller can read or write.

Logical block size is set to the page size of the system by default. The default logical block size is 8192 bytes (8 Kbytes) for UFS file systems. The UFS file system supports block sizes of 4096 or 8192 bytes (4 or 8 Kbytes). The recommended logical block size is 8 Kbytes.

SPARC only – You can specify only the 8192-byte block size on the sun4u platform.

To choose the best logical block size for your system, consider both the performance desired and the available space. For most UFS systems, an 8-Kbyte file system provides the best performance, offering a good balance between disk performance and the use of space in primary memory and on disk.

As a general rule, to increase efficiency, use a larger logical block size for file systems where most of the files are very large. Use a smaller logical block size for file systems where most of the files are very small. You can use the `quot -c` *file-system* command on a file system to display a complete report on the distribution of files by block size.

However, the page size set when the file system is created is probably best in most cases.

Fragment Size

As files are created or expanded, they are allocated disk space in either full logical blocks or portions of logical blocks called *fragments*. When disk space is needed for a file, full blocks are allocated first, and then one or more fragments of a block are allocated for the remainder. For small files, allocation begins with fragments.

The ability to allocate fragments of blocks to files, rather than just whole blocks, saves space by reducing *fragmentation* of disk space that results from unused holes in blocks.

You define the *fragment size* when you create a UFS file system. The default fragment size is 1 Kbyte. Each block can be divided into 1, 2, 4, or 8 fragments, which results in fragment sizes from 8192 bytes to 512 bytes (for 4-Kbyte file systems only). The lower bound is actually tied to the disk sector size, typically 512 bytes.

For multiterabyte file systems, the fragment size must be equal to the file system block size.

Note – The upper bound for the fragment is the logical block size, in which case the fragment is not a fragment at all. This configuration might be optimal for file systems with very large files when you are more concerned with speed than with space.

When choosing a fragment size, look at the trade-off between time and space: a small fragment size saves space, but requires more time to allocate. As a general rule, to increase storage efficiency, use a larger fragment size for file systems where most of the files are large. Use a smaller fragment size for file systems where most of the files are small.

Minimum Free Space

The *minimum free space* is the percentage of the total disk space that is held in reserve when you create the file system. The default reserve is ((64 Mbytes/partition size) * 100), rounded down to the nearest integer and limited between 1 percent and 10 percent, inclusively.

Free space is important because file access becomes less and less efficient as a file system gets full. As long as an adequate amount of free space exists, UFS file systems operate efficiently. When a file system becomes full, using up the available user space, only root can access the reserved free space.

Commands such as `df` report the percentage of space that is available to users, excluding the percentage allocated as the minimum free space. When the command reports that more than 100 percent of the disk space in the file system is in use, some of the reserve has been used by root.

If you impose quotas on users, the amount of space available to the users does not include the reserved free space. You can change the value of the minimum free space for an existing file system by using the `tunefs` command.

Rotational Delay

This parameter is obsolete. The value is always set to 0, regardless of the input value.

Optimization Type

The *optimization type* parameter is set to either *space* or *time*.

- **Space –** When you select space optimization, disk blocks are allocated to minimize fragmentation and disk use is optimized.
- **Time –** When you select time optimization, disk blocks are allocated as quickly as possible, with less emphasis on their placement. When there is enough free space, it is relatively easy to allocate disk blocks effectively, without resulting in too much fragmentation. The default is *time*.

  You can change the value of the optimization type parameter for an existing file system by using the `tunefs` command.

For more information, see `tunefs`(1M).

Number of Inodes (Files)

The number of bytes per inode specifies the density of inodes in the file system. The number is divided into the total size of the file system to determine the number of inodes to create. Once the inodes are allocated, you cannot change the number without re-creating the file system.

The default number of bytes per inode is 2048 bytes (2 Kbytes) if the file system is less than one Gbyte. If the file system is larger than one Gbyte, the following formula is used:

File System Size	Number of Bytes Per Inode
Less than or equal to 1 Gbyte	2048
Less than 2 Gbytes	4096
Less than 3 Gbytes	6144

File System Size	Number of Bytes Per Inode
3 Gbytes up to 1 Tbyte	8192
Greater than 1 Tbyte or created with `-T` option	1048576

If you have a file system with many symbolic links, they can lower the average file size. If your file system is going to have many small files, you can give this parameter a lower value. Note, however, that having too many inodes is much better than running out of inodes. If you have too few inodes, you could reach the maximum number of files on a disk slice that is practically empty.

Maximum UFS File and File System Size

The maximum size of a UFS file system is approximately 16 terabytes of usable space, minus approximately one percent overhead. A *sparse* file can have a logical size of one terabyte. However, the actual amount of data that can be stored in a file is approximately one percent less than one terabyte because of the file system overhead.

Maximum Number of UFS Subdirectories

The maximum number of subdirectories per directory in a UFS file system is 32,767. This limit is predefined and cannot be changed.

Commands for Creating a Customized File System

This section describes the two commands that you use to create a customized file system:

- `newfs`
- `mkfs`

The `newfs` Command Syntax, Options, and Arguments

The `newfs` command is a friendlier version of the `mkfs` command that is used to create file systems.

The syntax is as follows:

```
/usr/sbin/newfs [-NTv] [mkfs_options] raw_device
```

The following table describes the options and arguments for the `newfs` command.

TABLE 21–4 The `newfs` Command Options and Arguments

Option	Description
`-N`	Displays the file system parameters that would be used in creating the file system without actually creating it. This option does not display the parameters that were used to create an existing file system.
`-T`	Set the parameters of the file system to allow eventual growth to over a terabyte in total file system size. This option sets *fragsize* to be the same as *bsize*, and sets *nbpi* to 1 Mbyte, unless the `-i` option is used to make it even larger. If you use the `-f` or `-i` options to specify a *fragsize* or *nbpi* that is incompatible with this option, the user-supplied value of *fragsize* or *nbpi* is ignored.
`-v`	Displays the parameters that are passed to the `mkfs` command.
mkfs-options	Use the options in this table, from `-a apc` to `-t track`, to set the `mkfs` parameters. Separate the options with spaces.
`-a` *apc*	The number of alternate sectors per disk cylinder to reserve for bad block placement for SCSI devices only. The default is 0. This option is not applicable for disks with EFI labels and is ignored.
`-b` *bsize*	The logical block size of the file system, which is either 4096 or 8192 bytes. The default is 8192 bytes. The sun4u architecture does not support the 4096 block size.
`-c` *cgsize*	The number of cylinders per cylinder group, which ranges from 16 to 256. The default value is calculated by dividing the number of sectors in the file system by the number of sectors in 1 Gbyte. Then, the result is multiplied by 32. The default value is always between 16 to 256. Use the `mkfs` command to override the default value. This option is not applicable for disks with EFI labels and is ignored.

TABLE 21–4 The `newfs` Command Options and Arguments *(Continued)*

Option	Description
`-C` *maxcontig*	The maximum number of logical blocks, belonging to one file, that are allocated contiguously. The default is calculated as follows: `maxcontig` = *disk drive maximum transfer size* / *disk block size* If the disk drive maximum transfer size cannot be determined, the default value for `maxcontig` is calculated as follows: If `maxphys` is less than `ufs_maxmaxphys`, which is typically 1 Mbyte, then `maxcontig` is set to `maxphys`. Otherwise, `maxcontig` is set to `ufs_maxmaxphys`. You can set `maxcontig` to any positive integer value. The actual value will be the lesser of what has been specified and what the hardware supports. You can subsequently change this parameter by using the `tunefs` command. For more information, see`tunefs`(1M).
`-d` *gap*	Rotational delay. This option is obsolete. The value is always set to 0, regardless of the input value.
`-f` *fragsize*	The smallest amount of disk space in bytes that can be allocated to a file. *fragsize* must be a power of 2 divisor of *bsize*, where: *bsize* / *fragsize* is 1, 2, 4, or 8. This means that if the logical block size is 4096, legal values for *fragsize* are 512, 1024, 2048, and 4096. When the logical block size is 8192, legal values are 1024, 2048, 4096, and 8192. The default value is 1024. For file systems greater than 1 terabyte or for file systems created with the `-T` option, *fragsize* is forced to match block size (*bsize*).
`-i` *nbpi*	The number of bytes per inode, which specifies the density of inodes in the file system. The number is divided into the total size of the file system to determine the number of inodes to create. This value should reflect the expected average size of files in the file system. If fewer inodes are needed, specify a larger number. To create more inodes, specify a smaller number. For the default values, see "Number of Inodes (Files)" on page 363.

TABLE 21–4 The `newfs` Command Options and Arguments *(Continued)*

Option	Description
`-m` *free*	The minimum percentage of free space to maintain in the file system (between 0% and 99%, inclusively). This space is off–limits to regular users. Once the file system is filled to this threshold, only the superuser can continue writing to the file system. The default is ((64 Mbytes/partition size) * 100), rounded down to the nearest integer and limited between 1% and 10%, inclusively. This parameter can be modified after the file system is created by using the `tunefs` command.
`-n` *nrpos*	The number of different rotation positions in which to divide a cylinder group. The default is 8. This option is not applicable for disks with EFI labels and is ignored.
`-o` *opt*	Optimization type, `space` or `time`. The file system can either be instructed to try to minimize the *time* spent allocating blocks, or to try to minimize the *space* fragmentation on the disk. The default is `time`.
`-r` *rpm*	The rotational speed of the disk in revolutions per minute. This setting is driver- or device-specific. This parameter is converted to revolutions per second before it is passed to the `mkfs` command. This option is not applicable for disks with EFI labels and is ignored.
`-s` *size*	The size of the file system in sectors. The default is to use the entire partition.
`-t` *ntrack*	The number of tracks per cylinder on the disk. The default is taken from the disk label. This option is not applicable for disks with EFI labels and is ignored.
raw_device	The name of a raw special device residing in the `/dev` directory (for example, `/dev/rdsk/c0t0d0s6`) on which to create the file system. This argument is required.

EXAMPLE 21–1 `newfs` Command Options and Arguments

This example shows how to use the `-N` option to display file system information, including the backup superblocks.

```
# newfs -N /dev/rdsk/c0t0d0s0
/dev/rdsk/c0t0d0s0:  37260 sectors in 115 cylinders of 9 tracks, 36 sectors
        19.1MB in 8 cyl groups (16 c/g, 2.65MB/g, 1216 i/g)
superblock backups (for fsck -b #) at:
```

EXAMPLE 21–1 `newfs` Command Options and Arguments *(Continued)*

```
 32, 5264, 10496, 15728, 20960, 26192, 31424, 36656,
#
```

The Generic `mkfs` Command

The generic `mkfs` command calls a file system-specific `mkfs` command, which then creates a file system of a specified type on a specified disk slice. Although the `mkfs` command can support different types of file systems, in practice you would use it to create UFS, UDFS, or PCFS file systems. To make other types of file systems, you would have to write the software for the file system–specific versions of the `mkfs` command to use. Normally, you do not run the `mkfs` command directly. The `mkfs` command is called by the `newfs` command.

The generic `mkfs` command is located in the `/usr/sbin` directory. For a description of the arguments and options, see `mkfs`(1M).

CHAPTER 22

Backing Up and Restoring File Systems (Overview)

This chapter provides guidelines and planning information for backing up and restoring file systems by using the `ufsdump` and `ufsrestore` commands.

This is a list of the overview information in this chapter.

Where to Find Backup and Restore Tasks

Backup or Restore Task	For More Information
Back up file systems by using the `ufsdump` command	Chapter 23
Create UFS snapshots by using the `fssnap` command	Chapter 24
Restore file systems by using the `ufsrestore` command	Chapter 25

Backup or Restore Task	For More Information
Copy files and directories by using the `cpio`, `dd`, `pax`, and `cpio` commands	Chapter 27

Definition: Backing Up and Restoring File Systems

Backing up file systems means copying file systems to removable media, such as tape, to safeguard against loss, damage, or corruption. *Restoring* file systems means copying reasonably current backup files from removable media to a working directory.

This chapter describes the `ufsdump` and `ufsrestore` commands for backing up and restoring UFS file systems. Other commands are available for copying files and file systems for the purpose of sharing or transporting files. The following table provides pointers to all commands that copy individual files and file systems to other media.

TABLE 22–1 Commands for Backing Up and Restoring Files and File Systems

Task	Command	For More Information
Back up one or more file systems to a local tape device or a remote tape device	`ufsdump`	Chapter 23 or Chapter 26
Create read-only copies of file systems	`fssnap`	Chapter 24
Back up all file systems for systems on a network from a backup server	Solstice Backup software	*Solstice Backup 5.1 Administration Guide*
Back up and restore an NIS+ master server	`nisbackup` and `nisrestore`	*System Administration Guide: Naming and Directory Services (FNS and NIS+)*
Copy, list, and retrieve files on tape or diskette	`tar`, `cpio`, or `pax`	Chapter 27
Copy master disk to a clone disk	`dd`	Chapter 27
Restore complete file systems or individual files from removable media to a working directory	`ufsrestore`	Chapter 25

Why You Should Back Up File Systems

Backing up files is one of the most crucial system administration functions. You should perform regularly scheduled backups to prevent loss of data due to the following types of problems:

- System crashes
- Accidental deletion of files
- Hardware failures
- Natural disasters such as fire, hurricanes, or earthquakes
- Problems when you reinstall or upgrade a system

Planning Which File Systems to Back Up

You should back up all file systems that are critical to users, including file systems that change frequently. The following tables provide general guidelines on the file systems to back up for standalone systems and servers.

TABLE 22–2 File Systems to Back Up for Standalone Systems

File System to Back Up	Description	Back Up Interval
root (/) – slice 0	This file system contains the kernel and possibly the `/var` directory. The `/var` directory might include frequently modified files such as mail and accounting files.	At regular intervals such as weekly or daily
`/usr` – slice 6, `/opt`	The installation of new software and new commands typically affects the `/usr` and `/opt` file systems. The `/opt` directory is either part of root (/) or is its own file system.	Occasionally
`/export/home` – slice 7	This file system contains the directories and subdirectories of all users on the standalone system.	More often than root (/) or `/usr`, perhaps as often as once a day, depending on your site's needs

TABLE 22–2 File Systems to Back Up for Standalone Systems *(Continued)*

File System to Back Up	Description	Back Up Interval
`/export` , `/var`, or other file systems	During installation of Solaris software, you might have created these file systems.	As your site requires

TABLE 22–3 File Systems to Back Up for Servers

File System to Back Up	Description	Back Up Interval
root (/) – slice 0	This file system contains the kernel and executables.	Once a day to once a month depending on your site's needs. If you frequently add and remove users and systems on the network, you have to change configuration files in this file system. In this case, you should do a full backup of the root (/) file system at intervals between once a week and once a month. If your site keeps user mail in the `/var/mail` directory on a mail server, which client systems then mount, you might want to back up root (/) daily. Or, backup the `/var` directory, if it is a separate file system.
`/export` – slice 3	This file system can contain the kernel and executables for diskless clients.	Once a day to once a month, depending on your site's needs. Because the information in this file system is similar to the server's root directory in slice 0, the file system does not change frequently. You need to back up this file system only occasionally, unless your site delivers mail to client systems. Then, you should back up `/export` more frequently.
`/usr` – slice 6, `/opt`	An optional file system generally used to store non-system software.	Once a day to once a month, depending on your site's needs. These file systems are fairly static and need to be backed up once a week to once a month.

TABLE 22–3 File Systems to Back Up for Servers *(Continued)*

File System to Back Up	Description	Back Up Interval
`/export/home` – slice 7	This file system contains the home directories of all the users on the system. The files in this file system are volatile.	Once a day to once a week.

Choosing the Type of Backup

You can perform full or incremental backups by using the `ufsdump` command. You can create a temporary image of a file system by using the `fssnap` command. The following table lists the differences between these types of backup procedures.

TABLE 22–4 Differences Between Types of Backups

Backup Type	Result	Advantages	Disadvantages
Full	Copies a complete file system or directory	All data is in one place	Requires large numbers of backup tapes that take a long time to write. Takes longer to retrieve individual files because the drive has to move sequentially to the point on the tape where the file is located. You might have to search multiple tapes.
Snapshot	Creates a temporary image of a file system	System can be in multiuser mode	System performance might degrade while the snapshot is created.
Incremental	Copies only those files in the specified file system that have changed since a previous backup	Easier to retrieve small changes in file systems	Finding which incremental tape contains a file can take time. You might have to go back to the last full backup.

Choosing a Tape Device

The following table shows typical tape devices that are used for storing file systems during the backup process. The storage capacity depends on the type of drive and the data being written to the tape. For more information on tape devices, see Chapter 28.

TABLE 22–5 Typical Media for Backing Up File Systems

Backup Media	Storage Capacity
1/2-inch reel tape	140 Mbytes (6250 bpi)
2.5-Gbyte 1/4–inch cartridge (QIC) tape	2.5 Gbytes
DDS3 4-mm cartridge tape (DAT)	12–24 Gbytes
14-Gbyte 8-mm cartridge tape	14 Gbytes
DLT 7000 1/2-inch cartridge tape	35–70 Gbytes

High-Level View of Backing Up and Restoring File Systems (Task Map)

Use this task map to identify all the tasks for backing up and restoring file systems. Each task points to a series of additional tasks, such as determining the type of backup to perform.

Task	Description	For Instructions
1. Identify the file systems to back up	Identify which file systems need to be backed up on a daily, weekly, or monthly basis.	"Planning Which File Systems to Back Up" on page 371
2. Determine the type of backup	Determine the type of backup you need for the file systems at your site.	"Choosing the Type of Backup" on page 373
3. Create the backup	Use one of the following methods:	
	If you want to have full and incremental backups of your file systems, use the `ufsdump` command.	Chapter 23

Task	Description	For Instructions
	If you want to create a snapshot of file system while it is active and mounted, consider using the `fssnap` command.	Chapter 24
	If you want to create a snapshot of file system while it is active and mounted, consider using the `fssnap` command.	Chapter 27
4. (Optional) Restore a file system	Select the restoration method that is based on the command used to back up the files or file system:	
	Restore a file system backup that was created with the `ufsdump` command.	Chapter 25
	Restore a file system that was created with the `tar`, `cpio`, or `pax` command.	Chapter 27
5. (Optional) Restore the root (`/`) or `/usr` file system	Restoring the root (`/`) or `/usr` file system is more complicated than restoring a non critical file system. You need to boot from a local CD or from the network while these file systems are being restored.	"How to Restore the root (`/`) and `/usr` File Systems" on page 418

Guidelines for Scheduling Backups

A *backup schedule* is the schedule that you establish to run the `ufsdump` command. This section discusses guidelines on the factors to weigh when you create a backup schedule. This section also includes sample backup schedules.

The backup schedule that you create depends on the following:

- Your need to minimize the number of tapes that are used for backups
- The time available for doing backups
- The time available for doing a full restore of a damaged file system
- The time available for retrieving individual files that are accidentally deleted

How Often Should You Do Backups?

If you do not need to minimize time and the number of media that is used for backups, you can do full backups every day. However, this backup method is not realistic for most sites, so incremental backups are used most often. In this case, you should back up your site enough to so that you can restore files from the last four weeks. This schedule requires at least four sets of tapes, one set for each week. You would then reuse the tapes each month. In addition, you should archive the monthly backups for at least a year. Then, keep yearly backups for a number of years.

Backup Terms and Definitions

The following table describes backup terms and definitions.

Term	Definition
Snapshot	Creates a temporary image of a file system.
Full backup	Copies a complete file system or directory.
Incremental backup	Copies only those files in the specified file system that have changed since a previous backup. Incremental backup types include the following: ■ Daily, cumulative – Copies a day's worth of file changes on Monday. Then, overwrites Monday's backup with file changes from Tuesday, Wednesday, and so on. ■ Daily, incremental – Copies a day's worth of file changes so that you have distinct tapes of Monday's changes, Tuesday's changes, and so on. ■ Weekly cumulative – Copies the files that have changed during the week and includes the previous week's file changes. ■ Weekly incremental – Copies the files that have changed during the week since the previous weekly backup.

Suggestions for Scheduling Backups

The following table provides other suggestions for scheduling backups.

TABLE 22–6 Suggestions for Backup Schedules

File Restoration Need	Backup Interval	Comments
To restore different versions of files (for example, file systems that are used for word processing)	Do daily incremental backups every working day Do *not* reuse the same tape for daily incremental backups	This schedule saves all files modified that day, as well as those files still on disk that were modified since the last backup of a lower level. However, with this schedule, you should use a different tape each day because you might otherwise be unable to restore the needed version of the file. For example, a file that changed on Tuesday, and again on Thursday, goes onto Friday's lower-level backup appearing as it did Thursday night, not Tuesday night. If a user needs the Tuesday version, you cannot restore it unless you have a Tuesday backup tape (or a Wednesday backup tape). Similarly, a file that is present on Tuesday and Wednesday, but removed on Thursday, does not appear on the Friday lower-level backup.
To quickly restore a complete file system	Do lower-level backups more frequently.	—
To back up a number of file systems on the same server	Consider staggering the schedule for different file systems.	This way you're not doing all level 0 backups on the same day.
To minimize tapes	Increase the level of incremental backups that are done across the week.	Only changes from day to day are saved on each daily tape.
	Increase the level of backups that are done at the end of the week. Put each day's and week's incremental backups onto the same tape.	Only changes from week to week (rather than the entire month) are saved on the weekly tapes.
	Put each day's and week's incremental backups onto the same tape.	To do so, use the no rewind option of the `ufsdump` command, such as specifying `/dev/rmt/0n`.

Using Dump Levels to Create Incremental Backups

The dump level you specify in the `ufsdump` command (0–9) determines which files are backed up. Dump level 0 creates a full backup. Levels 1–9 are used to schedule incremental backups, but have *no defined meanings*. Levels 1–9 are just a range of numbers that are used to schedule cumulative or discrete backups. The only meaning levels 1–9 have is in relationship to each other, as a higher or lower number. A lower dump number always restarts a full or a cumulative backup. The following examples show the flexibility of the incremental dump procedure using levels 1–9.

Example—Dump Levels for Daily, Cumulative Backups

Doing daily, cumulative incremental backups is the most commonly used backup schedule and is recommended for most situations. The following example shows a schedule that uses a level 9 dump Monday through Thursday, and a level 5 dump on Friday to restart the process.

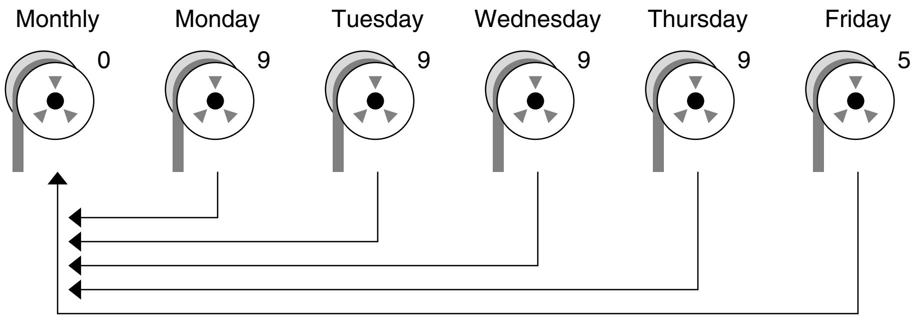

FIGURE 22–1 Incremental Backup: Daily Cumulative

In the preceding example, you could have used other numbers in the 1–9 range to produce the same results. The key is using the same number Monday through Thursday, with any *lower* number on Friday. For example, you could have specified levels 4, 4, 4, 4, 2 or 7, 7, 7, 7, 5.

Example—Dump Levels for Daily, Incremental Backups

The following example shows a schedule where you capture only a day’s work on different tapes. This type of backup is referred to as a daily, incremental backup. In this case, sequential dump level numbers are used during the week (3, 4, 5, 6) with a lower number (2) on Friday. The lower number on Friday restarts the processing.

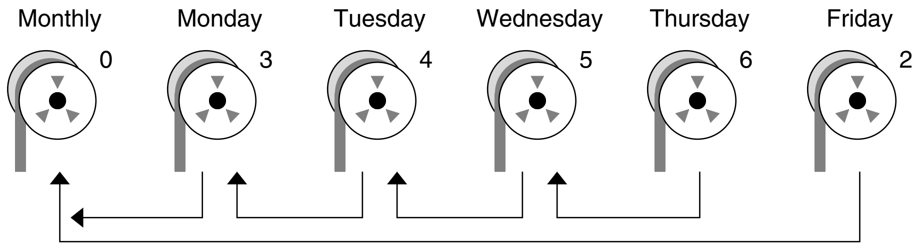

FIGURE 22–2 Incremental Backup: Daily Incremental

In the preceding example, you could have used the sequence 6, 7, 8, 9 followed by 2, or 5, 6, 7, 8 followed by 3. Remember, the numbers themselves have no defined meaning. You attribute meaning by ordering them in a specified sequence, as described in the examples.

Sample Backup Schedules

This section provides sample backup schedules. All schedules assume that you begin with a full backup (dump level 0), and that you use the `-u` option to record each backup in the `/etc/dumpdates` file.

Example—Daily Cumulative, Weekly Cumulative Backups

Table 22–7 shows the most commonly used incremental backup schedule. This schedule is recommended for most situations. With this schedule, the following occurs:

- All files that have changed since the lower-level backup at the end of the previous week are saved each day.
- For each weekday level 9 backup, the previous level 0 or level 5 backup is the closest backup at a lower level. Therefore, each weekday tape contains all the files that changed since the end of the previous week or the initial level 0 backup for the first week.
- For each Friday level 5 backup, the closest lower-level backup is the level 0 backup done at the beginning of the month. Therefore, each Friday's tape contains all the files changed during the month up to that point.

TABLE 22–7 Daily Cumulative or Weekly Cumulative Backup Schedule

	Floating	Mon	Tues	Wed	Thurs	Fri
1st of Month	0					
Week 1		9	9	9	9	5
Week 2		9	9	9	9	5
Week 3		9	9	9	9	5
Week 4		9	9	9	9	5

The following table shows how the contents of the tapes can change across two weeks with the daily cumulative, weekly cumulative schedule. Each letter represents a different file.

TABLE 22–8 Contents of Tapes for Daily Cumulative/Weekly Cumulative Backup Schedule

	Mon	Tues	Wed	Thurs	Fri
Week 1	a b	a b c	a b c d	a b c d e	a b c d e f
Week 2	g	g h	g h i	g h i j	a b c d e f g h i j k

Tape Requirements for the Daily Cumulative, Weekly Cumulative Schedule

With this schedule, you need six tapes if you want to reuse daily tapes. However, you need nine tapes if you want to use four different daily tapes:

- One tape for the level 0 backup
- Four tapes for Fridays
- One or four daily tapes

If you need to restore a complete file system, you need the following tapes:

- The level 0 tape
- The most recent Friday tape
- The most recent daily tape since the last Friday tape, if any

Example—Daily Cumulative, Weekly Incremental Backups

The following table shows a schedule where each weekday tape accumulates all files that changed since the beginning of the week, or the initial level 0 backup for the first week. In addition, each Friday's tape contains all the files that changed that week.

TABLE 22–9 Daily Cumulative, Weekly Incremental Backup Schedule

	Floating	Mon	Tues	Wed	Thurs	Fri
1st of Month	0					
Week 1		9	9	9	9	3
Week 2		9	9	9	9	4
Week 3		9	9	9	9	5
Week 4		9	9	9	9	6

The following table shows how the contents of the tapes can change across two weeks with the daily cumulative, weekly incremental backup schedule. Each letter represents a different file.

TABLE 22–10 Contents of Tapes for Daily Cumulative, Weekly Incremental Backup Schedule

	Mon	Tues	Wed	Thurs	Fri
Week 1	a b	a b c	a b c d	a b c d e	a b c d e f
Week 2	g	g h	g h i	g h i j	g h i j k

Tape Requirements for the Daily Cumulative, Weekly Incremental Backup Schedule

With this schedule, you need six tapes if you want to reuse daily tapes. However, you need nine tapes if you want to use four different daily tapes:

- One tape for the level 0 backup
- Four tapes for Fridays
- One or four daily tapes

If you need to restore a complete file system, you need the following tapes:

- The level 0 tape
- All the Friday tapes
- The most recent daily tape since the last Friday tape, if any

Example—Daily Incremental, Weekly Cumulative Backups

The following table shows a schedule where each weekday tape contains only the files that changed since the previous day. In addition, each Friday's tape contains all files changed since the initial level 0 backup at the beginning of the month.

TABLE 22–11 Daily Incremental, Weekly Cumulative Backup Schedule

	Floating	Mon	Tues	Wed	Thurs	Fri
1st of Month	0					
Week 1		3	4	5	6	2
Week 2		3	4	5	6	2
Week 3		3	4	5	6	2
Week 4		3	4	5	6	2

The following table shows how the contents of the tapes can change across two weeks with the daily incremental, weekly cumulative schedule. Each letter represents a different file.

TABLE 22–12 Contents of Tapes for Daily Incremental, Weekly Cumulative Backup Schedule

	Mon	Tues	Wed	Thurs	Fri
Week 1	a b	c d	e f g	hi	a b c d e f g h i
Week 2	j k l	m	n o	p q	a b c d e f g h i j k l m n o p q r s

Tape Requirements for Daily Incremental, Weekly Cumulative Schedule

With this schedule, you need at least 9 tapes if you want to reuse daily tapes, which is not recommended. Preferably, you need 21 tapes if you save weekly tapes for a month: one tape for the level 0, four tapes for the Fridays, and four or 16 daily tapes.

- 1 tape for the level 0 backup.
- 4 tapes for all the Friday backups.
- 4 or 16 daily tapes.

If you need to restore the complete file system, you need the following tapes:

- The level 0 tape.
- The most recent Friday tape.
- All the daily tapes since the last Friday tape, if any.

Example—Monthly Backup Schedule for a Server

The following table shows an example backup strategy for a heavily used file server on a small network where users are doing file-intensive work, such as program development or document production. This example assumes that the backup period begins on a Sunday and consists of four seven-day weeks.

TABLE 22–13 Example of Monthly Backup Schedule for a Server

Directory	Date	Dump Level	Tape Name
root (/)	1st Sunday	0	*n* tapes
`/usr`	1st Sunday	0	*n* tapes
`/export`	1st Sunday	0	*n* tapes
`/export/home`	1st Sunday	0	*n* tapes
	1st Monday	9	A
	1st Tuesday	9	B
	1st Wednesday	5	C
	1st Thursday	9	D
	1st Friday	9	E
	1st Saturday	5	F
root (/)	2nd Sunday	0	*n* tapes
`/usr`	2nd Sunday	0	*n* tapes
`/export`	2nd Sunday	0	*n* tapes
`/export/home`	2nd Sunday	0	*n* tapes
	2nd Monday	9	G
	2nd Tuesday	9	H
	2nd Wednesday	5	I
	2nd Thursday	9	J
	2nd Friday	9	K
	2nd Saturday	5	L
root (/)	3rd Sunday	0	*n* tapes
`/usr`	3rd Sunday	0	*n* tapes
`/export`	3rd Sunday	0	*n* tapes
`/export/home`	3rd Sunday	0	*n* tapes
	3rd Monday	9	M
	3rd Tuesday	9	N
	3rd Wednesday	5	O
	3rd Thursday	9	P

TABLE 22–13 Example of Monthly Backup Schedule for a Server *(Continued)*

Directory	Date	Dump Level	Tape Name
	3rd Friday	9	Q
	3rd Saturday	5	R
root (/)	4th Sunday	0	*n* tapes
`/usr`	4th Sunday	0	*n* tapes
`/export`	4th Sunday	0	*n* tapes
`/export/home`	4th Sunday	0	*n* tapes
	4th Monday	9	S
	4th Tuesday	9	T
	4th Wednesday	5	U
	4th Thursday	9	V
	4th Friday	9	W
	4th Saturday	5	X

With this schedule, you use 4*n* tapes, the number of tapes needed for 4 full backups of the root (/), `/usr`, `/export`, and `/export/home` file systems. Also, you need 24 additional tapes for the incremental backups of the `/export/home` file systems. This schedule assumes that each incremental backup uses one tape and that you save the tapes for a month.

Here's how this schedule works:

1. On each Sunday, do a full backup (level 0) of the root (/), `/usr`, `/export`, and `/export/home` file systems. Save the level 0 tapes for at least three months.
2. On the first Monday of the month, use tape A to do a level 9 backup of the `/export/home` file system. The `ufsdump` command copies all files changed since the previous lower-level backup. In this case, the previous lower-level backup is the level 0 backup that you did on Sunday.
3. On the first Tuesday of the month, use tape B to do a level 9 backup of the `/export/home` file system. Again, the `ufsdump` command copies all files changed since the last lower-level backup, which is Sunday's level 0 backup.
4. On the first Wednesday of the month, use tape C to do a level 5 backup of the `/export/home` file system. The `ufsdump` command copies all files that changed since Sunday.
5. Do the Thursday and Friday level 9 backups of the `/export/home` file system on tapes D and E. The `ufsdump` command copies all files that changed since the last lower-level backup, which is Wednesday's level 5 backup.

6. On the first Saturday of the month, use tape F to do a level 5 backup of `/export/home`. The `ufsdump` command copies all files changed since the previous lower-level backup (in this case, the level 0 backup you did on Sunday). Store tapes A-F until the first Monday of the next four-week period, when you use them again.
7. Repeat steps 1–6 for the next three weeks, using tapes G–L and 4*n* tapes for the level 0 backup on Sunday, and so on.
8. For each four-week period, repeat steps 1–7, using a new set of tapes for the level 0 backups and reusing tapes A–X for the incremental backups. The level 0 tapes could be reused after three months.

   This schedule lets you save files in their various states for a month. This plan requires many tapes, but ensures that you have a library of tapes to draw upon. To reduce the number of tapes, you could reuse Tapes A–F each week.

CHAPTER 23

Backing Up Files and File Systems (Tasks)

This chapter describes the procedures for backing up file systems by using the `ufsdump` command.

For information on these procedures, see "Backing Up Files and File System (Task Map)" on page 387.

For overview information about performing backups, see Chapter 22.

For detailed information on the `ufsdump` command syntax, options, and arguments, see Chapter 26.

Backing Up Files and File System (Task Map)

Task	Description	For Instructions
1. Prepare for file system backups	Identify the file systems, the type of backup, and the tape device to be used for the backups.	"Preparing for File System Backups" on page 388
2. Determine the number of tapes needed to back up a file system	Determine the number of tapes that are needed for a full backup of a file system.	"How to Determine the Number of Tapes Needed for a Full Backup" on page 389

Task	Description	For Instructions
3. Back up file systems	Perform a full backup of file systems to get baseline copies of all files. Perform an incremental backup of file systems based on whether keeping copies of files that have changed on a daily basis is important at your site.	"How to Backup a File System to Tape" on page 390

Preparing for File System Backups

The preparation for backing up file systems begins with planning, which is described in Chapter 22 and includes choosing the following:

- The file systems to back up
- The type of backup (full or incremental) to perform
- A backup schedule
- A tape drive

For more information, see Chapter 22.

This section describes two other tasks you might need to perform before you back up file systems:

- Finding the names of file systems to back up
- Determining the number of tapes that are needed for a full backup

▼ How to Find File System Names

Steps

1. **Display the contents of the `/etc/vfstab` file.**

   ```
   $ more /etc/vfstab
   ```

2. **Look in the `mount point` column for the name of the file system.**

3. **Use the directory name listed in the `mount point` column when you back up the file system.**

Example 23–1 Finding File System Names

In this example, the file systems to be backed up are root (`/`), `/usr`, `/datab`, and `/export/home`.

```
$ more /etc/vfstab
#device           device              mount         FS    fsck mount   mount
```

```
#to mount         to fsck            point        type pass at boot options
#
fd                -                  /dev/fd      fd    -    no      -
/proc             -                  /proc        proc  -    no      -
/dev/dsk/c0t0d0s1 -                  -            swap  -    no      -
/dev/dsk/c0t0d0s0 /dev/rdsk/c0t0d0s0 /            ufs   1    no      -
/dev/dsk/c0t0d0s6 /dev/rdsk/c0t0d0s6 /usr         ufs   1    no      -
/dev/dsk/c0t0d0s5 /dev/rdsk/c0t0d0s5 /datab       ufs   2    yes     -
/dev/dsk/c0t0d0s7 /dev/rdsk/c0t0d0s7 /export/home ufs   2    yes     -
swap              -                  /tmp         tmpfs -    yes     -
```

▼ How to Determine the Number of Tapes Needed for a Full Backup

Steps

1. **Become superuser or assume an equivalent role.**
2. **Estimate the size of the backup in bytes.**

   ```
   # ufsdump S file-system
   ```

   The S option displays the estimated number of bytes that are needed to do the backup.
3. **Divide the estimated size by the capacity of the tape to see how many tapes you need.**

   For a list of tape capacities, see Table 22–5.

Example 23–2 Determining Number of Tapes

In this example, the file system of 489,472 bytes easily fits on a 150-Mbyte tape.

```
# ufsdump S /export/home
489472
```

Backing Up a File System

The following are general guidelines for performing backups:

- Use single-user mode or unmount the file system, unless you are creating a snapshot of a file system. For information about UFS snapshots, see Chapter 24.
- Be aware that backing up file systems when directory-level operations (such as creating, removing, and renaming files) and file-level activity are occurring simultaneously means that some data will not be included in the backup.

- You can run the `ufsdump` command from a single system and remotely back up groups of systems across the network through remote shell or remote login. In addition, you can direct the output to the system on which the tape device is located. Typically, the tape device is located on the system from which you run the `ufsdump` command, but it does not have to be.

  Another way to back up files to a remote device is to pipe the output from the `ufsdump` command to the `dd` command. For information about using the `dd` command, see Chapter 27.
- If you are doing remote backups across the network, the system with the tape device must have entries in its `/.rhosts` file for each client that will be using the drive. Also, the system that initiates the backup must be included in the `/.rhosts` file on each system that it will back up.
- To specify a remote tape device on a system, use the naming convention that matches the OS release of the system with the remote tape device. For example, use the `/dev/rst0` device for a remote device on a system that is running the SunOS 4.1.1 release or compatible versions. Use the `/dev/rmt/0` device for a system running the Solaris 9 release or compatible versions.

Note – Use the `nisbackup` command to back up an NIS+ master server. For information on using this command, see *System Administration Guide: Naming and Directory Services (FNS and NIS+)*.

▼ How to Backup a File System to Tape

The following are general steps for backing up file systems by using the `ufsdump` command. The examples show specific uses of options and arguments.

Steps

1. **Become superuser or assume an equivalent role.**
2. **Bring the system to run level S (single-user mode).**

   For example:

   ```
   # shutdown -g30 -y
   ```
3. **(Optional) Check the file system for consistency.**

   For example:

   ```
   # fsck -m /dev/rdsk/c0t0d0s7
   ```

   The `fsck -m` command checks for the consistency of file systems. For example, power failures can leave files in an inconsistent state. For more information on the `fsck` command, see Chapter 20.
4. **If you need to back up file systems to a remote tape drive, follow these steps:**

   a. **On the system to which the tape drive is attached (the tape server), add the following entry to its `/.rhosts` file.**

```
host root
```

The *host* entry specifies the name of the system on which you will run the `ufsdump` command to perform the backup.

b. **On the tape server, verify that the host added to the `/.rhosts` file is accessible through the name service.**

5. **Identify the device name of the tape drive.**

   The default tape drive is the `/dev/rmt/0` device.

6. **Insert a tape that is write-enabled into the tape drive.**

7. **Back up file systems.**

   ```
   # ufsdump options arguments filenames
   ```

   You can back up file systems or directories or files within file systems. For information on backing up individual files, see `tar`(1) or `cpio`(1).

   The following examples show how to use the most common `ufsdump` options and arguments:

   - Example 23–3
   - Example 23–4
   - Example 23–5
   - Example 23–6

   For other `ufsdump` options and arguments, see Chapter 26.

8. **If prompted, remove the tape and insert the next tape volume.**

9. **Label each tape with the volume number, dump level, date, system name, disk slice, and file system.**

10. **Bring the system back to run level 3 by pressing Control-D.**

11. **Verify that the backup was successful.**

    ```
    # ufsrestore tf device-name
    ```

Example 23–3

Performing a Full Backup of root (/)

The following example shows how to do a full backup of the root (/) file system. The system in this example is brought to single-user mode before the backup. The following `ufsdump` options are included:

- `0` specifies a `0` level dump (or a full backup).
- `u` specifies that the `/etc/dumpdates` file is updated with the date of this backup.
- `c` identifies a cartridge tape device.
- `f /dev/rmt/0` identifies the tape device.
- `/` is the file system being backed up.

For example:

```
# init 0
ok boot -s
# ufsdump 0ucf /dev/rmt/0 /
  DUMP: Date of this level 0 dump: Tue Oct 07 16:23:08 2003
  DUMP: Date of last level 0 dump: the epoch
  DUMP: Dumping /dev/rdsk/c0t0d0s0 (starbug:/) to /dev/rmt/0.
  DUMP: Mapping (Pass I) [regular files]
  DUMP: Mapping (Pass II) [directories]
  DUMP: Writing 63 Kilobyte records
  DUMP: Estimated 296644 blocks (144.85MB).
  DUMP: Dumping (Pass III) [directories]
  DUMP: Dumping (Pass IV) [regular files]
  DUMP: Tape rewinding
  DUMP: 296224 blocks (144.64MB) on 1 volume at 424 KB/sec
  DUMP: DUMP IS DONE
  DUMP: Level 0 dump on Tue Oct 07 16:23:08 2003
  # ufsrestore tf /dev/rmt/0
         2      .
         3      ./lost+found
      3776      ./usr
      7552      ./var
     11328      ./export
     15104      ./export/home
     18880      ./etc
     22656      ./etc/default
     22657      ./etc/default/sys-suspend
     22673      ./etc/default/cron
     22674      ./etc/default/devfsadm
     22675      ./etc/default/dhcpagent
     22676      ./etc/default/fs
     22677      ./etc/default/inetinit
     22678      ./etc/default/kbd
     22679      ./etc/default/mpathd
     22680      ./etc/default/nfslogd
     22681      ./etc/default/passwd
                .
                .
                .
  # (Press Control-D to bring system to run level 3)
```

Example 23–4 Performing an Incremental Backup of root (/)

The following example shows how to do an incremental backup of the root (/) file system in single-user mode. The following `ufsdump` options are included:

- `9` specifies a `9` level dump (or an incremental backup).
- `u` specifies that the `/etc/dumpdates` file is updated with the date of this backup.
- `c` identifies a cartridge tape device.
- `f /dev/rmt/0` identifies the tape device.
- `/` is the file system being backed up.

```
# init 0
ok boot -s
```

```
.
.
.
Rebooting with command: boot -sSunOS Release 5.9  Generic May 2002
Copyright 1983-2003 Sun Microsystems, Inc.  All rights reserved.
.
.
.
# ufsdump 9ucf /dev/rmt/0 /
 DUMP: Date of this level 9 dump: Mon Oct 06 12:36:10 2003
 DUMP: Date of last level 0 dump: Wed Oct 08 10:12:13 2003
 DUMP: Dumping /dev/rdsk/c0t0d0s0 (starbug:/) to /dev/rmt/0.
 DUMP: Mapping (Pass I) [regular files]
 DUMP: Mapping (Pass II) [directories]
 DUMP: Writing 63 Kilobyte records
 DUMP: Estimated 335844 blocks (163.99MB).
 DUMP: Dumping (Pass III) [directories]
 DUMP: Dumping (Pass IV) [regular files]
 DUMP: 335410 blocks (163.77MB) on 1 volume at 893 KB/sec
 DUMP: DUMP IS DONE
 DUMP: Level 9 dump on Mon Oct 06 12:36:10 2003
 # ufsrestore tf /dev/rmt/0
         2      .
         3      ./lost+found
      5696      ./usr
     11392      ./var
     17088      ./export
     22784      ./export/home
     28480      ./opt
      5697      ./etc
     11393      ./etc/default
     11394      ./etc/default/sys-suspend
     11429      ./etc/default/cron
     11430      ./etc/default/devfsadm
     11431      ./etc/default/dhcpagent
     11432      ./etc/default/fs
     11433      ./etc/default/inetinit
     11434      ./etc/default/kbd
     11435      ./etc/default/nfslogd
     11436      ./etc/default/passwd
     11437      ./etc/default/tar
                .
                .
                .
```

Example 23–5 Performing a Full Backup of a Home Directory

The following example shows how to do a full backup of the `/export/home/kryten` home directory. The following `ufsdump` options are included:

- `0` specifies that this is a `0` level dump (or a full backup)
- `u` specifies that the `/etc/dumpdates` file is updated with the date of this backup

- `c` identifies a cartridge tape device
- `f /dev/rmt/0` identifies the tape device
- `/export/home/kryten` is the directory being backed up

```
# umount /export/home

# ufsdump 0ucf /dev/rmt/0 /export/home/kryten
  DUMP: Date of this level 0 dump: Tue Oct 07 08:41:41 2003
  DUMP: Date of last level 0 dump: the epoch
  DUMP: Dumping /dev/rdsk/c0t0d0s0 (starbug:/) to /dev/rmt/0.
  DUMP: Mapping (Pass I) [regular files]
  DUMP: Mapping (Pass II) [directories]
  DUMP: Writing 63 Kilobyte records
  DUMP: Estimated 470 blocks (235KB).
  DUMP: Dumping (Pass III) [directories]
  DUMP: Dumping (Pass IV) [regular files]
  DUMP: 376 blocks (188KB) on 1 volume at 1205 KB/sec
  DUMP: DUMP IS DONE
# ufsrestore tf /dev/rmt/0
         2        .
         5      ./export
         6      ./export/home
     80799      ./export/home/kryten
     80800      ./export/home/kryten/filea
     80801      ./export/home/kryten/fileb
     80802      ./export/home/kryten/filec
     80803      ./export/home/kryten/letters
     80804      ./export/home/kryten/letters/letter1
     80805      ./export/home/kryten/letters/letter2
     80806      ./export/home/kryten/letters/letter3
     80807      ./export/home/kryten/reports
     80808      ./export/home/kryten/reports/reportA
     80809      ./export/home/kryten/reports/reportB
     80810      ./export/home/kryten/reports/reportC
#
```

Example 23–6 Performing a Full Backup to a Remote System (Solaris 9 Data to Solaris 9 System)

The following example shows how to do a full backup of a local `/export/home` file system on a Solaris 9 system (`starbug`) to a tape device on a remote Solaris 9 system (`earth`) in single-user mode. The following `ufsdump` options are included:

- `0` specifies a `0` level dump (or a full backup).
- `u` specifies that the `/etc/dumpdates` file is updated with the date of this backup.
- `c` identifies a cartridge tape device.
- `f earth:/dev/rmt/0` identifies the remote system name and tape device
- `/export/home` is the file system being backed up.

```
# ufsdump 0ucf earth:/dev/rmt/0 /export/home
  DUMP: Date of this level 0 dump: Mon Oct 06 12:46:50 2003
  DUMP: Date of last level 0 dump: the epoch
```

```
 DUMP: Dumping /dev/rdsk/c0t0d0s7 (starbug:/export/home) to
 earth:/dev/rmt/0.
 DUMP: Mapping (Pass I) [regular files]
 DUMP: Mapping (Pass II) [directories]
 DUMP: Writing 63 Kilobyte records
 DUMP: Estimated 410 blocks (205KB).
 DUMP: Dumping (Pass III) [directories]
 DUMP: Dumping (Pass IV) [regular files]
 DUMP: Tape rewinding
 DUMP: 376 blocks (188KB) on 1 volume at 546 KB/sec
 DUMP: DUMP IS DONE
 DUMP: Level 0 dump on Mon Oct 06 12:46:50 2003
# **ufsrestore tf earth:/dev/rmt/0**
         2      .
        3      ./lost+found
        4      ./kryten
        5      ./kryten/filea
        6      ./kryten/fileb
        7      ./kryten/filec
        8      ./kryten/letters
        9      ./kryten/letters/letter1
       10      ./kryten/letters/letter2
       11      ./kryten/letters/letter3
       12      ./kryten/reports
.
.
.
#
```

CHAPTER 24

Using UFS Snapshots (Tasks)

This chapter describes how to create and back up UFS snapshots.

For information on the procedures associated with creating UFS snapshots, see "Using UFS Snapshots (Task Map)" on page 397.

For overview information about performing backups, see Chapter 22.

Using UFS Snapshots (Task Map)

Task	Description	For Instructions
1. Create a UFS snapshot	Create a read-only copy of a file system by using the `fssnap` command.	"How to Create a UFS Snapshot" on page 400
2. Display UFS snapshot information	Identify UFS snapshot information such as the raw snapshot device.	"How to Display UFS Snapshot Information" on page 401
3. (Optional) Delete a UFS snapshot	Delete a snapshot that is already backed up or no longer needed.	"How to Delete a UFS Snapshot" on page 402
4. Back up a UFS snapshot	Choose one of the following backup methods:	
	Create a full backup of a UFS snapshot by using the `ufsdump` command.	"How to Create a Full Backup of a UFS Snapshot (`ufsdump`)" on page 403

Task	Description	For Instructions
	Create an incremental backup of a UFS snapshot by using the `ufsdump` command.	"How to Create an Incremental Backup of a UFS Snapshot (`ufsdump`)" on page 404
	Back up a UFS snapshot by using the `tar` command.	"How to Back Up a UFS Snapshot (`tar`)" on page 404
5. (Optional) Restore data from a UFS snapshot	Restore the UFS snapshot the same way as you would restore data by using the `ufsrestore` command.	"How to Restore a Complete File System" on page 415

UFS Snapshots Overview

The Solaris release includes the `fssnap` command for backing up file systems while the file system is mounted. You can use the `fssnap` command to create a read-only snapshot of a file system. A *snapshot* is a file system's temporary image that is intended for backup operations.

When the `fssnap` command is run, it creates a virtual device and a backing-store file. You can back up the *virtual device*, which looks and acts like a real device, with any of the existing Solaris backup commands. The *backing-store file* is a bitmap file that contains copies of presnapshot data that has been modified since the snapshot was taken.

Why Use UFS Snapshots?

The UFS snapshots feature enables you to keep the file system mounted and the system in multiuser mode during backups. Previously, you were advised to bring the system to single-user mode to keep the file system inactive when you used the `ufsdump` command to perform backups. You can also use additional Solaris backup commands, such as `tar` and `cpio`, to back up a UFS snapshot for more reliable backups.

The `fssnap` command gives administrators of nonenterprise-level systems the power of enterprise-level tools, such as Sun StorEdge™ Instant Image, without the large storage demands.

The UFS snapshots feature is similar to the Instant Image product. Although UFS snapshots can make copies of large file systems, Instant Image is better suited for enterprise-level systems. UFS snapshots is better suited for smaller systems. Instant Image allocates space equal to the size of the entire file system that is being captured. However, the backing-store file that is created by UFS snapshots occupies only as much disk space as needed.

This table describes specific differences between UFS snapshots and Instant Image.

UFS Snapshots	Sun StorEdge Instant Image
Size of the backing-store file depends on how much data has changed since the snapshot was taken	Size of the backing-store file equivalent equals the size of the entire file system being copied
Does not persist across system reboots	Persists across system reboots
Works on UFS file systems	Cannot be used with root (`/`) or `/usr` file systems
Available starting with the Solaris 8 1/01 release	Part of Sun StorEdge products

UFS Snapshots Performance Issues

When the UFS snapshot is first created, users of the file system might notice a slight pause. The length of the pause increases with the size of the file system to be captured. While the snapshot is active, users of the file system might notice a slight performance impact when the file system is written to, but they see no impact when the file system is read.

Creating and Deleting UFS Snapshots

When you use the `fssnap` command to create a UFS snapshot, observe how much disk space the backing-store file consumes. The backing-store file uses no space, and then it grows quickly, especially on heavily used systems. Make sure that the backing-store file has enough space to expand. Or, limit its size with the `-o maxsize=n [k,m,g]` option, where *n* `[k,m,g]` is the maximum size of the backing-store file.

Caution – If the backing-store file runs out of space, the snapshot might delete itself, which causes the backup to fail. Check the `/var/adm/messages` file for possible snapshot errors.

For more information, see the `fssnap_ufs`(1M) man page.

▼ How to Create a UFS Snapshot

Steps

1. **Become superuser or assume an equivalent role.**
2. **Make sure that the file system has enough disk space for the backing-store file.**

   ```
   # df -k
   ```
3. **Make sure that a backing-store file of the same name and location does not already exist.**

   ```
   # ls /backing-store-file
   ```
4. **Create the UFS snapshot.**

   ```
   # fssnap -F ufs -o bs=/backing-store-file /file-system
   ```

   Note – The backing-store file must reside on a different file system than the file system that is being captured using UFS snapshots.
5. **Verify that the snapshot has been created.**

   ```
   # /usr/lib/fs/ufs/fssnap -i /file-system
   ```

Example 24–1 Creating a UFS Snapshot

The following example shows how to create a snapshot of the `/usr` file system. The backing-store file is `/scratch/usr.back.file`. The virtual device is `/dev/fssnap/1`.

```
# fssnap -F ufs -o bs=/scratch/usr.back.file /usr
/dev/fssnap/1
```

The following example shows how to limit the backing-store file to 500 Mbytes.

```
# fssnap -F ufs -o maxsize=500m,bs=/scratch/usr.back.file /export/home
/dev/fssnap/1
```

▼ How to Display UFS Snapshot Information

You can display the current snapshots on the system by using the `fssnap -i` option. If you specify a file system, you see detailed information about that snapshot. If you don't specify a file system, you see information about all of the current UFS snapshots and their corresponding virtual devices.

Note – Use the UFS file system-specific `fssnap` command to view the extended snapshot information as shown in the following examples.

Steps

1. **Become superuser or assume an equivalent role.**
2. **List all current snapshots:**

   For example:

   ```
   # /usr/lib/fs/ufs/fssnap -i
   Snapshot number               : 0
   Block Device                  : /dev/fssnap/0
   Raw Device                    : /dev/rfssnap/0
   Mount point                   : /usr
   Device state                  : idle
   Backing store path            : /var/tmp/snapshot3
   Backing store size            : 256 KB
   Maximum backing store size    : Unlimited
   Snapshot create time          : Wed Oct 08 10:38:25 2003
   Copy-on-write granularity     : 32 KB
   Snapshot number               : 1
   Block Device                  : /dev/fssnap/1
   Raw Device                    : /dev/rfssnap/1
   Mount point                   : /
   Device state                  : idle
   Backing store path            : /tmp/bs.home
   Backing store size            : 448 KB
   Maximum backing store size    : Unlimited
   Snapshot create time          : Wed Oct 08 10:39:29 2003
   Copy-on-write granularity     : 32 KB
   ```

3. **Display detailed information about a specific snapshot:**

   For example:

   ```
   # /usr/lib/fs/ufs/fssnap -i /usr
   Snapshot number               : 0
   Block Device                  : /dev/fssnap/0
   Raw Device                    : /dev/rfssnap/0
   Mount point                   : /usr
   Device state                  : idle
   Backing store path            : /var/tmp/snapshot3
   Backing store size            : 256 KB
   Maximum backing store size    : Unlimited
   Snapshot create time          : Wed Oct 08 10:38:25 2003
   ```

```
Copy-on-write granularity     : 32 KB
```

Deleting a UFS Snapshot

When you create a UFS snapshot, you can specify that the backing-store file is unlinked. An unlinked backing-store file is removed after the snapshot is deleted. If you don’t specify the `-o unlink` option when you create a UFS snapshot, you must delete the backing-store file manually.

The backing-store file occupies disk space until the snapshot is deleted, whether you use the `-o unlink` option to remove the backing-store file or you delete the file manually.

▼ How to Delete a UFS Snapshot

You can delete a snapshot either by rebooting the system or by using the `fssnap -d` command. When you use this command, you must and specify the path of the file system that contains the UFS snapshot.

Steps

1. **Become superuser or assume an equivalent role.**
2. **Identify the snapshot to be deleted.**

   ```
   # /usr/lib/fs/ufs/fssnap -i
   ```
3. **Delete the snapshot.**

   ```
   # fssnap -d /file-system
   Deleted snapshot 1.
   ```
4. **(Optional) If you did not use the `-o unlink` option when you created the snapshot, manually delete the backing-store file.**

   ```
   # rm /file-system/backing-store-file
   ```

Example 24–2 Deleting a UFS Snapshot

The following example shows how to delete a snapshot and assumes that the `-o unlink` option was not used.

```
# fssnap -i
    0    /    1    /usr
# fssnap -d /usr
 Deleted snapshot 1.
# rm /scratch/usr.back.file
```

Backing Up a UFS Snapshot

You can create a full backup or an incremental backup of a UFS snapshot. You can use the standard Solaris backup commands to back up a UFS snapshot.

The virtual device that contains the UFS snapshot acts as a standard read-only device. So, you can back up the virtual device as if you were backing up a file system device.

If you are using the `ufsdump` command to back up a UFS snapshot, you can specify the snapshot name during the backup. See the following procedure for more information.

▼ How to Create a Full Backup of a UFS Snapshot (`ufsdump`)

Steps

1. **Become superuser or assume an equivalent role.**
2. **Identify the UFS snapshot to be backed up.**

   ```
   # /usr/lib/fs/ufs/fssnap -i /file-system
   ```

   For example:

   ```
   # /usr/lib/fs/ufs/fssnap -i /usr
   Snapshot number               : 0
   Block Device                  : /dev/fssnap/0
   Raw Device                    : /dev/rfssnap/0
   Mount point                   : /usr
   Device state                  : idle
   Backing store path            : /var/tmp/snapshot3
   Backing store size            : 256 KB
   Maximum backing store size    : Unlimited
   Snapshot create time          : Wed Oct 08 10:38:25 2003
   Copy-on-write granularity     : 32 KB
   ```

3. **Back up the UFS snapshot.**

   ```
   # ufsdump 0ucf /dev/rmt/0 /snapshot-name
   ```

   For example:

   ```
   # ufsdump 0ucf /dev/rmt/0 /dev/rfssnap/1
   ```

4. **Verify that the snapshot is backed up.**

   For example:

   ```
   # ufsrestore tf /dev/rmt/0
   ```

▼ How to Create an Incremental Backup of a UFS Snapshot (`ufsdump`)

Backing up a UFS snapshot incrementally means that only the files that have been modified since the last snapshot are backed up. Use the `ufsdump` command with the new `N` option. This option specifies the file system device name to be inserted into the `/etc/dumpdates` file for tracking incremental dumps.

The following `ufsdump` command specifies an embedded `fssnap` command to create an incremental backup of a file system.

Steps

1. **Become superuser or assume an equivalent role.**
2. **Create an incremental backup of a UFS snapshot.**

   For example:

   ```
   # ufsdump 1ufN /dev/rmt/0 /dev/rdsk/c0t1d0s0 `fssnap -F ufs -o raw,bs=
   /export/scratch,unlink /dev/rdsk/c0t1d0s0`
   ```

   In this example, the `-o raw` option is used to display the name of the raw device instead of the block device. By using this option, you make it easier to embed the `fssnap` command in commands (such as the `ufsdump` command) that require the raw device instead.
3. **Verify that the snapshot is backed up.**

   ```
   # ufsrestore ta /dev/rmt/0
   ```

▼ How to Back Up a UFS Snapshot (`tar`)

If you are using the `tar` command to back up the snapshot, mount the snapshot before backing it up.

Steps

1. **Become superuser or assume an equivalent role.**
2. **Create a mount point for the snapshot.**

   For example:

   ```
   # mkdir /backups/home.bkup
   ```
3. **Mount the snapshot.**

   ```
   # mount -F ufs -o ro /dev/fssnap/1 /backups/home.bkup
   ```
4. **Change to the mounted snapshot directory.**

   ```
   # cd /backups/home.bkup
   ```
5. **Back up the snapshot with the `tar` command.**

```
# tar cvf /dev/rmt/0 .
```

Restoring Data From a UFS Snapshot Backup

The backup created from the virtual device is essentially just a backup of what the original file system looked like when the snapshot was taken. When you restore a file system from the backup, restore as if you had taken the backup directly from the original file system. Such a restore uses the `ufsrestore` command. For information on using the `ufsrestore` command to restore a file or file system, see Chapter 25.

CHAPTER 25

Restoring Files and File Systems (Tasks)

This chapter describes how to use the `ufsrestore` command to restore files and file systems that were backed up by using the `ufsdump` command.

For information on the procedures associated with restoring files and file systems, see "Restoring Files and File System Backups (Task Map)" on page 407.

For information about other commands you can use to archive, restore, copy, or move files and file systems, see Chapter 27.

For information about backing up and restoring file systems, see Chapter 22.

Restoring Files and File System Backups (Task Map)

The following task map describes the procedures associated with restoring files and file systems.

Task	Description	For Instructions
Prepare to restore files and file systems	Identify the file systems or files to be restored, the tape device, and how you will restore them.	"Preparing to Restore Files and File Systems" on page 408
Determine which tapes to use	Refer to your backup tapes to find the date of the last backup that contains the file or file system that you need to restore.	"How to Determine Which Tapes to Use" on page 410

Task	Description	For Instructions
Restore files	Choose one of the following restore methods:	
	Restore files interactively–Use this method when you are unsure of the file names because you can browse the media contents and select individual files and directories.	"How to Restore Files Interactively" on page 411
	Restore files noninteractively–Use this method when you already know the few file names to be restored.	"How to Restore Specific Files Noninteractively" on page 413
	Restore a file system – Use this method when you get a new disk drive or as part of a recovery procedure.	"How to Restore a Complete File System" on page 415
Restore the root (/) or `/usr` file systems	Restoring the root (/) or `/usr` file systems involves booting the system from a local CD or the network.	"How to Restore the root (/) and `/usr` File Systems" on page 418

Preparing to Restore Files and File Systems

The `ufsrestore` command copies files to disk, relative to the current working directory, from backups that were created by using the `ufsdump` command. You can use the `ufsrestore` command to reload an entire file system hierarchy from a level 0 dump and incremental dumps that follow it. You can also use this command to restore one or more single files from any backup tape. If you run the `ufsrestore` command as superuser, files are restored with their original owner, last modification time, and mode (permissions).

Before you start to restore files or file systems, you need to know the following:

- The tapes (or diskettes) you need to restore from
- The raw device name on which you want to restore the file system
- The type of tape device you will use
- The device name (local or remote) for the tape device

Determining the File System Name

If you have properly labeled your backup tapes, you should be able to use the file system name (`/dev/rdsk/`*device-name*) from the tape label. For more information, see "How to Find File System Names" on page 388.

Determining the Type of Tape Device You Need

You must use a tape device that is compatible with the backup media to restore the files. The format of the backup media determines which drive you must use to restore files. For example, if your backup media is 8-mm tape, you must use an 8-mm tape device to restore the files.

Determining the Tape Device Name

You might have specified the tape device name (`/dev/rmt/`*n*) as part of the backup tape label information. If you are using the same drive to restore a backup tape, you can use the device name from the label. For more information on media devices and device names, see Chapter 28.

Restoring Files and File Systems

When you back up files and directories, you save them relative to the file system in which they belong. When you restore files and directories, the `ufsrestore` command re-creates the file hierarchy in the current working directory.

For example, files backed up from the `/export/doc/books` directory (where `/export` is the file system), are saved relative to `/export`. In other words, the `book1` file in the `books` directory is saved as `./doc/books/book1` on the tape. Later on, if you restored the `./doc/books/book1` file to the `/var/tmp` directory, the file would be restored to `/var/tmp/doc/books/book1`.

When you restore individual files and directories, you should restore them to a temporary location, such as the `/var/tmp` directory. After you verify the files, you can move them to their proper locations. However, you can restore individual files and directories to their original locations. If you do so, be sure you are not overwriting newer files with older versions from the backup tape.

To avoid conflicts with other users, you might want to create and change to a subdirectory, such as the`/var/tmp/restore` file, in which to restore the files.

If you are restoring a hierarchy, you should restore the files to a temporary directory on the same file system where the files will reside. Then, you can use the `mv` command to move the entire hierarchy where it belongs after it is restored.

Note – Do not restore files in the `/tmp` directory even temporarily. The `/tmp` directory is usually mounted as a TMPFS file system. TMPFS does not support UFS file system attributes such as ACLs.

▼ How to Determine Which Tapes to Use

Steps

1. **Ask the user for the approximate date the files to be restored were last modified.**

2. **Refer to your backup plan to find the date of the last backup that contains the file or file system.**

   To retrieve the most recent version of a file, work backward through the incremental backups from highest to lowest dump level and from most recent to least recent date, unless the user requests otherwise.

3. **(Optional) If you have online archive files, identify the correct media.**

   ```
   # ufsrestore ta archive-name ./path/filename ./path/filename
   ```

`t`	Lists each file on the tape.
`a`	Reads the table of contents from the online archive file instead of from the tape.
archive-name	Identifies the online archive file name.
./path/filename	Identifies the file name or file names you are looking for on the online archive. If successful, the `ufsrestore` command prints out the inode number and file name. If unsuccessful, `ufsrestore` prints an error message.

   For more information, see the `ufsrestore`(1M) man page.

4. **Insert the media that contains the files to be restored in the drive and verify the correct media.**

   ```
   # ufsrestore tf /dev/rmt/n ./path/filename ./path/filename
   ```

   Be sure to use the complete path for each *filename*. If a file is in the backup, its name and inode number is listed. Otherwise, a message states that the file is not on the volume.

5. **(Optional) If you have multiple backup files on the same tape, position the tape at the backup file you want to use.**

   ```
   # ufsrestore tfs /dev/rmt/n tape-number
   ```

Example 25–1 Determining Which Tapes to Use

The following example shows how to check if the `/etc/passwd` file is in the online archive.

```
# ufsrestore ta /var/tmp/root.archive ./etc/passwd
```

The following example shows how to verify that the `/etc/passwd` file is on the backup tape.

```
# ufsrestore tf /dev/rmt/0 ./etc/passwd
```

▼ How to Restore Files Interactively

Steps

1. **Become superuser or assume an equivalent role.**
2. **(Optional) Write-protect the tapes for safety.**
3. **Insert the volume 1 tape into the tape drive.**
4. **Change to a directory that will be used to restore the files to temporarily.**

   ```
   # cd /var/tmp
   ```
5. **Start the interactive restoration.**

   ```
   # ufsrestore if /dev/rmt/n
   ```
   Some informational messages and the `ufsrestore>` prompt are displayed.
6. **Create a list of files to be restored.**
   a. **List the contents of a directory.**

      ```
      ufsrestore> ls [directory-name]
      ```
   b. **Change to a directory.**

      ```
      ufsrestore> cd directory-name
      ```
   c. **Create a list of files and directories that you want to restore.**

      ```
      ufsrestore> add filenames
      ```
   d. **(Optional) Remove any directory or file from the list of files to be restored, if necessary.**

      ```
      ufsrestore> delete filename
      ```
7. **(Optional) Display the file names as they are being restored.**

   ```
   ufsrestore> verbose
   ```

8. **Restore the files.**

   ```
   ufsrestore> extract
   ```

   The `ufsrestore` command asks you which volume number to use.

9. **Type the volume number and press Return. If you have only one volume, type `1` and press Return.**

   ```
   Specify next volume #: 1
   ```

   The files and directories in the list are extracted and restored to the current working directory.

10. **To maintain the mode of the current directory, enter `n` at the `set owner/mode` prompt.**

    ```
    set owner/mode for '.'? [yn] n
    ```

    You must wait while the `ufsrestore` command performs its final cleanup.

11. **Quit the `ufsrestore` program.**

    ```
    ufsrestore> quit
    ```

    You then see the shell prompt.

12. **Verify the restored files.**

    a. **List the restored files and directories.**

       ```
       # ls -l
       ```

       A list of files and directories is displayed.

    b. **Check the list to be sure that all the files and directories you specified in the list have been restored.**

13. **Move the files to the proper directories.**

Example 25–2 Restoring Files Interactively

The following example shows how to extract the `/etc/passwd` and `/etc/shadow` files from the backup tape.

```
# cd /var/tmp
# ufsrestore if /dev/rmt/0
ufsrestore> ls
.:
 .cpr_config   etc/          lost+found/   sbin/
 TT_DB/        export/       mnt/          tmp/
 b/            home/         net/          usr
 bin           kernel/       opt/          var/
 dev/          lib           platform/     vol/
 devices/      license/      proc/
ufsrestore> cd etc
```

```
ufsrestore> add passwd shadow
ufsrestore> verbose
verbose mode on
ufsrestore> extract
Extract requested files
You have not read any volumes 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
extract file ./etc/shadow
extract file ./etc/passwd
Add links
Set directory mode, owner, and times.
set owner/mode for '.'? [yn] n
ufsrestore> quit
#
```

▼ How to Restore Specific Files Noninteractively

Steps

1. **Become superuser or assume an equivalent role.**
2. **(Optional) Write-protect the tape for safety.**
3. **Insert the volume 1 tape into the tape drive.**
4. **Change to a directory that will be used to restore files to temporarily.**

   ```
   # cd /var/tmp
   ```

5. **Restore the file or files.**

   ```
   # ufsrestore xvf /dev/rmt/n filename
   ```

`x`	Tells `ufsrestore` to copy specific files or directories in the *filename* argument.
`v`	Displays the file names as they are restored.
`f /dev/rmt/`*n*	Identifies the tape device name.
filename	Specifies one or more file names or directory names, separated by spaces. For example: `./export/home/user1/mail ./export/home/user2/mail`.

6. **Type the volume number where files are located. Press Return.**

   ```
   Specify next volume #: 1
   ```

   The file or files are restored to the current working directory.

7. **To maintain the mode of the current directory, type `n` and press Return at the `set owner/mode` prompt.**

   ```
   set owner/mode for '.'? [yn] n
   ```

8. **Verify the restored files.**

   a. **List the restored files and directories.**

      ```
      # ls -l
      ```

      A list of files and directories is displayed.

   b. **Check the list to be sure that all the files and directories you specified in the list have been restored.**

9. **Move the files to the proper directories.**

Example 25–3 Restoring Specific Files Noninteractively

The following example shows how to noninteractively restore the `passwd` and `shadow` files to the `/var/tmp` directory.

```
# cd /var/tmp
# ufsrestore xvf /dev/rmt/0 ./etc/passwd ./etc/shadow
Verify volume and initialize maps
Media block size is 126
Dump   date: Mon Oct 06 12:36:10 2003
Dumped from: the epoch
Level 9 dump of / on starbug:/dev/dsk/c0t0d0s0
Label: none
Extract directories from tape
Initialize symbol table.
Make node ./etc
Extract requested files
You have not read any volumes 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
extract file ./etc/passwd
extract file ./etc/shadow
Add links
Set directory mode, owner, and times.
set owner/mode for `.'? [yn] n
# cd etc
# mv passwd /etc
# mv shadow /etc
# ls -l /etc
```

Example 25–4 Restoring Files From a Remote Tape Device

You can restore files from a remote tape drive by adding *remote-host*`:` to the front of the tape device name, when using the `ufsrestore` command.

The following example shows how to restore files by using a remote tape drive `/dev/rmt/0` on the system `venus`.

```
# ufsrestore xf venus:/dev/rmt/0 ./etc/hosts
```

▼ How to Restore a Complete File System

Occasionally, a file system becomes so damaged that you must completely restore it. Typically, you need to restore a complete file system after a disk failure. You might need to replace the hardware before you can restore the software. For information on how to replace a disk, see Chapter 12 or Chapter 13.

Full restoration of a file system such as `/export/home` can take a lot of time. If you have consistently backed up file systems, you can restore them to their state from the time of the last incremental backup.

Note – You cannot use this procedure to restore the root (`/`) or `/usr` file systems. For instructions on restoring these file systems, see "How to Restore the root (`/`) and `/usr` File Systems" on page 418.

Steps

1. **Become superuser or assume an equivalent role.**
2. **If necessary, unmount the file system.**

   ```
   # umount /dev/rdsk/device-name
   ```

   Or:

   ```
   # umount /file-system
   ```

3. **Create the new file system.**

   ```
   # newfs /dev/rdsk/device-name
   ```

   You are asked if you want to construct a new file system on the raw device. Verify that the *device-name* is correct so that you don't destroy the wrong file system.

   For more information, see the `newfs`(1M) man page.

4. **Confirm that the new file system should be created.**

   ```
   newfs: construct a new file system /dev/rdsk/cwtxdysz:(y/n)? y
   ```

   The new file system is created.

5. **Mount the new file system on a temporary mount point.**

   ```
   # mount /dev/dsk/device-name /mnt
   ```

6. **Change to the mount point directory.**

   ```
   # cd /mnt
   ```

7. **(Optional) Write-protect the tapes for safety.**

8. **Insert the first volume of the level 0 tape into the tape drive.**

9. **Restore the files.**

   ```
   # ufsrestore rvf /dev/rmt/n
   ```

   The dump level 0 backup is restored. If the backup required multiple tapes, you are prompted to load each tape in numeric order.

10. **Remove the tape and load the next level tape in the drive.**

    Always restore tapes starting with dump level 0 and continuing until you reach the highest dump level.

11. **Repeat Step 8 through Step 10 for each dump level, from the lowest to the highest level.**

12. **Verify that the file system is restored.**

    ```
    # ls
    ```

13. **Remove the `restoresymtable` file.**

    ```
    # rm restoresymtable
    ```

    The `restoresymtable` file that is created and used by the `ufsrestore` command to check-point the restore is removed.

14. **Change to another directory.**

    ```
    # cd /
    ```

15. **Unmount the newly restored file system.**

    ```
    # umount /mnt
    ```

16. **Remove the last tape and insert a new tape that is not write-protected in the tape drive.**

17. **Make a level 0 backup of the newly restored file system.**

    ```
    # ufsdump 0ucf /dev/rmt/n /dev/rdsk/device-name
    ```

    A level 0 backup is performed. Always immediately do a full backup of a newly created file system because the `ufsrestore` command repositions the files and changes the inode allocation.

18. **Mount the restored file system.**

    ```
    # mount /dev/dsk/device-name mount-point
    ```

    The restored file system is mounted and available for use.

19. **Verify that the restored and mounted file system is available.**

    ```
    # ls mount-point
    ```

Example 25–5 Restoring a Complete File System

The following example shows how to restore the /export/home file system.

```
# umount /export/home
# newfs /dev/rdsk/c0t0d0s7
newfs: /dev/rdsk/c0t0d0s7 last mounted as /export/home
newfs: construct a new file system /dev/rdsk/c0t0d0s7: (y/n)? y
819314 sectors in 867 cylinders of 15 tracks, 63 sectors
        400.1MB in 55 cyl groups (16 c/g, 7.38MB/g, 3584 i/g)
super-block backups (for fsck -F ufs -o b=#) at:
 32, 15216, 30400, 45584, 60768, 75952, 91136, 106320, 121504, 136688,
 681264, 696448, 711632, 725792, 740976, 756160, 771344, 786528, 801712,
 816896,
Verify volume and initialize maps
Media block size is 126
Dump   date: Tue Oct 07 08:41:41 2003
Dumped from: the epoch
Level 0 dump of a partial file system on starbug:/export/home/kryten
Label: none
Begin level 0 restore
Initialize symbol table.
Extract directories from tape
Calculate extraction list.
Extract new leaves.
Check pointing the restore
extract file ./export/home/kryten/filea
extract file ./export/home/kryten/fileb
extract file ./export/home/kryten/filec
extract file ./export/home/kryten/letters/letter1
extract file ./export/home/kryten/letters/letter2
extract file ./export/home/kryten/letters/letter3
extract file ./export/home/kryten/reports/reportA
extract file ./export/home/kryten/reports/reportB
extract file ./export/home/kryten/reports/reportC
Add links
Set directory mode, owner, and times.
Check the symbol table.
Check pointing the restore
# mount /dev/dsk/c0t0d0s7 /mnt
# cd /mnt
# ufsrestore rvf /dev/rmt/0
# ls
# rm restoresymtable
```

```
# cd /
# umount /mnt
# ufsdump 0ucf /dev/rmt/0 /export/home
                    .
                    .
                    .
# mount /dev/dsk/c0t3d0s7 /export/home
# ls /export/home
```

▼ How to Restore the root (/) and /usr File Systems

Steps

1. **Become superuser or assume an equivalent role.**
2. **Add a new system disk to the system where the root (/) and `/usr` file systems will be restored.**

   For a detailed description about adding a system disk, refer to Chapter 12 or Chapter 13.
3. **Mount the new file system on a temporary mount point.**

   ```
   # mount /dev/dsk/device-name /mnt
   ```
4. **Change to the `/mnt` directory.**

   ```
   # cd /mnt
   ```
5. **(Optional) Write-protect the tapes for safety.**
6. **Create the links for the tape device.**

   ```
   # tapes
   ```
7. **Restore the root file system.**

   ```
   # ufsrestore rvf /dev/rmt/n
   ```
   The dump level 0 tape is restored.
8. **Remove the tape and load the next level tape in the drive.**

   Always restore tapes starting with dump level 0 and continuing from the lowest to highest dump level.
9. **Continue restoring as needed.**

   ```
   # ufsrestore rvf /dev/rmt/n
   ```
   The next level tape is restored.
10. **Repeat Step 8 and Step 9 for each additional tape.**

11. **Verify that the file system is restored.**

    ```
    # ls
    ```

12. **Remove the `restoresymtable` file.**

    ```
    # rm restoresymtable
    ```

    The `restoresymtable` file that is created and used by the `ufsrestore` command to check-point the restore is removed.

13. **Change to the root (/) directory.**

    ```
    # cd /
    ```

14. **Unmount the newly created file system.**

    ```
    # umount /mnt
    ```

15. **Check the new file system.**

    ```
    # fsck /dev/rdsk/device-name
    ```

    The restored file system is checked for consistency.

16. **Create the boot blocks on the root partition.**

    ```
    # installboot  /usr/platform/`uname-i`/lib/fs/ufs/bootblk
    /dev/rdsk/device-name
    ```

    For more information, see the`installboot`(1M) man page.

    For an example of using the `installboot` command on a SPARC based system, see Example 25–6. For an example of using the `installboot` command on an x86 based system, see Example 25–7.

17. **Insert a new tape in the tape drive.**

18. **Back up the new file system.**

    ```
    # ufsdump 0uf /dev/rmt/n /dev/rdsk/device-name
    ```

    A dump level 0 backup is performed. Always immediately do a full backup of a newly created file system because the `ufsrestore` command repositions the files and changes the inode allocation.

19. **Repeat steps 5 through 16 for the `/usr` file system, if necessary.**

20. **Reboot the system.**

    ```
    # init 6
    ```

    The system is rebooted.

Example 25–6 SPARC: Restoring the root (/) File System

This example shows how to restore the root (/) file system on a SPARC system. This example assumes that the system is booted from a local CD or from the network.

```
# mount /dev/dsk/c0t3d0s0 /mnt
# cd /mnt
# tapes
# ufsrestore rvf /dev/rmt/0
# ls
# rm restoresymtable
# cd /
# umount /mnt
# fsck /dev/rdsk/c0t3d0s0
# installboot /usr/platform/sun4u/lib/fs/ufs/bootblk /dev/rdsk/c0t3d0s0
# ufsdump 0uf /dev/rmt/0 /dev/rdsk/c0t3d0s0
# init 6
```

Example 25–7 x86: Restoring the root (/) File System

This example shows how to restore the root (/) file system on an x86 system. This example assumes that the system is booted from a local CD or from the network.

```
# mount /dev/dsk/c0t3d0s0 /mnt
# cd /mnt
# tapes
# ufsrestore rvf /dev/rmt/0
# ls
# rm restoresymtable
# cd /
# umount /mnt
# fsck /dev/rdsk/c0t3d0s0
# installboot /usr/platform/‘uname -i‘/lib/fs/ufs/pboot /usr/platform/‘uname -i‘/lib/fs/
ufs/bootblk /dev/rdsk/c0t3d0s2
# ufsdump 0uf /dev/rmt/0 /dev/rdsk/c0t3d0s0
# init 6
```

CHAPTER 26

UFS Backup and Restore Commands (Reference)

This chapter contains reference information on the `ufsdump` and `ufsrestore` commands.

This is a list of information in this chapter.

For overview information about performing backups, see Chapter 22.

For information about backup tasks, see Chapter 23.

How the `ufsdump` Command Works

The `ufsdump` command makes two passes when it backs up a file system. On the first pass, this command scans the raw device file for the file system and builds a table of directories and files in memory. Then, this command writes the table to the backup media. In the second pass, the `ufsdump` command goes through the inodes in numerical order, reading the file contents and writing the data to the backup media.

Determining Device Characteristics

The `ufsdump` command needs to know only an appropriate tape block size and how to detect the end of media.

Detecting the End of Media

The `ufsdump` command writes a sequence of fixed-size records. When the `ufsdump` command receives notification that a record was only partially written, it assumes that it has reached the physical end of the media. This method works for most devices. If a device is not able to notify the `ufsdump` command that only a partial record has been written, a media error occurs as the `ufsdump` command tries to write another record.

Note – DAT devices and 8-mm tape devices detect end-of-media. Cartridge tape devices and 1/2–inch tape devices do not detect end-of-media.

The `ufsdump` command automatically detects the end-of-media for most devices. Therefore, you do not usually need to use the `-c`, `-d`, `-s`, and `-t` options to perform multivolume backups.

The only times you need to use the end-of-media options are under the following conditions:

- The `ufsdump` command does not understand the way the device detects the end-of-media.
- You are going to restore the files on a SunOS 4. 1 system with the `restore` command.

To ensure compatibility with the `restore` command, the size option can still force the `ufsdump` command to go to the next tape or diskette before reaching the end of the current tape or diskette.

Copying Data With the `ufsdump` Command

The `ufsdump` command copies data only from the raw disk slice. If the file system is still active, any data in memory buffers is probably not copied. The backup done by the `ufsdump` command does not copy free blocks, nor does it make an image of the disk slice. If symbolic links point to files on other slices, the link itself is copied.

Purpose of the `/etc/dumpdates` File

The `ufsdump` command, when used with the `-u` option, maintains and updates the `/etc/dumpdates` file. Each line in the `/etc/dumpdates` file shows the following information:

- The file system backed up
- The dump level of the last backup
- The day, date, and time of the backup

For example:

```
/dev/rdsk/c0t0d0s7              0 Wed Oct  8 10:30:52 2003
/dev/rdsk/c0t0d0s0              0 Tue Oct  6 10:12:13 2003
/dev/rdsk/c0t0d0s0              9 Wed Oct  8 10:26:14 2003
```

When you do an incremental backup, the `ufsdump` command checks the `/etc/dumpdates` file to find the date of the most recent backup of the next lower dump level. Then, this command copies to the media all files that were modified since the date of that lower-level backup. After the backup is complete, a new information line, which describes the backup you just completed, replaces the information line for the previous backup at that level.

Use the `/etc/dumpdates` file to verify that backups are being done. This verification is particularly important if you are having equipment problems. If a backup cannot be completed because of equipment failure, the backup is not recorded in the `/etc/dumpdates` file.

If you need to restore an entire disk, check the `/etc/dumpdates` file for a list of the most recent dates and levels of backups so that you can determine which tapes you need to restore the entire file system.

Note – The `/etc/dumpdates` file is a text file that can be edited. However, edit it only at your own risk. If you make changes to the file that do not match your archive tapes, you might be unable to find the tapes (or files) you need.

Backup Device (*dump-file*) Argument

The *dump-file* argument (to the `-f` option) specifies the destination of the backup. The destination can be one of the following:

- Local tape drive
- Local diskette drive
- Remote tape drive
- Remote diskette drive
- Standard output

Use this argument when the destination is not the default local tape drive `/dev/rmt/0`. If you use the `-f` option, then you must specify a value for the*dump-file* argument.

Note – The *dump-file* argument can also point to a file on a local disk or on a remote disk. If done by mistake, this usage can fill up a file system.

Local Tape or Diskette Drive

Typically, the *dump-file* argument specifies a raw device file for a tape device or diskette. When the `ufsdump` command writes to an output device, it creates a single backup file that might span multiple tapes or diskettes.

You specify a tape device or a diskette on your system by using a device abbreviation. The first device is always `0`. For example, if you have a SCSI tape controller and one QIC-24 tape drive that uses medium-density formatting, use this device name:

```
/dev/rmt/0m
```

When you specify a tape device name, you can also type the letter "n" at the end of the name to indicate that the tape drive should not rewind after the backup is completed. For example:

```
/dev/rmt/0mn
```

Use the "no-rewind" option if you want to put more than one file onto the tape. If you run out of space during a backup, the tape does not rewind before the `ufsdump` command asks for a new tape. For a complete description of device naming conventions, see "Backup Device Names" on page 456.

Remote Tape or Diskette Drive

You specify a remote tape device or a remote diskette by using the syntax *host:device*. The `ufsdump` command writes to the remote device when superuser on the local system has access to the remote system. If you usually run the `ufsdump` command as superuser, the name of the local system must be included in the `/.rhosts` file on the remote system. If you specify the device as *user@host:device*, the `ufsdump` command tries to access the device on the remote system as the specified user. In this case, the specified user must be included in the `/.rhosts` file on the remote system.

Use the naming convention for the device that matches the operating system for the system on which the device resides, not the system from which you run the `ufsdump` command. If the drive is on a system that is running a previous SunOS release (for example, 4.1.1), use the SunOS 4.1 device name (for example, `/dev/rst0`). If the system is running Solaris software, use the SunOS 5.9 convention (for example, `/dev/rmt/0`).

Using Standard Output With the `ufsdump` Command

When you specify a dash (`-`) as the *dump-file* argument, the `ufsdump` command writes to standard output.

Note – The `-v` option (verify) does not work when the *dump-file* argument is standard output.

You can use the `ufsdump` and `ufsrestore` commands in a pipeline to copy a file system by writing to standard output with the `ufsdump` command and reading from standard input with the `ufsrestore` command. For example:

```
# ufsdump 0f - /dev/rdsk/c0t0d0s7 | (cd /home; ufsrestore xf -)
```

Specifying Files to Back Up

You must always include *filenames* as the last argument on the command line. This argument specifies the source or contents of the backup.

For a file system, specify the raw device file as follows:

`/dev/rdsk/`*c0t0d0s6*

You can specify the file system by its mount point directory (for example, `/export/home`), as long as an entry for it exists in the `/etc/vfstab` file.

For a complete description of device– naming conventions, see "Backup Device Names" on page 456.

For individual files or directories, type one or more names separated by spaces.

Note – When you use the `ufsdump` command to back up one or more directories or files (rather than a complete file system), a level 0 backup is done. Incremental backups do not apply.

Specifying Tape Characteristics

If you do not specify any tape characteristics, the `ufsdump` command uses a set of defaults. You can specify the tape cartridge (`c`), density (`d`), size (`s`), and number of tracks (`t`). Note that you can specify the options in any order, as long as the arguments that follow match the order of the options.

Limitations of the `ufsdump` Command

The `ufsdump` command cannot do the following:

- Automatically calculate the number of tapes or diskettes that are needed for backing up file systems. You can use the dry run mode (`S` option) to determine how much space is needed before actually backing up file systems.
- Provide built-in error checking to minimize problems when it backs up an active file system.
- Back up files that are remotely mounted from a server. Files on the server must be backed up on the server itself. Users are denied permission to run the `ufsdump` command on files they own that are located on a server.

Options and Arguments for the `ufsdump` Command

This section describes in detail the options and arguments for the `ufsdump` command. The syntax for the `ufsdump` command is as follows:

`/usr/sbin/ufsdump` *options arguments filenames*

options	Is a single string of one-letter option names.
arguments	Identifies option arguments and might consist of multiple strings. The option letters and their associated arguments must be in the same order.
filenames	Identifies the files to back up. These arguments must always come last, each separated by a space.

Default `ufsdump` Options

If you run the `ufsdump` command without any options, use this syntax:

```
# ufsdump filenames
```

The `ufsdump` command uses these options and arguments, by default:

```
ufsdump 9uf /dev/rmt/0 filenames
```

These options do a level 9 incremental backup to the default tape drive at its preferred density.

Options for the `ufsdump` Command

The following table describes the options for the `ufsdump` command.

TABLE 26–1 Options for the `ufsdump` Command

Option	Description
`0-9`	Dump level. Level 0 is for a full backup of the complete file system or file systems specified by *filenames*. Levels 1–9 are for incremental backups of files that have changed since the last lower-level backup.
`a` *archive-file*	Archive file. Specifies a file that stores (archives) a backup table of contents. The file can be understood only by the `ufsrestore` command. This command uses the table of contents to determine whether a file to be restored is present in a backup file, and if so, on which volume of the media the file resides.
`b` *factor*	Blocking factor. Specifies the number of 512-byte blocks to write to tape at a time.
`c`	Cartridge. Identifies the backup media as cartridge tape. When end-of-media detection applies, this option sets the block size to 126.
`d` *bpi*	Tape density. Specifies the tape density. Use this option only when the `ufsdump` command cannot detect the end of the media.
`D`	Diskette. Identifies the backup media as a diskette.
`f` *dump-file*	Dump file. Writes the files to the destination that is specified by *dump-file* instead of the default device. If the file is specified as *user@system:device*, the `ufsdump` command attempts to execute as the specified user on the remote system. The specified user must have a `/.rhosts` file on the remote system that allows the user who is invoking the command on the local system to access the remote system.
`l`	Autoload. Use this option if you have an autoloading (stackloader) tape drive. When the end of a tape is reached, this option takes the drive offline and waits up to two minutes for the tape drive to be ready again. If the drive is ready within two minutes, the autoload continues. If the drive is not ready after two minutes, autoload prompts the operator to load another tape.
`n`	Notify. When intervention is needed, this option sends a message to all terminals of all users in the `sys` group.
`o`	Offline. When the command is finished with a tape or diskette, this option takes the drive offline, rewinds (if tape), and if possible removes the media. For example, this option ejects a diskette or removes an 8-mm autoloaded tape.
`s` *size*	Size. Specifies the size of the backup media. For tapes, the size is specified in feet. For diskettes, the size is specified by the number of 1024–byte blocks.. Use this option only when the `ufsdump` command cannot detect the end of the media.

TABLE 26–1 Options for the `ufsdump` Command *(Continued)*

Option	Description
`S`	Size. Estimates the size of the backup. Determines the amount of space that is needed to perform the backup, without actually doing it. Outputs a single number that indicates the estimated size of the backup in bytes.
`t` *tracks*	Tracks. Specifies the number of tracks for a 1/4-inch cartridge tape. Use this option only when the `ufsdump` command cannot detect the end of the media.
`u`	Update. Updates the dump record. A completed backup of a file system adds an entry to the `/etc/dumpdates` file. The entry indicates the device name for the file system's disk slice, the dump level (0–9), and the date. No record is written when you do not use the `u` option or when you back up individual files or directories. If a record already exists for a backup at the same level, it is replaced.
`v`	Verify. After each tape or diskette is written, verifies the contents of the media against the source file system. If any discrepancies occur, prompts the operator to mount new media, then repeats the process. Use this option only on an unmounted file system, because any activity in the file system causes the `ufsdump` command to report discrepancies.
`w`	Warning. Lists the file systems that appear in the `/etc/dumpdates` file that have not been backed up within a day. When you use this option, all other options are ignored.
`W`	Warning with highlight. Shows all the file systems that appear in the `/etc/dumpdates` file and highlights those file systems that have not been backed up within a day. When you use this option, all other options are ignored.

Note – The `/etc/vfstab` file does not contain information about how often to back up a file system.

The `ufsdump` Command and Security Issues

If you are concerned about security, you should do the following:

- Require superuser access for the `ufsdump` command.

- Ensure superuser access entries are removed from `/.rhosts` files on clients and servers if you are doing centralized backups.

  For general information on security, see *System Administration Guide: Security Services*.

Options and Arguments for the `ufsrestore` Command

The syntax of the `ufsrestore` command is as follows:

`/usr/sbin/ufsrestore` *options arguments filenames*

options	Is a single string of one-letter option names. You must choose one and only one of these options: `i`, `r`, `R`, `t`, or `x`. The additional options listed in Table 26–3 are optional.
arguments	Follows the option string with the arguments that match the options. The option letters and their associated arguments must be in the same order.
filenames	Specifies the file or files to be restored as arguments to the `x` or `t` options. These arguments must always come last, separated by spaces.

You must use one (and only one) of the `ufsrestore` command options described in the following table.

TABLE 26–2 One Required Option for the `ufsrestore` Command

Option	Description
`i`	Interactive. Runs the `ufsrestore` command in interactive mode. In this mode, you can use a limited set of shell-like commands to browse the contents of the media and select individual files or directories to restore. For a list of interactive commands, see Table 26–4.
`r`	Recursive. Restores the entire contents of the media into the current working directory (which should be the top level of the file system). Information used to restore incremental backups on top of the full backup (for example, `restoresymtable`) is also included. To completely restore a file system, use this option to restore the full (level 0) backup and each subsequent incremental backup. Although this option is intended for a new file system (that was just created with the `newfs` command), files not on the backup media are preserved.

TABLE 26–2 One Required Option for the `ufsrestore` Command *(Continued)*

Option	Description
`R`	Resume restoring. Prompts for the volume from which to resume restoring and restarts from a checkpoint. You rerun the `ufsrestore` command with this option after a full restore (`r` option) is interrupted.
`x` [*filenames*]	Extract. Selectively restores the files you specify by the *filenames* argument. *filenames* can be a list of files and directories, each separated by a space. All files under a specified directory are restored unless you also use the `h` option. If you omit *filenames* or enter "`.`" for the root directory, all files on all volumes of the media (or from standard input) are restored. Existing files are overwritten, and warnings are displayed.
`t` [*filenames*]	Table of contents. Checks the files that are specified in the *filenames* argument against the media. For each file, lists the full file name and the inode number (if the file is found) or indicates that the file is not on the "volume" (meaning any volume in a multivolume backup). If you do not enter the *filenames* argument, all files on all volumes of the media are listed (without distinguishing on which volume files are located). If you also use the `h` option, only the directory files that are specified in *filenames*, not their contents, are checked and listed. The table of contents is read from the first volume of the media, or, if you use the `a` option, from the specified archive file. This option is mutually exclusive with the `x` and `r` options.

Additional `ufsrestore` options are described in the following table. These options are optional.

TABLE 26–3 Additional Options for the `ufsrestore` Command

Option	Description
`a` *archive-file* [*filenames*]	Archive file. Takes the backup table of contents from the specified *archive-file* instead of from the media (first volume). You can use this option with the `t`, `i`, or `x` options to see if files are on the media without having to mount any media. If you use this option with the `x` and interactive (`i`) extract options, you are prompted to mount the appropriate volume before extracting the file or files.
`b` *factor*	Blocking factor. Specifies number of 512-byte blocks read at a time from a tape. By default, the `ufsrestore` command tries to figure out the block size that was used when the tape was being written to.
`d`	Debug. Turns on debugging messages.

TABLE 26–3 Additional Options for the `ufsrestore` Command *(Continued)*

Option	Description
`f` *backup-file*	Backup file. Reads the files from the source indicated by *backup-file*, instead of from the default device file `/dev/rmt/0m`. If you use the `f` option, you must specify a value for *backup-file*. When *backup-file* is of the form *system:device*, the `ufsrestore` command reads from the remote device. You can also use the *backup-file* argument to specify a file on a local or remote disk. If the*backup-file* consistes of '-', the files are read from standard input.
`h`	Turns off directory expansion. Only the directory file you specify is extracted or listed.
`m`	Restores specified files into the current directory on the disk, regardless of where they are located in the backup hierarchy. Also, renames the specified files with their inode number. For example, if the current working directory is `/files`, a file in the backup named `./dready/fcs/test` with inode number 42 is restored as `/files/42`. This option is useful only when you are extracting a few files.
`s` *n*	Skip. Skips to the *n*th backup file on the media (first volume). This option is useful when you put more than one backup on a single tape.
`v`	Verbose. Displays the names and inode numbers of each file as it is restored.
`y`	Specifies that the command continues when errors occur while reading the media, trying to skip over bad blocks instead of stopping and asking whether to continue. This option tells the command to assume a yes response.

The following table describes `ufsrestore`'s interactive commands.

TABLE 26–4 Commands for Interactive Restore

Option	Description
`ls` [*directory-name*]	Lists the contents of either the current directory or the specified directory. Directories are marked by a `/` suffix. Entries in the current list to be restored (extracted) are marked by an `*` prefix. Inode numbers are shown if the verbose option (`v`) is used.
`cd` *directory-name*	Changes to the specified directory in the backup hierarchy.

TABLE 26–4 Commands for Interactive Restore *(Continued)*

Option	Description
`add` [*filename*]	Adds the current directory or the specified file or directory to the list of files to extract (restore). If you do not use the `h` option, all files in a specified directory and its subdirectories are added to the list. All the files you want to restore to a directory might not be on a single backup tape or diskette. You might need to restore from multiple backups at different levels to get the latest versions of all the files.
`delete` [*filename*]	Deletes the current directory or the specified file or directory from the list of files to extract (restore). If you do not use the `h` option, all files in the specified directory and its subdirectories are deleted from the list. The files and directories are deleted only from the extract list you are building. They are not deleted from the media or the file system.
`extract`	Extracts the files in the list and restores them relative to the current working directory on the disk. When you are asked for a volume number for a single-volume backup, specify `1`. If you are doing a multiple tape or multiple diskette restore and restoring a small number of files, start instead with the last tape or diskette.
`help`	Displays a list of commands that you can use in interactive mode.
`pwd`	Displays the path name of the current working directory in the backup hierarchy.
`q`	Quits interactive mode without restoring any additional files.
`setmodes`	Lets you set the mode for files to be restored to match the mode of the root directory of the file system from which they were backed up. You are prompted with: `set owner/mode for '.' [yn] ?` Type `y` (for yes) to set the mode (permissions, owner, times) of the current directory to match the root directory of the file system from which they were backed up. Use this mode when you restore a complete file system. Type `n` (for no) to leave the mode of the current directory unchanged. Use this mode when you restore part of a backup to a directory other than the directory from which the files were backed up.
`verbose`	Turns on or off the verbose option (which can also be typed as `v` on the command line outside of interactive mode). When verbose is on, the interactive `ls` command lists inode numbers, and the `ufsrestore` command displays information on each file as it is extracted.
`what`	Displays the backup header from the tape or diskette.

CHAPTER 27

Copying UFS Files and File Systems (Tasks)

This chapter describes how to copy UFS files and file systems to disk, tape, and diskettes by using various backup commands.

This is a list of the step-by-step instructions in this chapter.

Commands for Copying File Systems

When you need to back up and restore complete file systems, use the `ufsdump` and `ufsrestore` commands described in Chapter 26. When you want to copy or move individual files, portions of file systems, or complete file systems, you can use the procedures described in this chapter instead of the `ufsdump` and `ufsrestore` commands.

The following table describes when to use the various backup commands.

TABLE 27–1 When to Use Various Backup Commands

Task	Command	For More Information
Back up file systems to tape	`ufsdump`	"How to Backup a File System to Tape" on page 390
Create a file system snapshot	`fssnap`	Chapter 24
Restore file systems from tape	`ufsrestore`	"How to Restore a Complete File System" on page 415
Transport files to other systems	`pax`, `tar`, or `cpio`	"Copying Files and File Systems to Tape" on page 439
Copy files or file systems between disks	`dd`	"How to Copy a Disk (`dd`)" on page 435
Copy files to diskette	`tar`	"How to Copy Files to a Single Formatted Diskette (`tar`)" on page 452

The following table describes various backup and restore commands.

TABLE 27–2 Summary of Various Backup Commands

Command Name	Aware of File System Boundaries?	Supports Multiple Volume Backups?	Physical or Logical Copy?
`volcopy`	Yes	Yes	Physical
`tar`	No	No	Logical
`cpio`	No	Yes	Logical
`pax`	Yes	Yes	Logical
`dd`	Yes	No	Physical
`ufsdump`/`ufsrestore`	Yes	Yes	Logical

The following sections describe the advantages and disadvantages of each command. Also provided are step-by-step instructions and examples of how to use the commands.

Copying File Systems Between Disks

Two commands are used to copy file systems between disks:

- `volcopy`
- `dd`

For more information about `volcopy`, see the `volcopy`(1M).

The next section describes how to use the `dd` command to copy file systems between disks.

Making a Literal File System Copy

The `dd` command makes a literal (block-level) copy of a complete UFS file system to another file system or to a tape. By default, the `dd` command copies standard input to standard output.

Note – Do not use the `dd` command with variable-length tape drives without first specifying an appropriate block size.

You can specify a device name in place of standard input or standard output, or both. In this example, the contents of the diskette are copied to a file in the `/tmp` directory:

```
$ dd < /floppy/floppy0 > /tmp/output.file
2400+0 records in
2400+0 records out
```

The `dd` command reports on the number of blocks it reads and writes. The number after the + is a count of the partial blocks that were copied. The default block size is 512 bytes.

The `dd` command syntax is different from most other commands. Options are specified as *keyword=value* pairs, where *keyword* is the option you want to set and *value* is the argument for that option. For example, you can replace standard input and standard output with this syntax:

```
$ dd if=input-file of=output-file
```

To use the *keyword=value* pairs instead of the redirect symbols in the previous example, you would type the following:

```
$ dd if=/floppy/floppy0 of=/tmp/output.file
```

▼ How to Copy a Disk (`dd`)

Steps

1. **Make sure that the source disk and destination disk have the same disk geometry.**

2. **Become superuser or assume an equivalent role.**

3. **Create the `/reconfigure` file so the system will recognize the destination disk to be added when it reboots.**

```
# touch /reconfigure
```

4. **Shut down the system.**

   ```
   # init 0
   ```

5. **Attach the destination disk to the system.**

6. **Boot the system.**

   ```
   ok boot
   ```

7. **Copy the source disk to the destination disk.**

   ```
   # dd if=/dev/rdsk/device-name of=/dev/rdsk/device-name bs=block-size
   ```

`if=/dev/rdsk/`*device-name*	Represents the overlap slice of the master disk device, usually slice 2.
`of=/dev/rdsk/`*device-name*	Represents the overlap slice of the destination disk device, usually slice 2.
`bs=`*blocksize*	Identifies block size, such as 128 Kbytes or 256 Kbytes. A large block size value decreases the time it takes to copy the disk.

   For more information, see the `dd`(1M)

8. **Check the new file system.**

   ```
   # fsck /dev/rdsk/device-name
   ```

9. **Mount the destination disk's root (/) file system.**

   ```
   # mount /dev/dsk/device-name /mnt
   ```

10. **Change to the directory where the `/etc/vfstab` file is located.**

    ```
    # cd /mnt/etc
    ```

11. **Using a text editor, edit the destination disk's `/etc/vfstab` file to reference the correct device names.**

    For example, change all instances of `c0t3d0` to `c0t1d0`.

12. **Change to the destination disk's root (/) directory.**

    ```
    # cd /
    ```

13. **Unmount the destination disk's root (/) file system.**

    ```
    # umount /mnt
    ```

14. **Shut down the system.**

    ```
    # init 0
    ```

15. **Boot from the destination disk to single-user mode.**

```
# boot diskn -s
```

Note – The `installboot` command is not needed for the destination disk because the boot blocks are copied as part of the overlap slice.

16. **Unconfigure the destination disk.**

```
# sys-unconfig
```

The system is shut down after it is unconfigured.

17. **Boot from the destination disk again and provide its system information, such as host name, time zone, and so forth.**

```
# boot diskn
```

18. **After the system is booted, log in as superuser to verify the system information.**

```
hostname console login:
```

Example 27–1 Copying a Disk (`dd`)

This example shows how to copy the master disk `/dev/rdsk/c0t0d0s2` to the destination disk `/dev/rdsk/c0t2d0s2`.

```
# touch /reconfigure
# init 0
ok boot
# dd if=/dev/rdsk/c0t0d0s2 of=/dev/rdsk/c0t2d0s2 bs=128k
# fsck /dev/rdsk/c0t2d0s2
# mount /dev/dsk/c0t2d0s2 /mnt
# cd /mnt/etc
# vi vfstab
(Modify entries for the new disk)
# cd /
# umount /mnt
# init 0
# boot disk2 -s
# sys-unconfig
# boot disk2
```

Copying Directories Between File Systems (`cpio` Command)

You can use the `cpio` (copy in and out) command to copy individual files, groups of files, or complete file systems. This section describes how to use the `cpio` command to copy complete file systems.

The `cpio` command is an archiving program that copies a list of files into a single, large output file. This command inserts headers between the individual files to facilitate recovery. You can use the `cpio` command to copy complete file systems to another slice, another system, or to a media device, such as a tape or diskette.

Because the `cpio` command recognizes end-of-media and prompts you to insert another volume, it is the most effective command, other than `ufsdump`, to use to create archives that require multiple tapes or diskettes.

With the `cpio` command, you frequently use the `ls` and `find` commands to list and select the files you want to copy, and then to pipe the output to the `cpio` command.

▼ How to Copy Directories Between File Systems (`cpio`)

Steps

1. **Become superuser or assume an equivalent role.**
2. **Change to the appropriate directory.**

   ```
   # cd filesystem1
   ```

3. **Copy the directory tree from *filesystem1* to *filesystem2* by using a combination of the `find` and `cpio` commands.**

   ```
   # find . -print -depth | cpio -pdm filesystem2
   ```

`.`	Starts in the current working directory.
`-print`	Prints the file names.
`-depth`	Descends the directory hierarchy and prints file names from the bottom up.
`-p`	Creates a list of files.
`-d`	Creates directories as needed.

`-m`	Sets the correct modification times on directories.

For more information, see the `cpio`(1) man page.

The files from the directory name you specify are copied. The symbolic links are preserved.

You might also specify the `-u` option. This option forces an unconditional copy. Otherwise, older files do not replace newer files. This option might be useful if you want an exact copy of a directory, and some of the files being copied might already exist in the target directory.

4. **Verify that the copy was successful by displaying the contents of the destination directory.**

```
# cd filesystem2
# ls
```

5. **If appropriate, remove the source directory.**

```
# rm -rf filesystem1
```

Example 27–2 Copying Directories Between File Systems (`cpio`)

```
# cd /data1
# find . -print -depth | cpio -pdm /data2
19013 blocks
# cd /data2
# ls
# rm -rf /data1
```

Copying Files and File Systems to Tape

You can use the `tar`, `pax`, and `cpio` commands to copy files and file systems to tape. The command that you choose depends on how much flexibility and precision you require for the copy. Because all three commands use the raw device, you do not need to format or make a file system on tapes before you use them.

TABLE 27–3 Advantages and Disadvantages of `tar`, `pax`, and `cpio` Commands

Command	Function	Advantages	Disadvantages
`tar`	Use to copy files and directory subtrees to a single tape.	■ Available on most UNIX operating systems ■ Public domain versions are readily available	■ Is not aware of file system boundaries ■ Full path-name length cannot exceed 255 characters ■ Does not copy empty directories or special files such as device files ■ Cannot be used to create multiple tape volumes
`pax`	Use to copy files, special files, or file systems that require multiple tape volumes. Or, use when you want to copy files to and from POSIX-compliant systems	■ Better portability than the `tar` or `cpio` commands for POSIX-compliant systems ■ Multiple vendor support	Same disadvantages as the `tar` command, except that the `pax` command can create multiple tape volumes
`cpio`	Use to copy files, special files, or file systems that require multiple tape volumes. Or, use when you want to copy files from SunOS 5.9 systems to SunOS 4.0/4.1 systems	■ Packs data onto tape more efficiently than the `tar` command ■ Skips over any bad spots in a tape when restoring ■ Provides options for writing files with different header formats, such as (`tar`, `ustar`, `crc`, `odc`, `bar`), for portability between different system types ■ Creates multiple tape volumes	The command syntax is more difficult than the `tar` or `pax` commands

The tape drive and device name that you use depend on the hardware configuration for each system. For more information about tape device names, see "Choosing Which Media to Use" on page 455.

Copying Files to Tape (`tar` Command)

Here is information that you should know before you copy files to tape with the `tar` command:

- Copying files to a tape with the `-c` option to the `tar` command destroys any files already on the tape at or beyond the current tape position.
- You can use file name substitution wildcards (`?` and `*`) as part of the file names that you specify when copying files. For example, to copy all documents with a `.doc` suffix, type `*.doc` as the file name argument.
- You cannot use file name substitution wildcards when you extract files from a `tar` archive.

▼ How to Copy Files to a Tape (`tar`)

Steps

1. **Change to the directory that contains the files you want to copy.**
2. **Insert a write-enabled tape into the tape drive.**
3. **Copy the files to tape.**

```
$ tar cvf /dev/rmt/n filenames
```

`c`	Indicates that you want to create an archive.
`v`	Displays the name of each file as it is archived.
`f /dev/rmt/n`	Indicates that the archive should be written to the specified device or file.
filenames	Indicates the files and directories that you want to copy. Separate multiple files with spaces.

The file names that you specify are copied to the tape, overwriting any existing files on the tape.

4. **Remove the tape from the drive. Write the names of the files on the tape label.**
5. **Verify that the files you copied are on the tape.**

```
$ tar tvf /dev/rmt/n
```

For more information on listing files on a `tar` tape, see "How to List the Files on a Tape (`tar`)" on page 442.

Example 27–3 Copying Files to a Tape (`tar`)

The following example shows how to copy three files to the tape in tape drive 0.

```
$ cd /export/home/kryten
$ ls reports
reportA reportB reportC
$ tar cvf /dev/rmt/0 reports
a reports/ 0 tape blocks
a reports/reportA 59 tape blocks
a reports/reportB 61 tape blocks
a reports/reportC 63 tape blocks
$ tar tvf /dev/rmt/0
```

▼ How to List the Files on a Tape (`tar`)

Steps

1. **Insert a tape into the tape drive.**
2. **Display the tape contents.**

```
$ tar tvf /dev/rmt/n
```

`t`	Lists the table of contents for the files on the tape.
`v`	Used with the `t` option, and provides detailed information about the files on the tape.
`f /dev/rmt/`*n*	Indicates the tape device.

Example 27–4 Listing the Files on a Tape (`tar`)

The following example shows a listing of files on the tape in drive 0.

```
$ tar tvf /dev/rmt/0
drwxr-xr-x 1001/10       0 Oct  7 08:18 2003 reports/
-r--r--r-- 1001/10     382 Oct  7 08:18 2003 reports/reportA
-r--r--r-- 1001/10     382 Oct  7 08:18 2003 reports/reportB
-r--r--r-- 1001/10     382 Oct  7 08:18 2003 reports/reportC
```

▼ How to Retrieve Files From a Tape (`tar`)

Steps

1. **Change to the directory where you want to put the files.**
2. **Insert the tape into the tape drive.**
3. **Retrieve the files from the tape.**

```
$ tar xvf /dev/rmt/n  [filenames]
```

`x`	Indicates that the files should be extracted from the specified archive file. All files on the tape in the specified drive are copied to the current directory.
`v`	Displays the name of each file as it is retrieved.
`f /dev/rmt/`*n*	Indicates the tape device that contains the archive.
filenames	Specifies a file to retrieve. Separate multiple files with spaces.

For more information, see the `tar`(1) man page.

4. **Verify that the files are copied.**

```
$ ls -l
```

Example 27–5 Retrieving the Files on a Tape (`tar`)

The following example shows how to retrieve all the files from the tape in drive 0.

```
$ cd /var/tmp
$ tar xvf /dev/rmt/0
x reports/, 0 bytes, 0 tape blocks
x reports/reportA, 0 bytes, 0 tape blocks
x reports/reportB, 0 bytes, 0 tape blocks
x reports/reportC, 0 bytes, 0 tape blocks
x reports/reportD, 0 bytes, 0 tape blocks
$ ls -l
```

Troubleshooting

Note – The names of the files extracted from the tape must exactly match the names of the files that are stored on the archive. If you have any doubts about the names or paths of the files, first list the files on the tape. For instructions on listing the files on the tape, see "How to List the Files on a Tape (`tar`)" on page 442.

Copying Files to a Tape With the `pax` Command

▼ How to Copy Files to a Tape (`pax`)

Steps

1. **Change to the directory that contains the files you want to copy.**
2. **Insert a write-enabled tape into the tape drive.**
3. **Copy the files to tape.**

   ```
   $ pax -w -f /dev/rmt/n filenames
   ```

`-w`	Enables the write mode.
`-f /dev/rmt/`*n*	Identifies the tape drive.
filenames	Indicates the files and directories that you want to copy. Separate multiple files with spaces.

   For more information, see the `pax`(1) man page.
4. **Verify that the files are copied to tape.**

   ```
   $ pax -f /dev/rmt/n
   ```
5. **Remove the tape from the drive. Write the names of the files on the tape label.**

Example 27–6 Copying Files to a Tape (`pax`)

The following example shows how to use the `pax` command to copy all the files in the current directory.

```
$ pax -w -f /dev/rmt/0 .
$ pax -f /dev/rmt/0
filea fileb filec
```

Copying Files to Tape With the `cpio` Command

▼ How to Copy All Files in a Directory to a Tape (`cpio`)

Steps

1. **Change to the directory that contains the files you want to copy.**
2. **Insert a write-enabled tape into the tape drive.**
3. **Copy the files to tape.**

   ```
   $ ls | cpio -oc > /dev/rmt/n
   ```

`ls`	Provides the `cpio` command with a list of file names.
`cpio -oc`	Specifies that the `cpio` command should operate in copy-out mode (`-o`) and write header information in ASCII character format (`-c`). These options ensure portability to other vendor's systems.
`> /dev/rmt/`*n*	Specifies the output file.

   All files in the directory are copied to the tape in the drive you specify, overwriting any existing files on the tape. The total number of blocks that are copied is shown.

4. **Verify that the files are copied to tape.**

   ```
   $ cpio -civt < /dev/rmt/n
   ```

`-c`	Specifies that the `cpio` command should read files in ASCII character format.
`-i`	Specifies that the `cpio` command should operate in copy-in mode, even though the command is only listing files at this point.
`-v`	Displays the output in a format that is similar to the output from the `ls -l` command.
`-t`	Lists the table of contents for the files on the tape in the tape drive that you specify.
`< /dev/rmt/`*n*	Specifies the input file of an existing `cpio` archive.

5. **Remove the tape from the drive. Write the names of the files on the tape label.**

Example 27–7 Copying All Files in a Directory to a Tape (`cpio`)

The following example shows how to copy all of the files in the `/export/home/kryten` directory to the tape in tape drive 0.

```
$ cd /export/home/kryten
$ ls | cpio -oc > /dev/rmt/0
16 blocks
$ cpio -civt < /dev/rmt/0
-r--r--r--    1 kryten   staff         76 Oct  7 08:17 2003, filea
-r--r--r--    1 kryten   staff         76 Oct  7 08:17 2003, fileb
-r--r--r--    1 kryten   staff         76 Oct  7 08:17 2003, filec
drwxr-xr-x    2 kryten   staff          0 Oct  7 08:17 2003, letters
drwxr-xr-x    2 kryten   staff          0 Oct  7 08:18 2003, reports
16 blocks
$
```

▼ How to List the Files on a Tape (`cpio`)

Note – Listing the table of contents on a tape takes a long time because the `cpio` command must process the entire archive.

Steps

1. **Insert an archive tape into the tape drive.**
2. **List the files on the tape.**

   ```
   $ cpio -civt < /dev/rmt/n
   ```

Example 27–8 Listing the Files on a Tape (`cpio`)

The following example shows how to list the files on the tape in drive 0.

```
$ cpio -civt < /dev/rmt/0
-r--r--r--    1 kryten   staff         76 Oct  7 08:17 2003, filea
-r--r--r--    1 kryten   staff         76 Oct  7 08:17 2003, fileb
-r--r--r--    1 kryten   staff         76 Oct  7 08:17 2003, filec
drwxr-xr-x    2 kryten   staff          0 Oct  7 08:17 2003, letters
drwxr-xr-x    2 kryten   staff          0 Oct  7 08:18 2003, reports
16 blocks
```

▼ How to Retrieve All Files From a Tape (`cpio`)

If the archive was created using relative path names, the input files are built as a directory within the current directory when you retrieve the files. If, however, the archive was created with absolute path names, the same absolute paths are used to re-create the file on your system.

Caution – The use of absolute path names can be dangerous because you might overwrite existing files on your system.

Steps

1. **Change to the directory where you want to put the files.**
2. **Insert the tape into the tape drive.**
3. **Extract all files from the tape.**

   ```
   $ cpio -icvd < /dev/rmt/n
   ```

`-i`	Extracts files from standard input.
`-c`	Specifies that the `cpio` command should read files in ASCII character format.
`-v`	Displays the files as they are retrieved in a format that is similar to the output from the `ls` command.
`-d`	Creates directories as needed.
`< /dev/rmt/`*n*	Specifies the output file.

4. **Verify that the files were copied.**

   ```
   $ ls -l
   ```

Example 27–9 Retrieving All Files From a Tape (`cpio`)

The following example shows how to retrieve all files from the tape in drive 0.

```
$ cd /var/tmp
cpio -icvd < /dev/rmt/0
answers
sc.directives
tests
8 blocks
$ ls -l
```

▼ How to Retrieve Specific Files From a Tape (`cpio`)

Steps

1. **Change to the directory where you want to put the files.**
2. **Insert the tape into the tape drive.**
3. **Retrieve a subset of files from the tape.**

   ```
   $ cpio -icv "*file" < /dev/rmt/n
   ```

`-i`	Extracts files from standard input.
`-c`	Specifies that the `cpio` command should read headers in ASCII character format.
`-v`	Displays the files as they are retrieved in a format that is similar to the output from the `ls` command.
`"*`*file*`"`	Specifies that all files that match the pattern are copied to the current directory. You can specify multiple patterns, but each pattern must be enclosed in double quotation marks.
`< /dev/rmt/`*n*	Specifies the input file.

   For more information, see the `cpio`(1) man page.

4. **Verify that the files were copied.**

   ```
   $ ls -l
   ```

Example 27–10 Retrieving Specific Files From a Tape (`cpio`)

The following example shows how to retrieve all files with the `chapter` suffix from the tape in drive 0.

```
$ cd /home/smith/Book
$ cpio -icv "*chapter" < /dev/rmt/0
Boot.chapter
Directory.chapter
Install.chapter
Intro.chapter
31 blocks
$ ls -l
```

Copying Files to a Remote Tape Device

▼ How to Copy Files to a Remote Tape Device (`tar` and `dd`)

Steps

1. **The following prerequisites must be met to use a remote tape drive:**

   a. **The local host name and optionally, the user name of the user doing the copy, must appear in the remote system's `/etc/hosts.equiv` file. Or, the user doing the copy must have his or her home directory accessible on the remote machine, and have the local machine name in `$HOME/.rhosts`.**

      For more information, see the `hosts.equiv`(4) man page.

   b. **An entry for the remote system must be in the local system's `/etc/inet/hosts` file or in the name service `hosts` file.**

2. **To test whether you have the appropriate permission to execute a remote command, try the following:**

   ```
   $ rsh remotehost echo test
   ```

   If `test` is echoed back to you, you have permission to execute remote commands. If `Permission denied` is echoed back to you, check your setup as described in step 1.

3. **Change to the directory where you want to put the files.**

4. **Insert the tape into the tape drive.**

5. **Copy the files to a remote tape drive.**

   ```
   $ tar cvf - filenames | rsh remote-host dd of=/dev/rmt/n obs=block-size
   ```

`tar cf`	Creates a tape archive, lists the files as they are archived, and specifies the tape device.
`v`	Provides additional information about the tar file entries.
- (Hyphen)	Represents a placeholder for the tape device.
filenames	Identifies the files to be copied. Separate multiple files with spaces.
| *rsh remote-host*	Pipes the `tar` command's output to a remote shell.

`dd of=/dev/rmt/`*n*	Represents the output device.
`obs=`*block-size*	Represents the blocking factor.

6. **Remove the tape from the drive. Write the names of the files on the tape label.**

Example 27–11 Copying Files to a Remote Tape Drive (`tar` and `dd`)

```
# tar cvf - * | rsh mercury dd of=/dev/rmt/0 obs=126b
a answers/ 0 tape blocks
a answers/test129 1 tape blocks
a sc.directives/ 0 tape blocks
a sc.directives/sc.190089 1 tape blocks
a tests/ 0 tape blocks
a tests/test131 1 tape blocks
6+9 records in
0+1 records out
```

▼ How to Extract Files From a Remote Tape Device

Steps

1. **Insert the tape into the tape drive.**
2. **Change to a temporary directory.**

   ```
   $ cd /var/tmp
   ```

3. **Extract the files from a remote tape device.**

   ```
   $ rsh remote-host dd if=/dev/rmt/n | tar xvBpf -
   ```

`rsh` *remote-host*	Indicates a remote shell that is started to extract the files from the tape device by using the `dd` command.
`dd if=/dev/rmt/`*n*	Indicates the input device.
`| tar xvBpf -`	Pipes the output of the `dd` command to the `tar` command, which is used to restore the files.

4. **Verify that the files have been extracted.**

   ```
   $ ls -l /var/tmp
   ```

Example 27–12 Extracting Files From a Remote Tape Drive

```
$ cd /var/tmp
$ rsh mercury dd if=/dev/rmt/0 | tar xvBpf -
x answers/, 0 bytes, 0 tape blocks
x answers/test129, 48 bytes, 1 tape blocks
```

```
20+0 records in
20+0 records out
x sc.directives/, 0 bytes, 0 tape blocks
x sc.directives/sc.190089, 77 bytes, 1 tape blocks
x tests/, 0 bytes, 0 tape blocks
x tests/test131, 84 bytes, 1 tape blocks
$ ls -l
```

Copying Files and File Systems to Diskette

Before you can copy files or file systems to diskette, you must format the diskette. For information on how to format a diskette, see Chapter 3.

Use the `tar` command to copy UFS files to a single formatted diskette.

Use the `cpio` command if you need to copy UFS files to multiple formatted diskettes. The `cpio` command recognizes end-of-media and prompts you to insert the next diskette.

What You Should Know When Copying Files to Diskettes

- Copying files to a formatted diskette by using the `tar -c` command destroys any files that are already on the diskette.
- A diskette that contains a `tar` image is not mountable.
- If you need a multiple-volume interchange utility, use the `cpio` command. The `tar` command is only a single-volume utility.

For more information, see the `tar`(1)man page.

▼ How to Copy Files to a Single Formatted Diskette (`tar`)

Steps

1. **Change to the directory that contains the files you want to copy.**
2. **Insert a formatted diskette that is not write-protected into the drive.**
3. **Make the diskette available.**

   ```
   $ volcheck
   ```

4. **Reformat the diskette, if necessary.**

   ```
   $ rmformat -U /dev/rdiskette
   Formatting will erase all the data on disk.
   Do you want to continue? (y/n)y
   ```

5. **Copy the files to diskette.**

   ```
   $ tar cvf /vol/dev/aliases/floppy0 filenames
   ```

   The file names that you specify are copied to the diskette, overwriting any existing files on the diskette.

6. **Verify that the files were copied.**

   ```
   $ tar tvf /vol/dev/aliases/floppy0
   ```

   For more information on listing files, see "How to List the Files on a Diskette (`tar`)" on page 453.

7. **Remove the diskette from the drive.**
8. **Write the names of the files on the diskette label.**

Example 27–13 Copying Files to a Single Formatted Diskette (`tar`)

The following example shows how to copy files named `evaluation*` to a diskette.

```
$ cd /home/smith
$ volcheck
$ ls evaluation*
evaluation.doc   evaluation.doc.backup
$ tar cvf /vol/dev/aliases/floppy0 evaluation*
a evaluation.doc 86 blocks
a evaluation.doc.backup 84 blocks
$ tar tvf /vol/dev/aliases/floppy0
```

▼ How to List the Files on a Diskette (`tar`)

Steps

1. **Insert a diskette into the drive.**
2. **Make the diskette available.**

   ```
   $ volcheck
   ```

3. **List the files on a diskette.**

   ```
   $ tar tvf /vol/dev/aliases/floppy0
   ```

Example 27–14 Listing the Files on a Diskette (`tar`)

The following example shows how to list the files on a diskette.

```
$ volcheck
$ tar tvf /vol/dev/aliases/floppy0
rw-rw-rw-6693/10  44032 Jun  9 15:45 evaluation.doc
rw-rw-rw-6693/10  43008 Jun  9 15:55 evaluation.doc.backup
$
```

▼ How to Retrieve Files From a Diskette (`tar`)

Steps

1. **Change to the directory where you want to put the files.**
2. **Insert the diskette into the drive.**
3. **Make the diskette available.**

   ```
   $ volcheck
   ```

4. **Retrieve files from the diskette.**

   ```
   $ tar xvf /vol/dev/aliases/floppy0
   ```

   All files on the diskette are copied to the current directory.

5. **Verify that the files have been retrieved.**

   ```
   $ ls -l
   ```

6. **Remove the diskette from the drive.**

Example 27–15 Retrieving Files From a Diskette (`tar`)

The following example shows how to retrieve all the files from a diskette.

```
$ cd /home/smith/Evaluations
$ volcheck
$ tar xvf /vol/dev/aliases/floppy0
x evaluation.doc, 44032 bytes, 86 tape blocks
x evaluation.doc.backup, 43008 bytes, 84 tape blocks
$ ls -l
```

The following example shows how to retrieve an individual file from a diskette. The file is extracted from the diskette and placed in the current working directory.

```
$ volcheck
$ tar xvf /vol/dev/aliases/floppy0 evaluation.doc
x evaluation.doc, 44032 bytes, 86 tape blocks
$ ls -l
```

How to Archive Files to Multiple Diskettes

If you are copying large files onto diskettes, you want to be prompted to replace a full diskette with another formatted diskette. The `cpio` command provides this capability. The `cpio` commands you use are the same that you would use to copy files to tape, except you would specify `/vol/dev/aliases/floppy0` as the device instead of the tape device name.

For information on how to use the `cpio` command, see "How to Copy All Files in a Directory to a Tape (`cpio`)" on page 445.

CHAPTER **28**

Managing Tape Drives (Tasks)

This chapter describes how to manage tape drives in the Solaris™ Operating System.

This is a list of the step-by-step instructions in this chapter.

- "How to Display Tape Drive Status" on page 458
- "How to Retension a Magnetic Tape Cartridge" on page 459
- "How to Rewind a Magnetic Tape Cartridge" on page 460

This is a list of overview information in this chapter:

- "Choosing Which Media to Use" on page 455
- "Backup Device Names" on page 456
- "Displaying Tape Drive Status" on page 458
- "Guidelines for Drive Maintenance and Media Handling" on page 460

Choosing Which Media to Use

You typically back up Solaris systems by using the following tape media:

- 1/2-inch reel tape
- 1/4-inch streaming cartridge tape
- 8-mm cartridge tape
- 4-mm cartridge tape (DAT)

You can perform backups with diskettes, but doing so is time-consuming and cumbersome.

The media that you choose depends on the availability of the equipment that supports it and of the media (usually tape) that you use to store the files. Although you must do the backup from a local system, you can write the files to a remote device.

The following table shows typical tape devices that are used for backing up file systems. shows The storage capacity for each device depends on the type of drive and the data being written to the tape.

TABLE 28–1 Media Storage Capacities

Backup Media	Storage Capacity
1/2-inch reel tape	140 Mbytes (6250 bpi)
2.5-Gbyte 1/4–inch cartridge (QIC) tape	2.5 Gbytes
DDS3 4-mm cartridge tape (DAT)	12–24 Gbytes
14-Gbyte 8-mm cartridge tape	14 Gbytes
DLT 7000 1/2-inch cartridge tape	35–70 Gbytes

Backup Device Names

You specify a tape or diskette to use for backup by supplying a logical device name. This name points to the subdirectory that contains the “raw” device file and includes the logical unit number of the drive. Tape drive naming conventions use a logical, not a physical, device name. The following table shows this naming convention.

TABLE 28–2 Basic Device Names for Backup Devices

Device Type	Name
Tape	`/dev/rmt/`*n*
Diskette	`/vol/dev/rdiskette0/unlabeled`

In general, you specify a tape device as shown in the following figure.

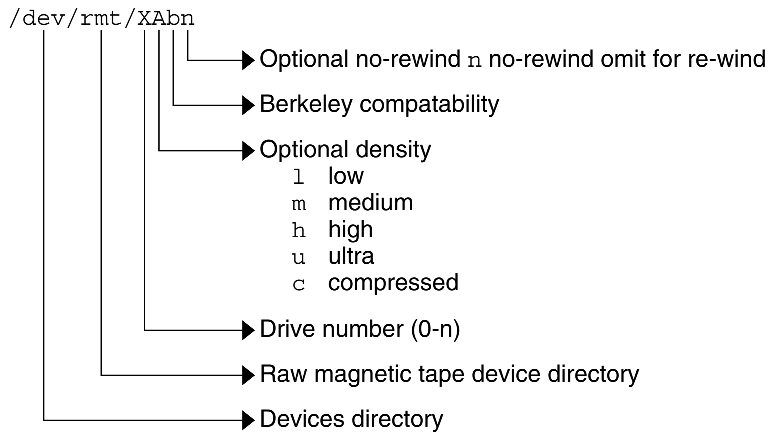

FIGURE 28–1 Tape Drive Device Names

If you don't specify the density, a tape drive typically writes at its "preferred" density. The preferred density usually means the highest density the tape drive supports. Most SCSI drives can automatically detect the density or format on the tape and read it accordingly. To determine the different densities that are supported for a drive, look at the `/dev/rmt` subdirectory. This subdirectory includes the set of tape device files that support different output densities for each tape.

Also, a SCSI controller can have a maximum of seven SCSI tape drives.

Specifying the Rewind Option for a Tape Drive

Normally, you specify a tape drive by its logical unit number, which can run from 0 to *n*. The following table describes how to specify tape device names with a rewind or a no rewind option.

TABLE 28–3 Specifying Rewind or No-Rewind for a Tape Drive

Drive and Rewind Value	Use This Option
First drive, rewind	`/dev/rmt/0`
First drive, no rewind	`/dev/rmt/0n`
Second drive, rewind	`/dev/rmt/1`
Second drive, no rewind	`/dev/rmt/1n`

Specifying Different Densities for a Tape Drive

By default, the drive writes at its “preferred” density, which is usually the highest density the tape drive supports. If you do not specify a tape device, the command writes to drive number 0 at the default density the device supports.

To transport a tape to a system whose tape drive supports only a certain density, specify a device name that writes at the desired density. The following table describes how to specify different densities for a tape drive.

TABLE 28–4 Specifying Different Densities for a Tape Drive

Drive, Density, and Rewind Value	Use This Option
First drive, low density, rewind	`/dev/rmt/0l`
First drive, low density, no rewind	`/dev/rmt/0ln`
Second drive, medium density, rewind	`/dev/rmt/1m`
Second drive, medium density, no rewind	`/dev/rmt/1mn`

The additional density values are shown in “Backup Device Names” on page 456.

Displaying Tape Drive Status

You can use the `status` option with the `mt` command to get status information about tape drives. The `mt` command reports information about any tape drives that are described in the `/kernel/drv/st.conf` file.

▼ How to Display Tape Drive Status

Steps

1. **Load a tape into the drive you want information about.**
2. **Display the tape drive status.**

   ```
   # mt -f /dev/rmt/n status
   ```

3. **Repeat steps 1-2, substituting tape drive numbers 0, 1, 2, 3, and so on to display information about all available tape drives.**

Example 28–1 Displaying Tape Drive Status

The following example shows the status for a QIC-150 tape drive (`/dev/rmt/0`):

```
$ mt -f /dev/rmt/0 status
Archive QIC-150 tape drive:
   sense key(0x0)= No Additional Sense   residual= 0   retries= 0
   file no= 0   block no= 0
```

The following example shows the status for an Exabyte tape drive (`/dev/rmt/1`):

```
$ mt -f /dev/rmt/1 status
Exabyte EXB-8200 8mm tape drive:
sense key(0x0)= NO Additional Sense residual= 0  retries= 0
file no= 0   block no= 0
```

The following example shows a quick way to poll a system and locate all of its tape drives:

```
$ for drive in 0 1 2 3 4 5 6 7
> do
> mt -f /dev/rmt/$drive status
> done
Archive QIC-150 tape drive:
   sense key(0x0)= No Additional Sense   residual= 0   retries= 0
   file no= 0   block no= 0
/dev/rmt/1: No such file or directory
/dev/rmt/2: No such file or directory
/dev/rmt/3: No such file or directory
/dev/rmt/4: No such file or directory
/dev/rmt/5: No such file or directory
/dev/rmt/6: No such file or directory
/dev/rmt/7: No such file or directory
$
```

Handling Magnetic Tape Cartridges

If errors occur when a tape is being read, you can retension the tape, clean the tape drive, and then try again.

How to Retension a Magnetic Tape Cartridge

Retension a magnetic tape cartridge with the `mt` command.

For example:

```
$ mt -f /dev/rmt/1 retension
$
```

Note – Do not retension non-QIC tape drives.

How to Rewind a Magnetic Tape Cartridge

To rewind a magnetic tape cartridge, use the `mt` command.

For example:

```
$ mt -f /dev/rmt/1 rewind
$
```

Guidelines for Drive Maintenance and Media Handling

A backup tape that cannot be read is useless. So, periodically clean and check your tape drives to ensure correct operation. See your hardware manuals for instructions on procedures for cleaning a tape drive. You can check your tape hardware by doing either of the following:

- Copying some files to the tape, reading the files back, and then comparing the original files with the copied files.
- Using the `-v` option of the `ufsdump` command to verify the contents of the media with the source file system. The file system must be unmounted or completely idle for the `-v` option to be effective.

Be aware that hardware can fail in ways that the system does not report.

Always label your tapes after a backup. If you are using a backup strategy similar to the strategies suggested in Chapter 22, you should indicate on the label "Tape A," "Tape B," and so forth. This label should never change. Every time you do a backup, make another tape label that contains the following information:

- The backup date
- The name of the machine and file system backed up
- The backup level
- The tape number (1 of *n*, if the backup spans multiple volumes)
- Any information specific to your site

Store your tapes in a dust-free safe location, away from magnetic equipment. Some sites store archived tapes in fireproof cabinets at remote locations.

You should create and maintain a log that tracks which media (tape volume) stores each job (backup) and the location of each backed-up file.

Index