Disk and File System Administration

Student Guide

Sun Microsystems Computer Corporation
Technical Education Services
MS UMIL07-14
2550 Garcia Avenue
Mountain View, CA 94043
U.S.A.

Part Number SA-285
Revision B, July 1993

Contents

Disk Device Names 1

Objectives

Upon completion of this lesson, you will be able to:

- Identify the device name used by administrators to reference disk devices and explain when it is used in the Solaris® 2.*x* environment.
- Recall two tasks that system administrators do that reference disk devices (and partitions) by their logical device names.
- Determine the type(s) of disk devices and disk device interfaces on your system using the `format` utility.

References

Solaris 2.x Handbook for SMCC Peripherals

Introduction

To administer the disk devices on your system, you should be familiar with the hardware and corresponding device naming conventions that are used to identify your disks.

Disk devices (and partitions) are referenced by their *logical device name*s when you:

- Move a disk from one system to another.
- Add a new disk to the system.
- Access (or mount) a file system residing on a local disk.
- Back up a local file system.

This lesson describes the different ways in which disk devices are referenced in the Solaris 2.*x* environment.

Disk Device Interfaces

Sun™ supports the following interfaces for disk devices:

- Small computer system interface (SCSI)
- Intelligent peripheral interface (IPI)
- Xylogics (451 or 7053) controller

The above interfaces (or controllers) are used to communicate with their connected disks, tapes, and compact disc read-only memory (CD-ROM) devices on Sun systems. However, only the SCSI interface is used to connect CD-ROM devices.

- SCSI devices have embedded controllers that are connected to a SCSI host adapter that is connected to an SBUS or VME peripheral bus.
- IPI devices are connected to IPI controllers that are connected to the VMEbus peripheral bus.
- Storage module device (SMD) and enhanced storage module device (ESMD) disks are connected to Xylogics controllers that are connected to the VMEbus peripheral bus.

A *bus* is a channel or pathway between hardware devices.

Disk Device Interfaces

You can tell which disk devices and disk device interfaces are connected to your system by using the `format` utility, the `dmesg` command, or the `prtconf` command.

Example:

```
# format
Searching for disks...done

AVAILABLE DISK SELECTIONS:
 0. c0t3d0 <SUN0424 cyl 1151 alt 2 hd 9 sec 80>
/iommu@f,e0000000/sbus@f,e0001000/espdma@f,400000/esp@f,800000/sd@3,0
Specify disk (enter its number):
```

The above `format` example identifies one SCSI (`sd@...`) disk device connected to the SCSI host adapter (`esp@...`), which is connected to SCSI DMA controller (`espdma@...`), which is connected to the SBUS interface (`sbus@1...`).

IPI and Xylogic disk devices are prefaced with `ip` and `xd` respectively.

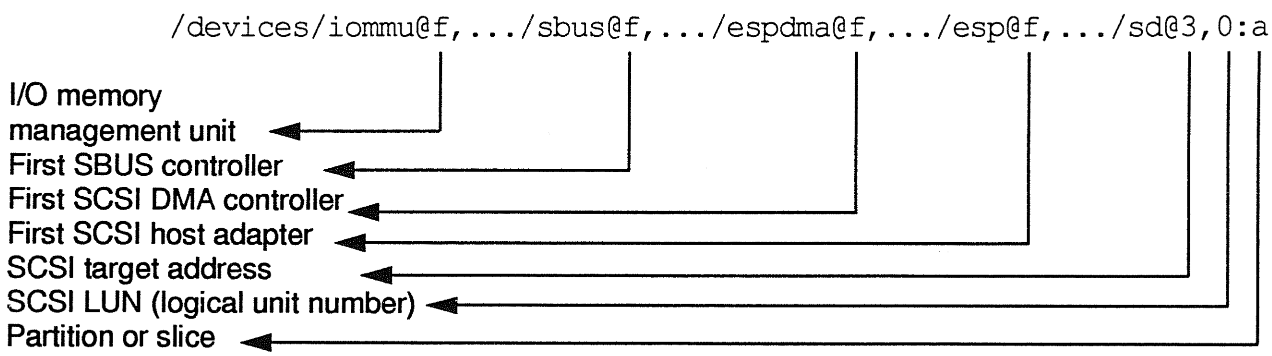
The `format` output displays both the logical and physical device names.

Physical Disk Device Names Review

Physical device names describe every device connected to the system. Their corresponding device files are stored in the `/devices` directory.

SCSI Devices

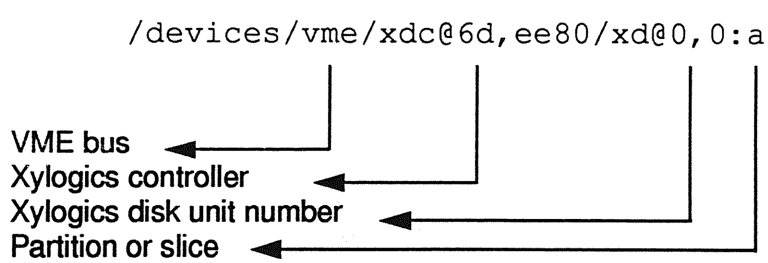
The example below describes the first SCSI disk (target address 3) connected to the first SCSI host adapter.



The physical name for the SCSI host adapter is `esp@0`. Each SCSI host adapter supports seven devices (0-7), and the target address differentiates one device from another. A second SCSI host adapter would be identified with `esp@1`.

Xylogics Devices

The example below describes the first Xylogics disk (drive unit 0) connected to the first external Xylogics controller.

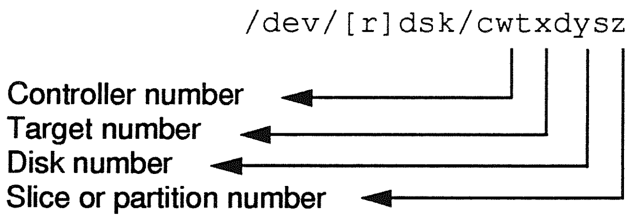


Logical Disk Device Names

The logical disk device name is specified by system administrators (and users) when using disk and file system-related commands.

Bus-Oriented Controllers

Sun systems use the following naming convention to describe the logical device name for a disk device connected to a bus-oriented controller (SCSI and IPI).



Example:

```
/dev/dsk/c0t3d0s0 ->
/devices/iommu@f,e0000000/sbus@f,e0001000/espdma@f,400000/esp@f,800000/sd@3,0:a
```

Logical device names are found in the `/dev` directory and are symbolically linked to their corresponding physical device names in the `/devices` directory.

Controller Number

Controller (or interface) numbers such as `c0`, `c1`, and `c3` are automatically assigned in sequential order to each interface card.

If your system has a built-in SCSI interface, the operating system automatically assigns a `0` (zero) to that card. Therefore, any disk drive that is connected to the built-in SCSI card will have a device address that starts with `c0`.

Controller number `c1` would correspond to a second SCSI host adapter (or `esp@1`) which is explained under the Physical Disk Device Names Review section of this lesson.

Logical Disk Device Names

Target Address

Target addresses such as `t0`, `t1`, and `t3` correspond to the address switch setting that is selected for each device. An external disk drive has an address switch that is located on the rear panel.

An internal disk drive usually has a jumper setting that has been preset to `3`. A second internal disk drive is usually set to `1`.

Disk Number

The disk number is always set to `d0` for any embedded SCSI device or IPI disk.

Slice (Partition) Number

Slice numbers range from `0` to `7`. To specify an entire disk, use slice `2`.

Disk devices are accessed by their logical device names, and this name must include the slice number. Disks cannot be accessed by just their *controller/disk/target* designation.

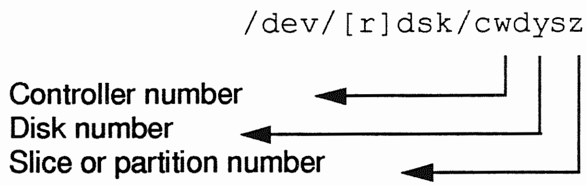
Note the contents of the `/dev/dsk` directory that contains the logical device names of the system's disk devices and partitions.

```
% ls /dev/dsk
/dev/dsk/c0t1d0s0     /dev/dsk/c0t2d0s3     /dev/dsk/c0t3d0s6
/dev/dsk/c0t1d0s1     /dev/dsk/c0t2d0s4     /dev/dsk/c0t3d0s7
/dev/dsk/c0t1d0s2     /dev/dsk/c0t2d0s5     /dev/dsk/c0t6d0s0
/dev/dsk/c0t1d0s3     /dev/dsk/c0t2d0s6     /dev/dsk/c0t6d0s1
/dev/dsk/c0t1d0s4     /dev/dsk/c0t2d0s7     /dev/dsk/c0t6d0s2
/dev/dsk/c0t1d0s5     /dev/dsk/c0t3d0s0     /dev/dsk/c0t6d0s3
/dev/dsk/c0t1d0s6     /dev/dsk/c0t3d0s1     /dev/dsk/c0t6d0s4
/dev/dsk/c0t1d0s7     /dev/dsk/c0t3d0s2     /dev/dsk/c0t6d0s5
/dev/dsk/c0t2d0s0     /dev/dsk/c0t3d0s3     /dev/dsk/c0t6d0s6
/dev/dsk/c0t2d0s1     /dev/dsk/c0t3d0s4     /dev/dsk/c0t6d0s7
/dev/dsk/c0t2d0s2     /dev/dsk/c0t3d0s5
%
```

Logical Disk Device Names

Direct Controllers

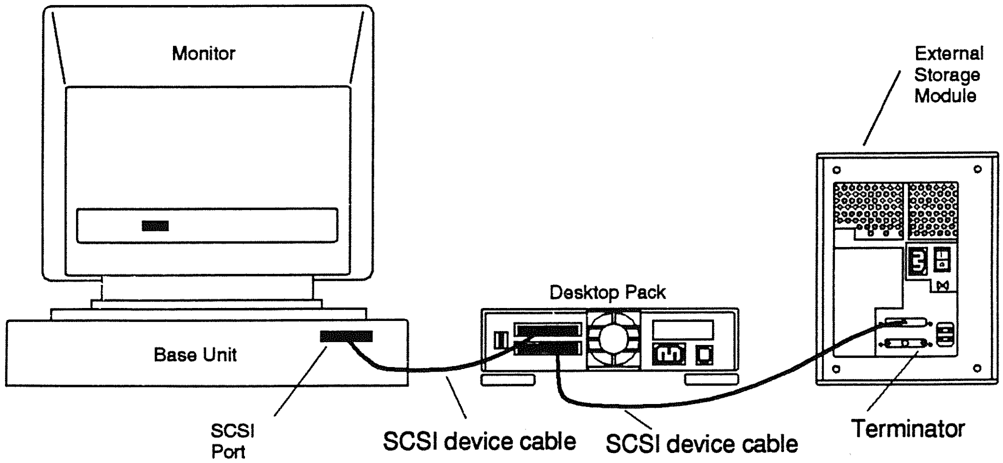
Disks connected to *direct* (or non-bus oriented) controllers such as the Xylogics 451 or 7053, do not have a target number in their logical device name.



Example:

```
/dev/dsk/c1d0s0 -> /devices/vme/xdc@6d,ee80/xd@0:0
```

Setting Disk Drive Addresses



Rear View

If you have more than one disk drive, you must set each SCSI target address switch to a different address.

Both the external address switch setting and the internal jumper setting provide unique target addresses for disk drives.

The logical controller numbers are automatically assigned by the operating system.

When you add a new disk device (or SCSI device) to your system, you should verify that the target address does not conflict with any existing target numbers.

The `probe-scsi`, `probe-scsi-all`, and `probe-ipi` commands can be used from the PROM monitor mode (`ok` prompt) to display the current target device numbers set for your disk devices.

You can have up to seven devices daisy-chained to each SCSI host adapter or eight devices connected to an IPI controller, so it is important that each SCSI device and IPI disk have a unique target address (or number).

Summary

In this lesson, you learned that:

- Sun supports different types of disk device interfaces.
  - Small computer system interface (SCSI)
  - Intelligent peripheral interface (IPI)
  - Xylogics (451 or 7053) controller
- Disk devices are referenced by their logical device names when using disk and file system-related commands.
- Disk devices connected to the same device interface (or controller) must be uniquely identified by a target address (or unit number).

Exercise 1-1

Part I

Write down your answers to these questions:

1. Identify four tasks that system administrators do that reference disk devices (and partitions) by their logical device names.

   ______
   ______
   ______
   ______

2. When you administer disk devices on your system, how do you reference the devices?

   ______

3. Name at least one command you can use to identify the disk interfaces on your system.

   ______

4. How are two disk devices connected to the same controller uniquely identified?

   ______
   ______

Part II

Use the `format` utility to write down your answers to these questions:

1. What type of disk device interface is connected to your system?

   SCSI
   ______

2. How many disk devices are connected to your system?

   1
   ______

3. What type of disk devices are connected to your system?

   Seagate
   ______

Adding a New Disk 2

Objectives

Upon completion of this lesson, you will be able to:

- Write the five major steps to adding a new disk to a system.
- Define the following terms: sector, track, cylinder, partition table, and disk label.
- Partition a disk.
- Display a disk's volume table of contents with the `format` and `prtvtoc` commands.

References

SunOS 5.1 Adding and Maintaining Devices and Drivers,
Chapter 1, "Disks"

Introduction

There are five major steps to adding a new disk to a system:

1. Setting up the hardware, including setting switches and attaching cables.
2. Configuring the system to recognize the new disk.
3. Partitioning the disk.
4. Making file systems on the partitions.
5. Adding information about the partitions to the file system table (the `/etc/vfstab` file).

This lesson covers the middle three steps. Follow the instructions that come with the disk to complete the first step. The last step is covered in Lesson 3, "Mounting File Systems."

Reconfiguring a System to Recognize a New Disk

Use the following steps to reconfigure a system to recognize a new disk:

1. Create the /reconfigure file so the system will perform a reconfiguration boot when it is rebooted.

   ```
   # touch /reconfigure
   ```

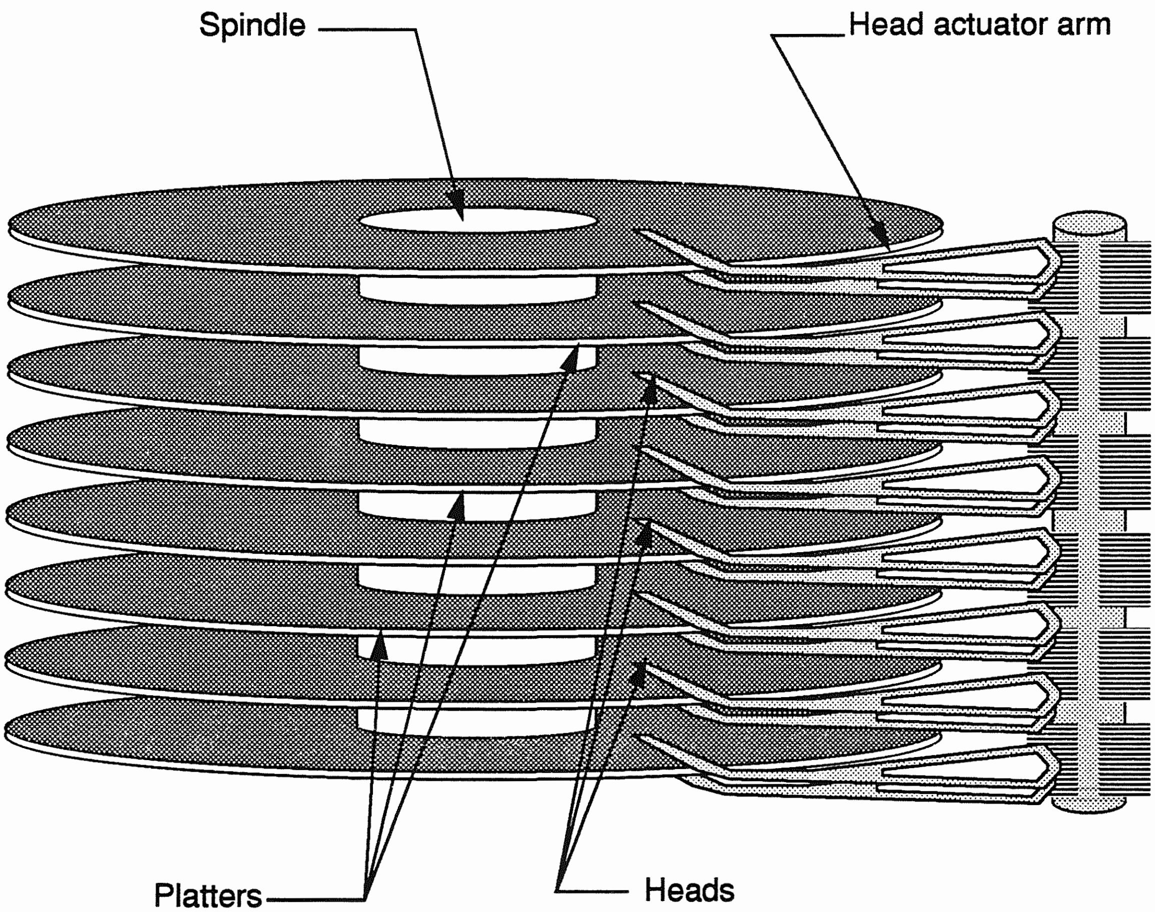
2. Halt the system.
3. Turn the power off.
4. Use the appropriate cable to connect the new disk to the system.
5. Turn the power back on.

The reconfiguration boot process creates the physical and logical device entries for the new disk.

After configuring the system to recognize a disk, you can begin the process of setting up the disk's partitions.

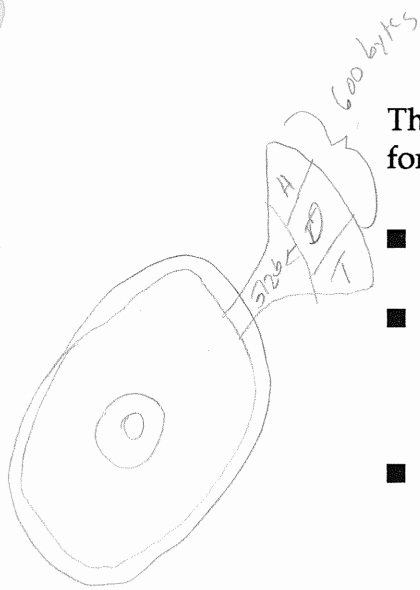
Describing Disk Architecture

The physical features of the disk are illustrated below.



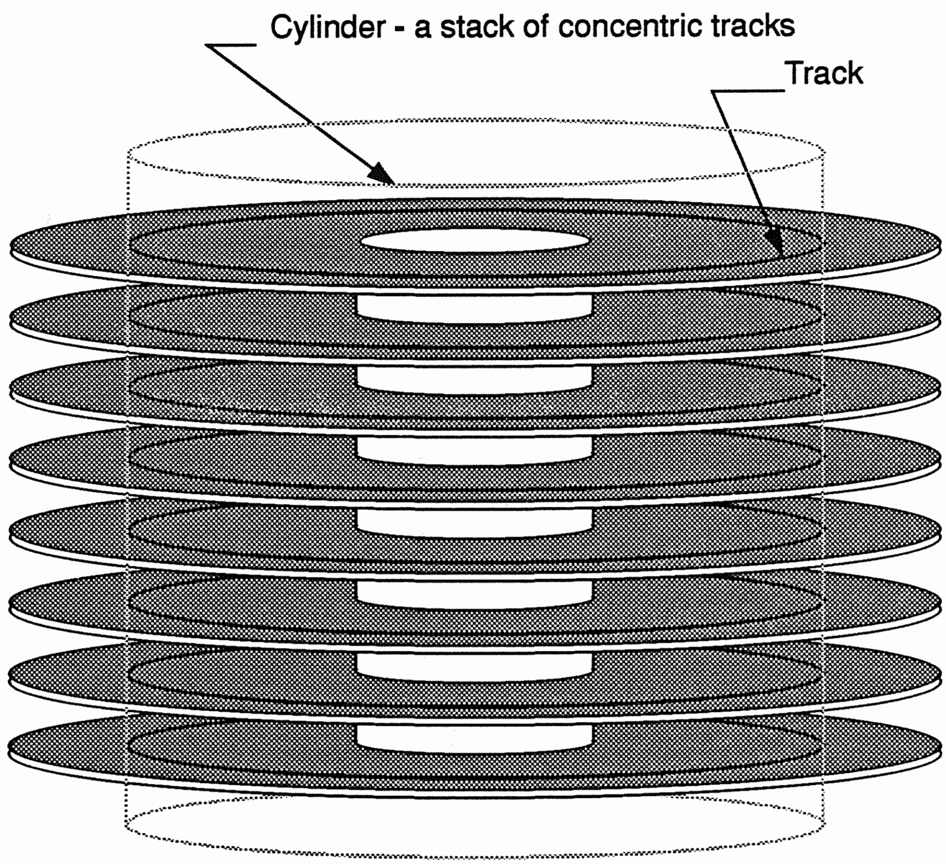
- Disks are composed of several *platters* that are read and written on both sides.
- The platters rotate around a *spindle*.
- The *read/write heads* are moved as a unit by the *head actuator arm*.

Describing Disk Architecture



The Solaris 2.*x* file system also takes advantage of the low-level formatting parameters to improve disk I/O performance.

- A file's blocks are stored in *sectors* of 512 bytes each.
- Sectors are sections of a *track*. The sectors making up a track can be read or written by a given head in one position during a single disk revolution.
- The sum of tracks provided by all the heads at a given position is known as a *cylinder*.



- Because a disk is constantly spinning and because read/write heads move as a unit, the most efficient seeking occurs when the blocks to be read or written are located in a single cylinder.
- Partitions begin at a cylinder boundary.

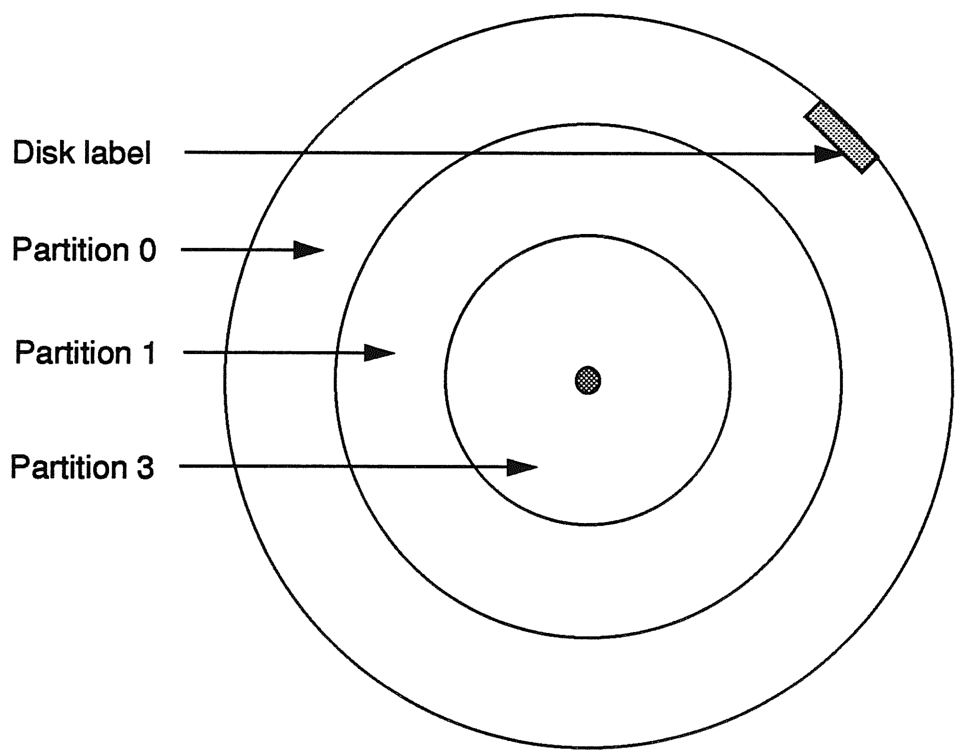
Describing Disk Partitions and Labels

Partitions are described by an offset (distance) from the outer edge of the disk and a size.

The offsets and sizes for a disk's partitions are defined by a *partition table*.

A disk's label, also called the disk's *Volume Table of Contents* (VTOC), contains:

- *Partition tables* for the disk
- An optional *volume name* that identifies the disk device
- Optional *partition tags* that name the standard mount points for each of the partitions, and
- Optional *partition flags* that label whether each partition is writable and/or mountable.



Using the `format` Utility

The SunInstall™ utility creates partitions on the disk(s) you choose to install software on. The `format` utility creates partitions on any disks you add to the system after installation or disks that need to be repartitioned after system installation. This includes adding an additional swap partition.

The `format` utility is a disk maintenance tool that is run from the shell or from an installation CD-ROM, if `format` is used to modify a disk that contains the / (`root`) or `/usr` file systems or a swap partition.

The basic tasks for repartitioning a disk are:

- Repartition the disk.
- Relabel the disk with the new disk label.
- Create the file system interface for the new partition. (This task can be skipped if adding an additional swap partition.)

Using the `format` Utility

1. Type `format` at the shell prompt and press Return.

   ```
   # format
   ```

   When the `format` utility is first started, each available disk is described by its logical name, its marketing name, some of its physical parameters, and its physical name.

   The marketing name usually begins with SUN*nnnn*. In the example below, the marketing name is SUN0424.

   You are prompted to choose a disk from the disks currently recognized by the system.

2. To choose a disk, enter the number corresponding to its description.

Example:

```
# format
Searching for disks...done
 AVAILABLE DISK SELECTIONS:
       0. c0t3d0 <SUN0424 cyl 1151 alt 2 hd 9 sec 80>
        /iommu@f,e0000000/sbus@f,e0001000/espdma@f,400000/esp@f,800000/sd@3,0
       1. c0t0d0 <SUN0424 cyl 1151 alt 2 hd 9 sec 80>
        /iommu@f,e0000000/sbus@f,e0001000/espdma@f,400000/esp@f,800000/sd@0,0
Specify disk (enter its number): 1
```

Caution: If you repartition an existing disk after SunInstall, data on those file systems will be lost. Back up your file systems before repartitioning existing disks.

Examining the `format` Main Menu

After you select a disk, the `format` utility displays its main menu.

```
selecting c0t0d0
[disk formatted]

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> partition
```

This lesson describes disk partitioning and labeling.

Displaying the `partition` Menu

The partition menu is used to display and modify partition tables.

1. To display this menu, type `partition` and press Return.

```
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 menu selections allow you to perform the following functions:

`0-7` Specify the offset and size of up to eight partitions.

`select` Select a predefined partition table (several predefined tables are supplied for each disk type supported by Sun and you can add other definitions).

`modify` Modify a predefined partition table.

`name` Name the current partition table.

`print` Display the current partition table.

`label` Write the current table to the disk label.

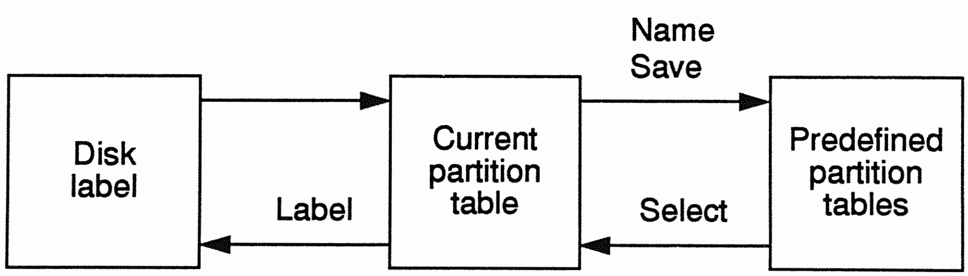
Exploring Where Partition Tables Are Stored

Partition tables exist in several different forms:

- All formatted disks have a partition table as part of their disk label.
- A set of predefined partition tables are stored in a file named `/etc/format.dat` that can be accessed by `format`.
- When you select a disk within `format`, the partition table stored in the disk's label becomes the current label. The current label is the label that is currently in memory. You can also select a predefined label to be the current label.

  The current label can be modified and:

  - Written to the disk label using the `label` command under the partition menu, or
  - Named using `name`, under the partition menu, and then added to the list of predefined tables that come with the release, using the `save` command under the `format` utility's main menu.



Different New Disk Scenarios

New disks require different treatment depending on the type of disk and how the disk was prepared by the distributor.

- New disks purchased from Sun are preformatted, have a disk label, and have several predefined partition tables stored in the `/etc/format.dat` file.

- New disks that are supported by Sun and purchased from other vendors also have several predefined partition tables stored in the `/etc/format.dat` file, and may or may not be preformatted and have a disk label.

- New disks that are not supported by Sun do not have any predefined partition tables stored in the `/etc/format.dat` file. You will be required to supply disk geometry information. These disks also may or may not be preformatted and have a disk label.

The steps described in this lesson assume that your new disk was purchased from Sun.

Displaying Partition Information

The `print` selection allows you to view the current partition table.

1. To display the current partition table, type `print` and press the Return key.

```
partition> print
Current partition table (SUN0424):
Part      Tag          Flag     Cylinders     Size        Blocks
  0       root          wm      0    - 45     16.17MB     (46/0/0)
  1       swap          wu      46   - 136    31.99MB     (91/0/0)
  2       backup        wu      0    - 1150   404.65MB    (1151/0/0)
  3       unassigned    wm      0             0           (0/0/0)
  4       unassigned    wm      0             0           (0/0/0)
  5       unassigned    wm      0             0           (0/0/0)
  6       usr           wm      137 - 1150    356.48MB    (1014/0/0)
  7       unassigned    wm      0             0           (0/0/0)
partition>
```

- If the disk had a volume name it would be listed after the word `Volume` in the first line of the table.
- The name of the partition table is displayed in parentheses in the second line of the table.
- The columns of the table have the following meanings:

`Part`	The partition number
`Tag`	The partition tag
`Flag`	The partition flags
`Cylinders`	The range of cylinders occupied by the partition.
`Size`	The size of the partition in Megabytes
`Blocks`	The size of the partition in *cylinders/tracks/sectors* notation

Examining a Working Partition Table

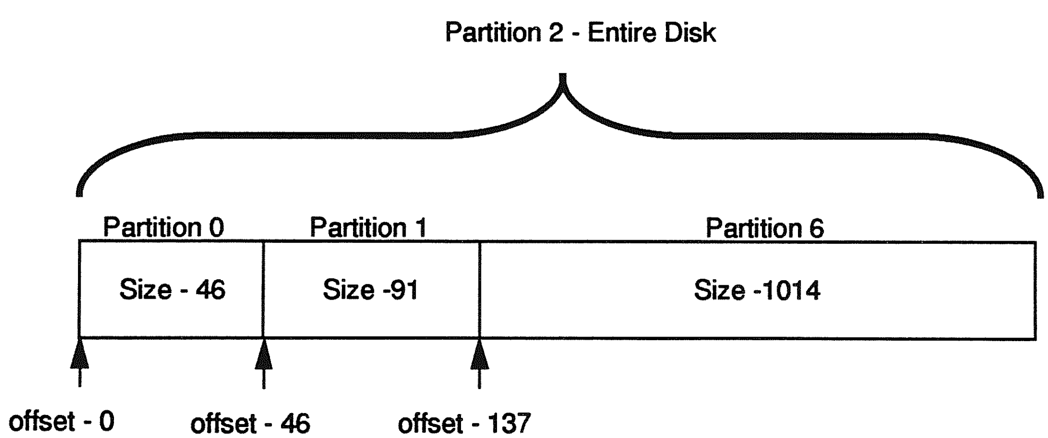
Partitions should be contiguous. The first partition should start at cylinder 0. The second partition should start immediately after the first partition, and so on.

The `Cylinders` column of the following table indicates that the partitions described by the table are contiguous.

Part	Tag	Flag	Cylinders	Size	Blocks
0	root	wm	0 - 45	16.17MB	(46/0/0)
1	swap	wu	46 - 136	31.99MB	(91/0/0)
2	backup	wu	0 - 1150	404.65MB	(1151/0/0)
3	unassigned	wm	0	0	(0/0/0)
4	unassigned	wm	0	0	(0/0/0)
5	unassigned	wm	0	0	(0/0/0)
6	usr	wm	137 - 1150	356.48MB	(1014/0/0)
7	unassigned	wm	0	0	(0/0/0)

Partition 2 is a special case. This partition typically describes the whole disk.

The following figure is a graphic representation of the above table.



Changing a Partition's Size

The following steps show how to modify the size of a partition.

1. Increase the size of partition 0 to 20 megabytes.

   Type `modify` and press Return to change the existing partition table.

```
partition> modify
Select partitioning base:
        0. Current partition table (SUN0424)
        1. All Free Hog
Choose base (enter number) [0]? Return
```

   Press Return or type `0` to change the current partition table.

2. The current partition table is displayed.

```
Part     Tag          Flag    Cylinders     Size       Blocks
  0      root         wm      0    - 45     16.17MB    (46/0/0)
  1      swap         wu      46   - 136    31.99MB    (91/0/0)
  2      backup       wu      0    - 1150   404.65MB   (1151/0/0)
  3      unassigned   wm      0             0          (0/0/0)
  4      unassigned   wm      0             0          (0/0/0)
  5      unassigned   wm      0             0          (0/0/0)
  6      usr          wm      137 - 1150    356.48MB   (1014/0/0)
  7      unassigned   wm      0             0          (0/0/0)>
```

3. Press Return or type `yes` to confirm the creation of a new partition table.

```
Do you wish to continue creating a new partition
table based on above table[yes]? Return
```

Changing a Partition's Size

4. Press Return to accept partition 6 as the Free Hog partition.

```
Free Hog partition[6]? Return
```

The Free Hog partition is used as disk space accumulator that expands and contracts as other partition sizes are changed. (This functionality is similar to SunInstall's Unallocated Space Meter.)

5. Type the size of partition 0 as `20mb` and press Return. Let the other partition sizes default to their current sizes.

```
Enter size of partition '0' [33120b, 46c, 16.17mb]:  20mb
Enter size of partition '1' [65520b, 91c, 31.99mb]:  Return
Enter size of partition '3' [0b, 0c, 0.00mb]:  Return
Enter size of partition '4' [0b, 0c, 0.00mb]:  Return
Enter size of partition '5' [0b, 0c, 0.00mb]:  Return
Enter size of partition '7' [0b, 0c, 0.00mb]:  Return
```

You are not prompted to change the size of partition 6 because it has been designated as the Free Hog. It is decreased in size because partition 0 increased to 20 megabytes.

6. The new partition table is displayed automatically. Verify the new partition sizes are correct.

```
Part      Tag    Flag     Cylinders        Size        Blocks
  0      root     wm       0 -   56       20.04MB    (57/0/0)
  1      swap     wu      57 -  147       31.99MB    (91/0/0)
  2    backup     wu       0 - 1150      404.65MB    (1151/0/0)
  3 unassigned    wm       0                0        (0/0/0)
  4 unassigned    wm       0                0        (0/0/0)
  5 unassigned    wm       0                0        (0/0/0)
  6       usr     wm     148 - 1150      352.62MB    (1003/0/0)
  7 unassigned    wm       0                0        (0/0/0)
```

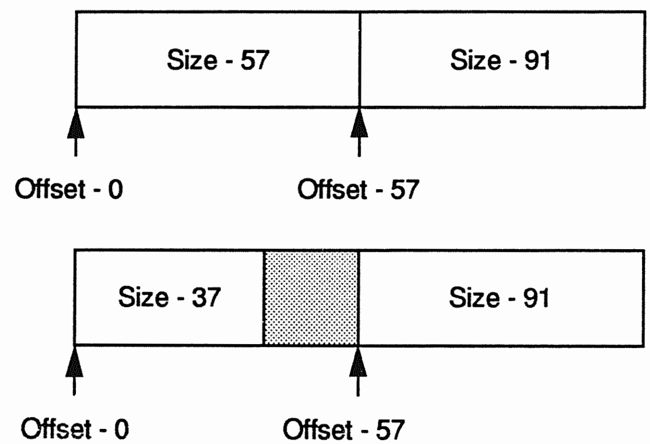
The `format` utility rounds the Mbytes to the next whole cylinder. Using the `modify` option (and designating a Free Hog partition) automatically adjusts the starting cylinder boundaries of the other partitions so there are no "holes" between partition boundaries.

Changing a Partition's Size

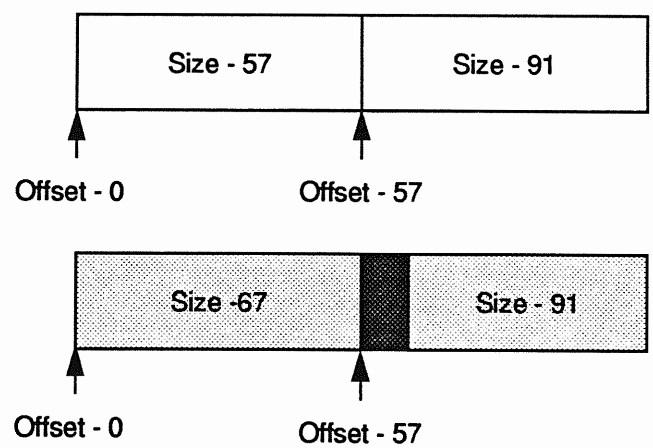
Exploring Partition Boundaries

This diagram illustrates what happens if you change two partition sizes using the partition menu's `0-7` options without adjusting the partitions' starting cylinder numbers (or *offset*).

Decreasing the size of a partition leaves wasted space between it and the next partition.



Increasing the size of a partition creates overlapping partitions.



Changing a Partition's Size

Exploring Partition Boundaries (continued)

When using the 0–7 options to change partition sizes, make sure all starting and ending cylinders are consecutive.

Part	Tag	Flag	Cylinders	Size	Blocks
0	root	wm	0 - **56**	20.04MB	(57/0/0)
1	swap	wu	**57** - **147**	31.99MB	(91/0/0)
2	backup	wu	0 - 1150	404.65MB	(1151/0/0)
3	unassigned	wm	0	0	(0/0/0)
4	unassigned	wm	0	0	(0/0/0)
5	unassigned	wm	0	0	(0/0/0)
6	usr	wm	**148** - 1150	352.62MB	(1003/0/0)
7	unassigned	wm	0	0	(0/0/0)

Changing a Partition's Size

Writing the Modified Table to the Disk Label

Once the partition sizes are changed, the modified disk label is written to disk.

7. Press Return to confirm using this partition table.

```
Okay to make this the current partition table[yes]?  Return
```

8. Name the new partition table name and press Return.

```
Enter table name (remember quotes): "c0t0d0.424"
```

Partition table names usually contain some reference to the disk name and/or size.

9. Write the modified table to the disk by typing `label` and pressing Return.

```
Ready to label disk, continue?  yes
```

10. Type `quit` (or `q`) and press Return to exit the partition menu.

```
partition> quit
```

The main `format` menu is displayed.

Changing a Partition's Size

Verifying the New Disk Label

11. Verify the new disk label with the `verify` command from the `format` main menu.

```
format> verify
Primary label contents:
ascii name  = <SUN0424 cyl 1151 alt 2 hd 9 sec 80>
pcyl        = 2500
ncyl        = 1151
acyl        =    2
nhead       =    9
nsect       =   80
Part      Tag    Flag     Cylinders        Size       Blocks
  0      root     wm       0 -   56      20.04MB    (57/0/0)
  1      swap     wu      57 -  147      31.99MB    (91/0/0)
  2    backup     wu       0 - 1150     404.65MB    (1151/0/0)
  3 unassigned    wm       0                0       (0/0/0)
  4 unassigned    wm       0                0       (0/0/0)
  5 unassigned    wm       0                0       (0/0/0)
  6       usr     wm     148 - 1150     352.62MB    (1003/0/0)
  7 unassigned    wm       0                0       (0/0/0)
format> quit
```

Changing a Partition's Size

Verifying the New Disk Label

The `prtvtoc` (print vtoc) command is also used to display a disk's volume table of contents.

```
# prtvtoc /dev/rdsk/c0t0d0s0
* /dev/rdsk/c0t0d0s0 partition map
* Dimensions:
*     512 bytes/sector
*      80 sectors/track
*       9 tracks/cylinder
*     720 sectors/cylinder
*    2500 cylinders
*    1151 accessible cylinders
* Flags:
*   1: unmountable
*  10: read-only
*                          First     Sector    Last
* Partition  Tag  Flags    Sector     Count    Sector  Mount Directory
       0      2    00          0     41040     41039
       1      3    01      41040     65520    106559
       2      5    01          0    828720    828719
       6      4    00     106560    722160    828719
#
```

Changing a Partition's Size

Creating a File System

After using the `format` utility to change a partition's size, the next step is to create a file system in order to add new data. Data stored on a partition is actually accessed through the file system interface.

The `newfs` command is a "friendly" front end to the `mkfs` command which actually creates the file system.

1. To create a file system on the first partition of the newly partitioned disk, enter the following command.

```
# newfs /dev/rdsk/c0t0d0s0
newfs: construct a new file system /dev/rdsk/c0t0d0s0:
(y/n)? y
/dev/rdsk/c0t0d0s0:     41040 sectors in 57 cylinders of 9
tracks, 80 sectors
        21.0MB in 4 cyl groups (16 c/g, 5.90MB/g, 2688 i/g)
super-block backups (for fsck -F ufs -o b=#) at:
 32, 11632, 23232, 34832,
#
```

You are asked if you want to proceed. After making sure that you named the correct partition, answer yes.

Information about the file system being created is displayed on the screen.

Caution - This command erases any data on an existing file system. Make sure this is what you want to do before answering yes.

This command creates a default file system including a `root` inode, a `root` directory, and a `lost+found` directory which is used by the file system check and repair (`fsck`) utility.

2. Repeat the previous step for each partition that will be used to contain a file system.

Summary

In this lesson, you learned that:

- A reconfiguration boot is performed to recognize a new device.
- A disk is partitioned using the `format` utility.
- A disk's VTOC is displayed using the `format` utility or the `prtvtoc` command.
- A file system is created using the `newfs` command.

This lesson included procedures for repartitioning a new disk, which can be used to create an additional swap area and for creating a new file system interface for the new partition.

The following lesson describes how to modify the appropriate file so that these partitions are recognized during system startup.

Exercise 2-1

In this exercise you will use the `format` utility to create and save a working partition table on an unused disk that does not contain any part of the operating system. (*Follow Option 2.*)

Some lab environments may not have spare disks to modify partition tables. (*If that is the case, follow Option 1.*)

Option 1

Use the `format` utility to identify disk information on your system.

1. Become superuser and invoke the `format` utility.

2. Identify the marketing names of the disk(s) on your system.

   Seagate

   Select the first disk from the available disk selections and press Return.

3. The `format` utility's main menu is displayed. Use the `verify` option to display the disk label information.

   Identify the following information including the logical device name:

   Logical Disk Name:________________

Partition (Slice)	Size in Mbytes
0	23.20
1	64.34
3	33.05
4	30.23
5	64.69
6	181.41
7	7.73

Exercise 2-1

4. If you have more than one disk, use the `disk` option from the main menu to select the next disk.

5. Use the `verify` option to identify the following information about the second disk.

Logical Disk Name:_________________

Partition (Slice)	Size in Mbytes
0	
1	
3	
4	
5	
6	
7	

6. Exit the `format` utility.

Do not proceed to option 2.

Option 2

Use the `format` utility to increase a partition size on the unused disk.

1. Become superuser and invoke the `format` utility.
2. Select the unused disk from the available disk selections.
3. Display the partition menu by using the `partition` option.
4. Display the current partition table by using the `print` option.
5. Increase the size of one partition by 10 Mbytes by make another partition smaller by 10 Mbytes.

Exercise2-1

6. Use the `modify` option to increase the partition size as described in this lesson. Set the Free Hog partition to the partition to be made smaller.

7. Use the `print` option to display the partition changes.

8. Name the partition table using a reference to its size such as `d669`.

9. When the partition sizes are changed, use the `label` option to create the new partition table.

10. Exit the `partition` menu.

11. Use the `verify` option from the main menu to display the new partition table. Verify that there are no overlapping partitions or holes in your new partition table.

12. Exit the `format` utility.

13. Use the `newfs` command to create a new file system on the newly expanded partition.

Mounting File Systems 3

Objectives

Upon completion of this lesson, you will be able to:

- Mount and unmount local file systems.
- Mount a file system of a specified file system type.
- Set up your system to automatically mount a local file system at boot time.

References

SunOS 5.1 Routine System Administration Guide,
Chapter 2, "Understanding and Planning File Systems"
Chapter 4, "Mounting and Unmounting File Systems"

SunOS 5.1 How-To Book: Basic System Administration Tasks,
"File Systems"

Introduction

This lesson presents the concepts and procedures involved in mounting and unmounting local file systems and modifying the `/etc/vfstab` file to automatically mount file systems at boot time.

Local and Distributed File Systems

The two file systems that most administrators will be working with are the disk-based (or local) file system type and the remote (or distributed) file system type.

This lesson only discusses the disk-based (or local) file system type.

Disk-Based (Local) File Systems

These file systems are stored on physical media such as disk, CD-ROM, or diskette.

`ufs`	The default disk-based file system is based on the BSD Fat Fast File system.
`hsfs`	The High Sierra and CD-ROM file system.
`pcfs`	A file system that supports read/write access to data on disk operating system (DOS) diskettes.

Distributed File Systems

The distributed file system supports access to file systems attached to other systems on a network.

`nfs`	Network file system
`rfs`	Remote file system

Mounting and Unmounting File Systems

Mounting is the process by which separate file systems become integrated into a single directory hierarchy. Typically, mounting and unmounting occur automatically during system startup and shutdown.

It is not, however, uncommon for system administrators to encounter situations that require them to mount and unmount file systems by hand. These situations include making backups, checking the file systems for inconsistencies, and re-partitioning.

In addition, disk reconfiguration sometimes requires system administrators to make changes to `/etc/vfstab`, which specifies how file systems are automatically mounted during system startup and shutdown.

Test Mounting a New File System

A scenario where you might mount a file system by hand is to test out a newly created file system, such as the one created at the end of the last lesson.

```
# mkdir /database
# mount /dev/dsk/c0t0d0s0 /database
```

Unmounting a file system makes the files on the system inaccessible to users. To unmount a file system, use one of the following examples.

- The device special file system name:

  ```
  # umount /dev/dsk/c0t0d0s0
  ```

- The mount-point for the file system as specified in the `/etc/vfstab` file:

  ```
  # umount /database
  ```

While it is possible to mount and unmount file systems by hand, as illustrated above, the mounting and unmounting of file systems usually occurs automatically during system startup and shutdown.

Mounting All Local File Systems

During start-up and shutdown all local file systems are mounted and unmounted with a single command.

Mounting All Local File Systems

Local file systems are mounted by the `MOUNTFSYS` script when the system enters run level 2. The command in this script used to mount all local file systems is:

```
mountall -l
```

The `-l` option indicates local file systems.

How Does `mountall` "Know" What to Mount Where?

The `mountall` command is something like a step in an Assembly Instruction book for an "assembly required" item that says "attach all screws." Somewhere there must be a listing of all of the screws and where they should be attached. For the `mountall` command, this information is provided by the `mount at boot` field of the `/etc/vfstab` file.

Note: File systems are no longer exported, but /etc/dfs/dfstab with the "shareall" command is used.

The /etc/vfstab File

This file provide default settings for mounting file systems.

The format of the file is one record per line, seven fields per record, with a dash (–) indicating a null value for a field.

```
#device                 device               mount          FS      fsck  mount    mount
#to mount               to fsck              point          type    pass  at boot  options
#
#/dev/dsk/c1d0s2        /dev/rdsk/c1d0s2     /usr           ufs     1     yes      -
/proc                   -                    /proc          proc    -     no       -
fd                      -                    /dev/fd        fd      -     no       -
swap                    -                    /tmp           tmpfs   -     yes      -
/dev/dsk/c0t3d0s0       /dev/rdsk/c0t3d0s0   /              ufs     1     no       -
/dev/dsk/c0t3d0s6       /dev/rdsk/c0t3d0s6   /usr           ufs     2     no       -
/dev/dsk/c0t3d0s3       /dev/rdsk/c0t3d0s3   /export        ufs     5     yes      -
/dev/dsk/c0t3d0s7       /dev/rdsk/c0t3d0s7   /export/homeufs        6     yes      -
kathmandu:/export/packages - /export/packages               nfs     -     yes      -
/dev/dsk/c0t3d0s4       /dev/rdsk/c0t3d0s4   /export/swapufs        8     yes      -
/dev/dsk/c0t3d0s5       /dev/rdsk/c0t3d0s5   /opt           ufs     9     yes      -
/dev/dsk/c0t3d0s1       -                    -              swap    -     no       -
```

The fields are:

`device to mount`
: Identifies the logical (block) device name of a local `ufs` file system

`device to fsck`
: Identifies the logical (raw) device name of a local `ufs` file system

`mount point`
: The default mount point for the local file resource.

`FS type`
: Always `ufs` for local file resources.

The /etc/vfstab File

fsck pass

The ufs file system is checked if this field contains a value greater than zero; a dash (–) means no check. If the value is 1, each ufs file system is checked sequentially. If the value is greater than 1, then fsck checks multiple ufs file systems in parallel on different disks for maximum efficiency.

mount at boot

Either yes or no indicating whether the file resource should be mounted when the system enters run level 2, or when the mountall command is issued.

Two exceptions to this functionality is the / (root) and /usr file systems. Both of these are mounted by the mount commands specified in the /etc/rcS.d/S30rootusr.sh script. This is why their mount-at-boot fields are set to no.

mount options

A comma-separated list of mount options.

The Fields that Specify What Gets Mounted Where

The mountall command uses entries in the file to determine if file systems should be mounted at boot time.

Specifically, the mount-at-boot field determines what file systems are mounted a boot time.

Solaris 2.*x* supports several file system types. The fourth field of the /etc/vfstab file, FS type, specifies the file system type. Focus your attention on fields 1 and 3 of the records for ufs (local) file systems.

```
#device             mount
#to mount           point
/dev/dsk/c0t0d0s0   /
/dev/dsk/c0t0d0s6   /usr
/dev/dsk/c0t3d0s5   /opt
```

These fields list the local file systems to be mounted and their mount points.

Displaying Information About Mounted File Systems

Since mounting usually occurs automatically, it is often useful to be able to determine what file systems are mounted at any given time.

To identify the file systems that have been mounted for you along with the options they were mounted with, type the `mount` command without any arguments.

```
# mount
/ on /dev/dsk/c0t3d0s0 read/write/setuid on Wed Jun 2 18:20:03 1993
/usr on /dev/dsk/c0t3d0s6 read/write/setuid on Wed Jun 2 18:20:03 1993
/proc on /proc read/write/setuid on Wed Jun 2 18:20:03 1993
/dev/fd on fd read/write/setuid on Wed Jun 2 18:20:03 1993
/tmp on swap read/write on Wed Jun 2 18:20:20 1993
/export on /dev/dsk/c0t3d0s3 setuid/read/write on Wed Jun 2 18:20:22 1993
/export/home on /dev/dsk/c0t3d0s7 setuid/read/write on Wed Jun 2 18:20:23 1993
/export/swap on /dev/dsk/c0t3d0s4 setuid/read/write on Wed Jun 2 18:20:24 1993
/opt on /dev/dsk/c0t3d0s5 setuid/read/write on Wed Jun 2 18:20:25 1993
/export/packages on kathmandu:/export/packages read/write/remote on Thu Jun 3
08:29:08 1993
```

The fields are:

Field 1	Names the currently used mount point
Field 2	The word "on"
Field 3	Names the partition containing that file system, or special file system type
Field 4	Specifies whether the file system is mounted read/write or read only
Field 5	The word "on"
Field 6	Specifies the date and time the file system was mounted

Mounting a Specific File System Type

Different file system types have different formats that affect the mechanics of how it is mounted. Therefore, the file system type must be specified when you mount it.

The `mount` command has a `-F` option that is used to specify the type of file system being mounted.

Examples:

1. The following command mounts a diskette as a `pcfs` file system.

   ```
   # mkdir /pcfs
   # mount -F pcfs /dev/diskette /pcfs
   ```

2. If the file system type must be specified when you mount it, how does the following command work without a `-F` option?

   ```
   # mount /dev/dsk/c0t0d0s0 /database
   ```

How File System Type Is Determined

File system type is determined in the following sequence:

1. From the `-F` option, if supplied.
2. By matching the block device (if found) with a file system type in the `/etc/vfstab` file and using that type.
3. By using the defaults specified in `/etc/default/fs` for local file systems and in `/etc/dfs/fstypes` for distributed file systems.

Assuming that `/dev/dsk/c0t0d0s0` was a newly created file system not listed in the `/etc/vfstab` file, the above command would work because `ufs` is the default specified in the `/etc/default/fs` file.

Summary

In this lesson, you learned how to:

- Mount and unmount file systems.
- Mount and unmount specific file system types.
- Display information about all currently mounted file systems.
- Add an entry for swap and file system partitions to the `/etc/vfstab` file.

Exercise 3-1 (Optional)

The purpose of this exercise is to add entries to the `/etc/vfstab` file.

The previous lesson described how to create partitions and file systems. To make these partitions usable, you must tell the system about them by creating entries in the `/etc/vfstab` file.

(If you were able to create a new partition and file system in the previous exercise, follow the steps listed below.)

Use these steps to modify the `/etc/vfstab` file.

1. Open the `/etc/vfstab` file for editing.
2. For each file system add an entry of the following form:

```
/dev/dsk/c0t0d0s0   /dev/rdsk/c0t0d0s0  /database     ufs   7    yes   -
```

3. For any additional swap partition, add an entry of the following form:

```
/dev/dsk/c0t2d0s1   -                   -             swap  -    no    -
```

Maintaining File Systems

4

Objectives

Upon completion of this lesson, you will be able to:

- List disk space used by directories and files.
- Summarize disk usage by file system.
- Identify files that match a specified criterion.
- List disk usage by user name.
- Define a cylinder group, cylinder group block, superblock, file system block, and file system fragment.
- Describe why `fsck` is necessary.
- Describe how to check and repair a file system.

References

SunOS 5.1 Routine System Administration Guide,
Chapter 14, "Checking the Integrity of File Systems"

M. Bach, *The Design of the UNIX Operating System*, Prentice-Hall, 1986

S. Leffler, M. McKusick, M. Karels, J. Quarterman, *The Design and Implementation of the 4.3BSD UNIX Operating System*, Addison-Wesley, 1989

Introduction

The first part of this lesson introduces commands to monitor disk usage.

No matter how much disk space there is available, users can find a way to use it all. One of the day-to-day tasks for most system administrators is monitoring disk usage and attempting to keep usage under control.

The second part of this lesson provides the concepts and procedures required to maintain the integrity of file systems.

Monitoring Disk Usage

The du Command

The du command displays the number of disk (512-byte) blocks used by directories and files. Without options or arguments, the du command displays the number of blocks used by each directory listed in the current directory and a grand total. If an optional directory path name is supplied, the command lists the number of blocks used by each subdirectory listed under that pathname and a grand total.

```
# du -k /usr
8       /usr/lost+found
143     /usr/kvm/lib/adb
144     /usr/kvm/lib
512     /usr/kvm
6940    /usr/bin
363     /usr/kernel/drv
```

Command format:

du [-a] [-s] [-k] [*directory*]

Options:

- -a Display the number of blocks used by all files and directories within the specified directory hierarchy.
- -s Display only the summary.
- -k Display in kilobytes (Kbytes).

Examples:

1. Display a summary of the /usr directory.

```
# du -s /usr
315742 /usr
```

Monitoring Disk Usage

Examples (continued):

2. Display all the files and subdirectories in the `/usr` directory.

```
# du -a /usr
16      /usr/lost+found
182     /usr/kvm/adb
2       /usr/kvm/arch
352     /usr/kvm/crash
...
752     /usr/kvm
...
315742 /usr
```

3. The following command displays the size of all the files and directories in the `/export/home` directory in reverse numeric order.

```
# du -a /export/home | sort -nr
```

Monitoring Disk Usage

The df Command

The df command displays information for each mounted file system.

Command format:

df [-k] [*directory*]

Options:

k Displays usage in Kbytes and subtracts the space reserved by the operating system from the amount of available space.

Example:

```
# df -k
Filesystem          kbytes   used   avail  capacity Mounted on
/dev/dsk/c0t3d0s0    22343  14876    5237     74%   /
/dev/dsk/c0t3d0s6   192151 155682   17259     90%   /usr
/proc                    0      0       0      0%   /proc
fd                       0      0       0      0%   /dev/fd
swap                 51700     12   51688      0%   /tmp
/dev/dsk/c0t3d0s3    29911   6340   20581     24%   /export
/dev/dsk/c0t3d0s7    21615      9   19446      0%   /export/home
/dev/dsk/c0t3d0s4    28831      9   25942      0%   /export/swap
/dev/dsk/c0t3d0s5    61999  45891    9918     82%   /opt
```

Root has a non-reported 10% reserved

The fields are:

Filesystem
The mounted file system.

kbytes Total size of file system's usable space.

avail Amount of available space

capacity Amount of space used as a percentage of the total capacity.

Mounted on
The mount point.

Identifying Who Is Using How Much Disk Space

The `quot` command displays how much disk space (in Kbytes) is used by users.

Command format:

quot [-af] [*filesystem* ...]

Options:

a	Report on all mounted file systems.

f	Show the number of files as well as the number of Kbytes owned by the user.

Example:

```
# quot -af
/dev/rdsk/c0t3d0s0 (/):
    11173   1856 root
     4140   18   lister
     2868   47   bin
      252   64   lp
       53   10   adm
       47   29   uucp
        9   5    sys
        1   1    daemon
/dev/rdsk/c0t3d0s6 (/usr):
   108927   8425 root
    49336   7203 bin
     2273   225  lp
      178   16   uucp
        1   1    adm
        1   1    sys
```

The first column identifies the number of Kbytes owned by the user.

The second column identifies the number of the files owned by the user.

quotas - disk quotas can be used on ufs filesystems

Finding Files

The `find` Command

The `find` command is used to perform actions on files within a directory hierarchy that match a specified criterion.

Examples:

The following examples show different ways of using the `find` command.

1. This command line displays the path names of all files in the `/export/home` hierarchy with a size greater than 10 disk blocks.

```
# find /export/home -size +10 -print
/export/home/hollie/dissertation
/export/home/rimmer/picture index
```

2. This command line displays the path names of all of the files in the `/export/home` hierarchy that are owned by the user named `rimmer`.

```
# find /export/home -user rimmer -print
/export/home/rimmer/file1
/export/home/rimmer/file2
...
/export/home/lister/letter
```

3. This command line displays the path names of all of the files in the `/export/home` hierarchy that are owned by the user named `rimmer` and are larger than 10 disk blocks.

```
# find /export/home -user rimmer -size +10 -print
/export/home/rimmer/picture_index
```

Checking File Systems

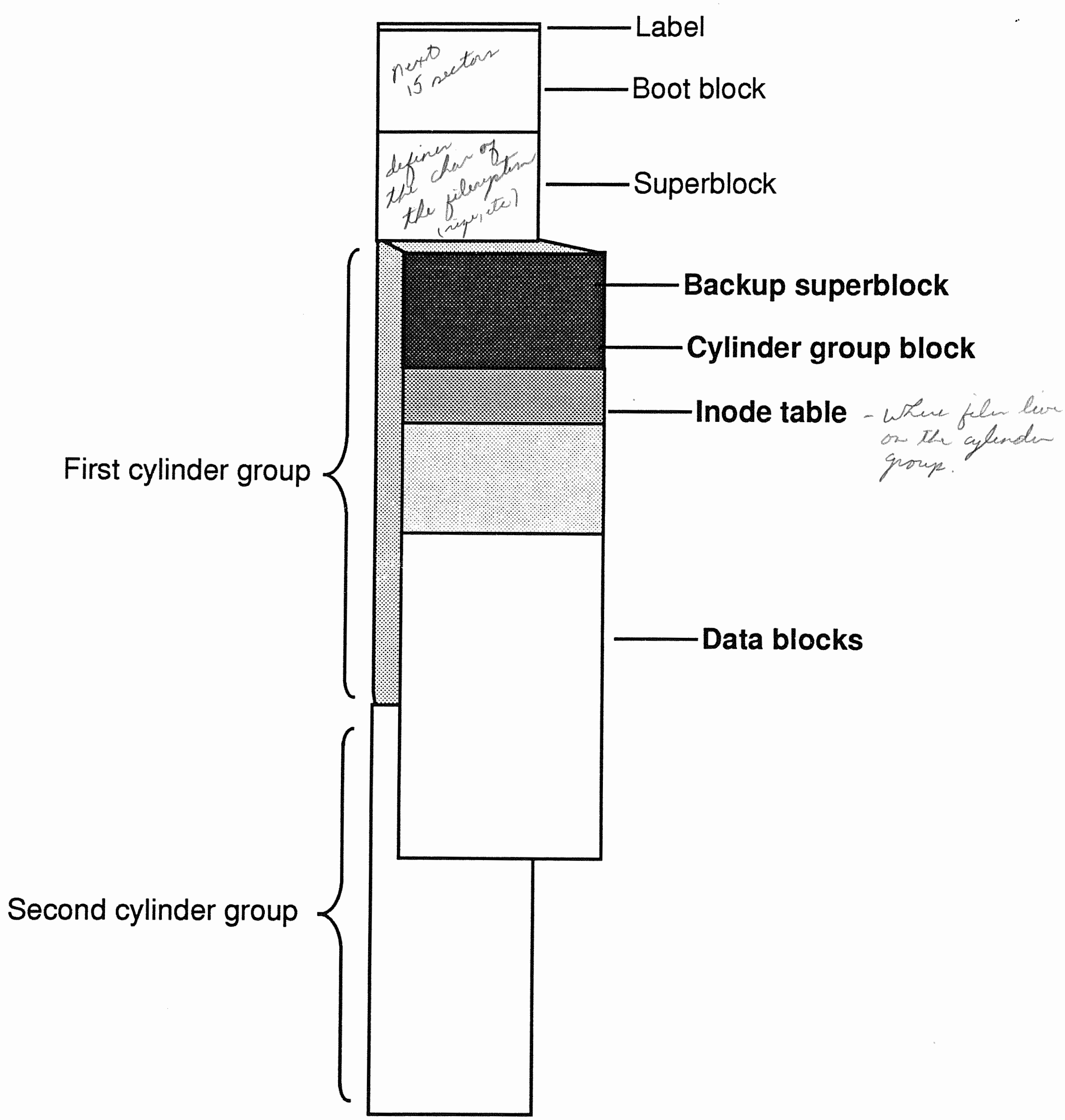
When the Solaris 2.*x* environment is booted, a consistency check of the file systems is automatically performed by the `fsck` program, which is used to check and repair file system inconsistencies.

Normally the program is able to make repairs without any action by the system administrator. Occasionally, inconsistencies are discovered that require action on the part of the system administrator.

The next section briefly introduces file system structures and concepts that are helpful in understanding how the `fsck` program works.

The File System Revisited

The SunInstall utility and the `newfs` command create the following file system structures.



Superblocks and Cylinder Group Blocks

The following section describes important file system structures.

Inode

An *inode* is the internal representation of a file that contains information such as the owner's UID and GID, number of bytes, and pointers to the file's data blocks.

Cylinder Groups

Another structure is the *cylinder group*. Old UNIX® file systems grouped all of their inodes together at the beginning of the file system, followed by all of the file system's data blocks. To improve performance, the new UNIX file system (the Berkeley Fat Fast File System) groups subsets of inodes and data blocks together into groups of consecutive cylinders called *cylinder groups*. The file system tries to keep each file's inode and all of its blocks in the same cylinder group.

Cylinder Group Blocks

The *cylinder group block* describes the number of inodes and data blocks, directories, free blocks and inodes, the free block list, and the used inode map in this cylinder group.

Superblocks

The *superblock* contains information about the entire file system such as the number of blocks and cylinder groups, the file system block and *fragment* size, a description of the hardware (derived from the label), and the mount point name. Because the superblock contains critical data, it is replicated in each cylinder group to protect against catastrophic loss. This is done when the file system is created. The copies are referenced if a disk failure causes the superblock to be corrupted.

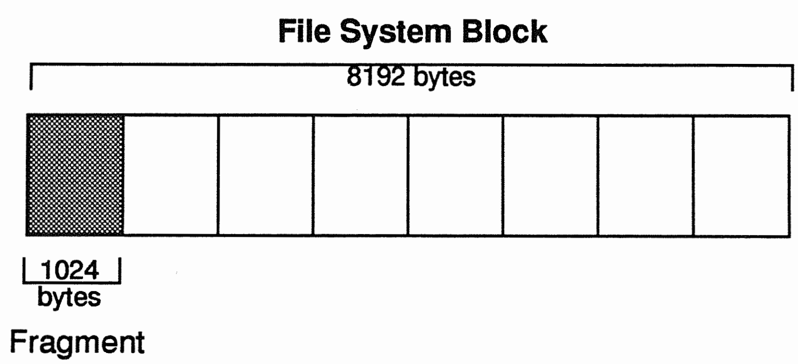
Block Fragments

The original UNIX file system used 512-byte file system blocks. This was expanded to 1024 bytes for System V Release 1. The advantage of a larger block size is that disk transfers are much more rapid when transferring large amounts of data.

The disadvantage of a large file system block size is that small files waste disk space. Each file that does not fill a large block to capacity wastes all of the empty portion of the block, since it cannot be allocated to another file. When this loss of space is multiplied by many such files, the amount of disk space lost can be quite high. Original studies at the University of California, Berkeley indicated that using, for example, a 4096-byte block size caused 45% of the disk to be wasted due to the large number of files that didn't fill blocks precisely.

A method for coping with the disk space loss is to divide each data block into fragments. The fragment or fragments can then be allotted in those instances when a file does not fill an entire block. Fragments can not be smaller than a disk sector. The usual approach is to break a block into eight fragments.

The Solaris 2.*x* file system uses an 8192-byte block and a 1024-byte fragment as its defaults.



Synchronizing Data

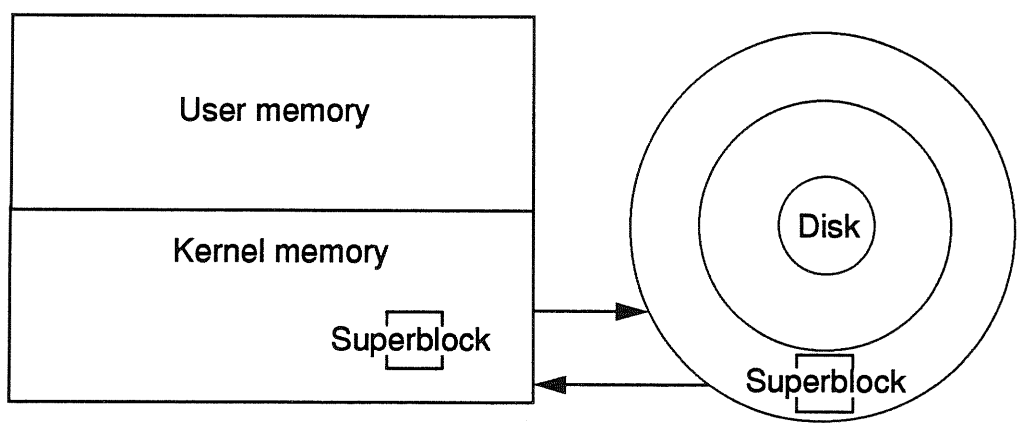
During normal file system I/O operations, the in-memory copy of data is not copied back to disk before the process continues to some other operation.

For added system performance, a copy of a file system's superblock is copied into memory during the mount process. Subsequent changes to the file system data is then reflected in the in-memory copy of the superblock.

The disk is automatically synchronized with memory data when the `sync` system call is executed by:

- The `fsflush` daemon (every 30 seconds)
- The `shutdown`, `reboot`, `halt`, or `init` commands during a graceful system shutdown

Proper system shutdown (using the above-mentioned commands) is important because the superblock information and data may be in transition if the system is suddenly aborted or system power is shut off unexpectedly.



The `fsck` (file system check) program is used to check and repair file system inconsistencies caused by improper system shutdowns.

How the `fsck` Program Works

The `fsck` command uses known parameters and redundant information to check the disk for inconsistencies.

For example, each inode contains a *link counter* that is incremented each time a link to the inode is created and decremented each time a link is removed. The `fsck` program checks whether the value of each inode's link counter is equal to the number of directory entries containing the inode's number.

Another example is a data block should never be claimed by more than one inode except for small files using file system fragments. The `fsck` program checks whether any data blocks are claimed by more than one inode.

The above examples show how the `fsck` program uses information contained in inodes and directories. The `fsck` command also checks information contained in two file system summary structures.

Inconsistencies Checked by the `fsck` Program

The `fsck` utility checks for inconsistencies in all of these structures. The most commonly corrupted item in a file system is the summary information associated with the superblock. This area is modified with every change to the file system's blocks or inodes.

Inconsistencies, checked in order, are as follows:

1. Blocks claimed by one or more inode and the free list.
2. Blocks claimed by an inode or the free list outside the range of the file system.
3. Incorrect link counts.
4. Incorrect directory sizes.
5. Bad inode format.
6. Blocks not accounted for anywhere.
7. Directory checks, files pointing to unallocated inodes, and inode numbers out of range.
8. Super Block checks: more blocks for inodes than there are in the file system.
9. Bad free block list format.
10. Total free block and/or free inode count incorrect.

Caution: The `fsck` program should *never* be run on a busy file system. Because of its multipass processing, the `fsck` utility would report errors and possibly delete files as users were trying to read or write their data. Run the `fsck` utility in single-user mode or on unmounted file systems *only*. Be aware that you cannot unmount the / (`root`) and `/usr` file systems.

When the File System Check Program Is Run

When the Solaris 2.*x* operating system is booted, the `fsck` program checks the state of each file system. If each file system state is determined clean, the `fsck` program exists without further checking.

The `fsck` program runs in two modes, *noninteractive* and *interactive.*

Noninteractive Mode

The `fsck` utility may be run in the noninteractive mode by the system during a normal boot up. In this mode on a corrupted file system, it only makes changes to the file system that are known to be correctable without operator intervention. If an unexpected inconsistency is found, the `fsck` program terminates with a non-zero exit status, leaving the system in single-user mode. The system administrator must then run the `fsck` utility interactively.

Interactive Mode

When running interactively, the `fsck` program lists each problem it finds followed by a suggested corrective action. The system administrator must decide if the correction is to be made.

Using the `fsck` Program

The following examples show how to use the `fsck` program interactively.

Without any arguments, the `fsck` program checks those entries in the `/etc/vfstab` file which has a entry in the `device to fsck` field and have a non-zero numeric entry in the `fsck pass` field.

```
# fsck
```

- Force a preen of a file system.

```
# fsck -o f,p /export/home
```

Preen mode checks and fixes the file system non-interactively, and exits immediately if there is a problem that needs intervention.

- Check the file system in a specific partition.

```
# fsck /dev/rdsk/c0t0d0s7
```

or

```
# fsck /export/home
```

- List the backup superblocks for a partition.

```
# newfs -N /dev/rdsk/c0t0d0s7
```

- Check a file system using a backup superblock.

```
# fsck -o b 32 /dev/rdsk/c0t0d0s7
```

Using The `fsck` Program

Command format:

fsck [-F *fstype*] [-V] [-m] [*special* ...]
fsck [-F *fstype*] [-V] [- y|Y|n|N] [-o *fstype options*] [*special* ...]

Options:

F *fstype* — Specify the file system type on which to operate.

V — Echo the complete command line, but do not execute the command. This option may be used to verify and validate the command line.

y|Y — Assume a *yes* response to all questions.

n|N — Assume a *no* response to all questions.

m — Check but do not repair. This option checks that the file system is suitable for mounting, and return the appropriate exit status. If the file system is ready for mounting, the `fsck` program displays a message such as:

```
ufs fsck: sanity check: /dev/rdsk/c0t3d0s7
okay
```

Using the fsck Program

Command format (continued):

```
fsck [ -F fstype ] [ -V ] [ -m ] [ special ... ]
fsck [ -F fstype ] [ -V ] [- y|Y|n|N ] [ -o fstype options ] [ special ... ]
```

Options (continued)

o The *fstype* options.

The *fstype* options are interpreted by the file system specific part of the program. These options are specified in a comma-separated (with no intervening spaces) list of options or keyword-attribute pairs. Two of the more common fsck_ufs options are:

p The *preen* option is used during the automated start-up procedure to make "safe" repairs.

f The *force* option is used to check a file system regardless of the state of its clean flag.

The *special* Argument

The *special* argument represents the block or character special device (for example, /dev/rdsk/c0t0d0s7) on which the file system resides. In general, the character special device should be used. If a file system type supports parallel checking (for example, ufs) some file systems eligible for checking can be checked in parallel. Consult the file system-specific man page (for example fsck_ufs) for more information.

An Example with No Inconsistencies

The following is the output from the `fsck` program when no inconsistencies were discovered.

```
# fsck /dev/rdsk/c0t3d0s7
** /dev/rdsk/c0t3d0s7
** 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, 21606 free (14 frags, 2699 blocks, 0.1% fragmentation)
#
```

The first thing worth noting in this output is that the `fsck` program does its work in phases. This is one reason why it is critical to run this program on an inactive file system.

Also note the last line. This line lists the following values for the file system:

- The number of files used (2 files)
- The number of Kbytes used (9 used)
- The number of Kbytes free (21606 free)
- A break down of the free space into free blocks (2699 blocks) and free block fragments (14 frags)
- The ratio of free block fragments to total Kbytes (0.1% fragmentation)

Adjusting a Link Counter

If the `fsck` program discovers any inconsistencies during interactive operation, the program prompts the operator about what action to take.

In this example, the `fsck` program discovers that the value of a directory inode's link counter and the actual number of directory links are inconsistent.

```
** Phase 4 - Check Reference Counts
LINK COUNT DIR I=2 OWNER=root MODE=40755
SIZE=512 MTIME=Jun 13 15:59 1990 COUNT 4 SHOULD BE 3
ADJUST? y
```

Correcting a link counter is a safe action, so the operator responds by typing `yes` or `y` and pressing Return.

Salvaging the Free List

In this example, the `fsck` program discovers the unallocated block count and the free block number listed in the superblock are inconsistent.

```
** Phase 5 - Check Cyl groups
CG 0: BAD MAGIC NUMBER
FREE BLK COUNT(S) WRONG IN SUPERBLK
SALVAGE? y
```

Salvaging the super block is another safe action, so the operator again responds by typing `yes` or `y` and pressing Return.

Reconnecting an Allocated but Unreferenced File

Sometimes saying `yes` to the `fsck` program results in an action that requires additional work.

In this example, the `fsck` program discovers an inode that is allocated but is unreferenced (not linked in any directory). A `yes` response to the "`RECONNECT?` question results in the inode being linked to the `lost+found` directory with its name being its number.

```
** Phase 3 - Check Connectivity
UNREF FILE I=788 OWNER=root MODE=100644
SIZE=19994 MTIME=May 18 10:49 1989
RECONNECT? y
```

After concluding the interaction with the `fsck` utility (and mounting the file system) the system administrator investigates the file.

1. Check the file's type.

```
# file /export/home/lost+found/788
/export/home/lost+found/788:     Data
```

2. If the file type is ASCII text, use `cat` or an editor to view the file. If the file is associated with an application, for example a FrameMaker™ document, use that application to view the file. In the above example the file is a data file so the system administrator uses the `more` command to view the text contained in the file.

```
# more /export/home/lost+found/788
```

If the `lost+found` file is intact, copy it to its correct location.

Summary

In this lesson, you learned that:

- Disk space used by directories and files is displayed with the `du` command.
- Summarize disk usage by file system using the `dk` command.
- Identify files that match a specified criterion using the `find` command.
- List disk usage by user name using the `quot` command.
- Improper system shutdown may cause file system corruption.
- The `fsck` program uses redundant information and known parameters to check for file system inconsistencies.
- Many corrective actions taken by the `fsck` program are safe but two, clearing and removing, lead to lost data.
- In addition to inodes and data blocks, file systems contain file system block fragments, cylinder groups, cylinder group blocks, and super blocks.

Exercise 4-1

Part I

Write down your answers to the following questions:

1. Use the `df -k` command to identify the largest file system on your system.

   ____

2. Use the `du` command to identify the largest directory in the `/var/spool` directory.

   ____

3. Identify the command to used to find all the files owned by `uucp` user in the / (`root`) file system.

   ____

4. Identify the amount of disk space (in Kbytes) used by the `lp` user in the / (`root`) file system.

   ____

Part II

Write down your answers to the following questions.

1. What is the purpose of the `fsck` program?

   ____
   ____
   ____

2. What might cause file system inconsistencies or corruption?

   ____
   ____

3. Identify four types of inconsistences checked by the `fsck` program.

   ____
   ____
   ____
   ____

Backup and Recovery 5

Objectives

Upon completion of this lesson, you will be able to:

- Dump a file system to tape.
- Restore files or a file system from tape.
- Recover the / (`root`) or `/usr` file system.

Reference

SunOS 5.0 Routine System Administration Guide, Chapter 8, "Backing Up Files and File Systems," and Chapter 9, "Restoring Files and File Systems"

Introduction

One of a system administrator's most important tasks is to ensure that a system's data is protected from system failures, natural disasters, and accidental deletions.

This lesson focuses on the commands used by system administrators to back up and restore file systems.

Why Perform Backups?

There are three main reasons for doing backups:

- To safeguard your data against a system crash or some natural disaster.
- To protect your users' files against accidental deletion.
- To ensure a smooth transition of data when reinstalling or upgrading a system.

Another important reason is job security; it is the system administrator's job to make sure copies of all system data are available.

Types of Backups

Different types of backups include:

- Backing up a file system, which is also referred to as a *full dump*; this copies the contents of an entire file system.
- Backing up only those files that have changed since the last lower level dump; referred to as an *incremental dump*.
- Selective backups; selected files can be backed up by specifying path names on the command line.
- *Multivolume* backups, meaning the data to be copied requires more than one tape reel, tape cartridge, or diskette.

Types of Backup Commands

There are several different backup commands available:

- `ufsdump/ufsrestore`
- `cpio` the -h option will allow you to get bar tapes like tar
- `tar`
- `dd`

The focus of this lesson is on using the `ufsdump` and `ufsrestore` commands; the other three commands are summarized in an appendix at the end of this module.

Backup Planning Considerations

Keep these considerations in mind when planning backup strategies:

- Which file systems should be backed up and how often
- What backup schedule is used
- What tape device(s) will be used for backups

Types of Tape Devices

Backup devices are usually magnetic tape units. On Sun systems, there
The following chart identiare four different tape devices available.

The following chart identifies Sun's tape device specifications:

Media Type	Density	Capacity	Tape Length
1/2-inch tape	800-6250 bpi	40-150 Mbytes	2300 feet
1/4-inch cartridge	800-6250 bpi	60-150 Mbytes	450-600 feet
Exabyte 8-mm cartridge	----	2.3 Gbytes 5.0 Gbytes	6000 feet 13000 feet
4-mm DAT cartridge	----	TBD	TBD

Some Sun workstations come with a diskette drive, but since the diskette's capacity is approximately 1.44 megabytes (Mbytes), it is not recommended for system backups.

The 1/4-inch Tape Device

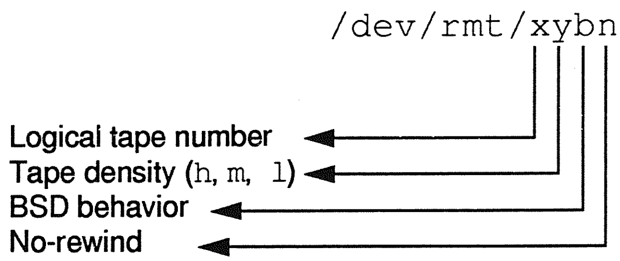
All cartridges for the 1/4-inch tape devices come in two capacities, approximately 450 feet and 600 feet of tape per cartridge. These tape devices read in three formats `QIC-11`, `QIC-24`, and `QIC-150`.

Format	Tracks	Length	Capacity
`QIC-11`	4	450 feet	20 Mbytes
`QIC-24`	9	450 feet 600 feet	45 Mbytes 60 Mbytes
`QIC-150`	18	600 feet	150 Mbytes

All tape device names are described on the following pages.

Tape Device Names

All tape devices, regardless of their type, are referenced by their logical device names. The logical tape device names use the following format:



Example:

`/dev/rmt/0->/devices/sbus@1,f8000000/esp@0,800000/st@4,0:`

Tape device names are numbered from 0, independently of their type, and have several different parameters:

Tape density — Three density values can be given: `h` (high), `m` (medium) and `l` (low). This parameter has a different value according to the type of tape device. For a `QIC-150` tape drive, all parameters cause the drive to have a capacity of 150 Mbytes.

BSD behavior — When a `b` is specified, the drive assumes BSD behavior. This means that when reading past an end of file mark, it returns the first record of the next file, and when closing the no-rewind device it skips a tape space forward.

No-rewind — By including the letter `n` at the end of the device name, the tape is not rewound when the current tape operation is completed.

Tape Device Names

Which SCSI Tape Device Name Do I Use?

Density Parameter	1/4-inch Cartridge Density	1/2-inch Tape Front-loaded
null	default preferred (highest)	default preferred (highest)
`l`	`QIC-11 format`	800 bpi
`m`	`QIC-24 format`	1600 bpi
`h`	`QIC-150`	6250 bpi

The *preferred* density for a 60-Mbyte `QIC-24` 1/4-inch cartridge drive is 1600 bpi. The preferred density for a 150-Mbyte `QIC-150` 1/4-inch cartridge is 6250 bpi.

The 18-track cartridge drive can only write in `QIC-150` format, so there is no reason to specify a density parameter.

The Exabyte 8-mm tape drives can write only at the preferred density, which is their highest capacity.

The `ufsdump` Command

The `ufsdump` command is used to backup a file system, which can be a full or incremental dump of the entire file system or individual files and directories.

Command format:

`ufsdump` *options* `[` *arguments* `]` *files_to_dump*

Options:

`0-9` Specify the *dump level* option. Level 0 is the lowest level (called the *epoch* level or a *full dump*), and level 9 is the highest level.

`a` Create an on-line archive of the file names dumped to tape.

`f` Specify the device to which the files are written. It requires an argument which is the device name.

`u` Update the dump record (`/etc/dumpdates`) with the date and dump level of this file system backup.

`c` Dump to a cartridge tape and set the blocking factor to 126 blocks.

files_to_dump
The *files_to_dump* can be the raw or block file system device name (`/dev/rdsk/c0t0d0s0`), the file system name (`/export/home`), or a file or directory name (`/export/home/lister`).

The blocking factor is the number of tape blocks (512 bytes) to write before inserting an *inter-block gap.*

When the `ufsdump` command is used to back up individual files or directories, the dump level will be set to 0.

The ufsdump Command

The /etc/dumpdates File

The u option of the ufsdump command creates or updates the /etc/dumpdates file, which contains a record of the date and level of each successful file system dump.

```
# cat /etc/dumpdates
/dev/rdsk/c0t2d0s6 0 Tue Dec  8 11:12:27 1992
/dev/rdsk/c0t2d0s0 0 Thu Dec 10 17:44:02 1992
/dev/rdsk/c0t2d0s4 0 Thu Dec 10 16:42:21 1992
/dev/rdsk/c0t2d0s3 0 Thu Dec 10 17:20:20 1992
#
```

Backup Preparation Steps

Here are some considerations before backing up your file systems:

- Check for system activity

  The best time to do backups on a busy system is when system activity is low, such as early in the morning or late in the evening.

- Bring the system to run level S

  It is *very important* that your backups are performed on *idle* file systems. The `ufsdump` command makes two passes of the data. The first pass gathers the inode information, and the second pass picks up the data blocks. If a file system is active during the back up, a file or directory could change between the first and second pass. The end result is a defective backup.

  If, for availability reasons, you can not take your system down to run level S, you should at least unmount the file system you are about to backup.

- Notify all users about system unavailability

  If you are going to back up a server, notify users and NFS distributed file system clients that the system will be unavailable during the back up.

  Use the `wall` or `rwall` commands to notify local and remote users:

  ```
  # rwall mars pluto orion
  the system will be unavailable from noon to 1 p.m.
  today for backups
  ```
  Press Control-D

  If you use the `shutdown` command to bring the system to run level S, users and NFS clients will be sent several messages about the impending shutdown.

- It is a good idea to verify the file system with the `fsck` program.

Using the `ufsdump` Command

Use the following `ufsdump` command to perform a full dump of the `/export/home` file system using a `QIC-150` tape drive (with a 600-foot tape:

```
# ufsdump 0uf /dev/rmt/0 /export/home
  DUMP: Date of this level 0 dump: Thu Dec 10 17:05:22 1992
  DUMP: Date of last level 0 dump: the epoch
  DUMP: Dumping /dev/rdsk/c0t2d0s7 (/export/home) to /dev/rmt/0
  DUMP: mapping (Pass I) [regular f iles]
  DUMP: mapping (Pass II) [directories]
  DUMP: estimated 128952 blocks (62.96MB)
  DUMP: Writing 32 Kilobyte records
  DUMP: dumping (Pass III) [directories]
  DUMP: dumping (Pass IV) [regular f iles]
  DUMP: 42.38% done, f inished in 0:06
  DUMP: 86.26% done, f inished in 0:01
  DUMP: level 0 dump on Thu Dec 10 17:05:22 1992
  DUMP: Tape rewinding
  DUMP: 128952 blocks (62.96MB) on 1 volume
  DUMP: DUMP IS DONE
#
```

The `s` (size) and the `c` (cartridge) options identify the number of tapes needed to do the full dump on a cartridge tape device.

```
# ufsdump 0ucsf 1500 /dev/rmt/0 /export/home
  DUMP: Date of this level 0 dump: Thu Dec 10 17:20:20 1992
  DUMP: Date of last level 0 dump: the epoch
  DUMP: Dumping /dev/rdsk/c0t2d0s7 (/export/home) to /dev/rmt/0
  DUMP: mapping (Pass I) [regular f iles]
  DUMP: mapping (Pass II) [directories]
  DUMP: estimated 129014 blocks (63.00MB) on 0.42 tape(s).
  DUMP: Writing 63 Kilobyte records
  DUMP: dumping (Pass III) [directories]
  DUMP: dumping (Pass IV) [regular f iles]
  DUMP: 46.78% done, f inished in 0:05
  DUMP: 94.44% done, f inished in 0:00
  DUMP: level 0 dump on Thu Dec 10 17:20:20 1992
  DUMP: Tape rewinding
  DUMP: 129014 blocks (63.00MB) on 1 volume
  DUMP: DUMP IS DONE
#
```

Notes

Sample Dump Schedules

Once a Month 0

MON	TUE	WED	THU	FRI
5	5	5	5	3
5	5	5	5	3
5	5	5	5	3
5	5	5	5	3

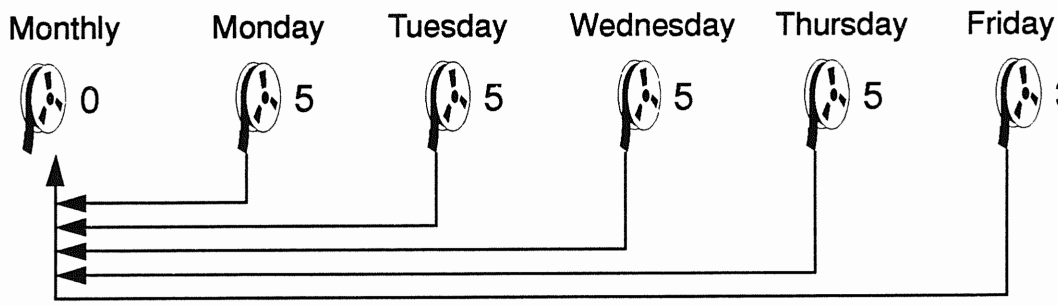
Once a Month 0

MON	TUE	WED	THU	FRI
3	4	5	6	2
3	4	5	6	2
3	4	5	6	2
3	4	5	6	2

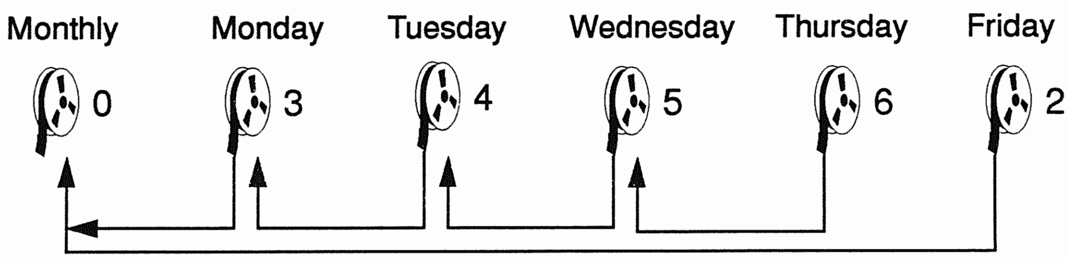
Sample Dump Schedules

Two sample dump schedules show the flexibility of the incremental dump procedure.

In the first example, each level 5 dump increases in size until Friday, when the level 3 dump restarts the process.



In the second example, each daily dump captures only the day's work until the process is restarted on Friday.



Many administrators use some form of *off-site* storage for their level 0 backups. This can involve having two level 0 dumps; one for off-site storage, the other for minor disaster recovery.

See the *SunOS 5.1 Routine System Administration Guide* for an excellent discussion of the issues involved in making backups, planning for enough tapes and the frequency of backups required for each partition.

Note: Your dump schedule will impact the number of tapes required in the event of a full/complete restore.

The `ufsrestore` Command

The `ufsrestore` command extracts files from a backup created by the `ufsdump` command.

Command format:

`ufsrestore` *options* `[` *arguments* `]` `[` *filename* `.... ]`

Options:

`t` List the table of contents of the backup.

`x` Restore only the files named on the command line.

`r` Restore the entire backup.

`i` Perform an interactive restore.

`a` *archive_file*
: Take the table of contents information from the named *archive_file* rather than the tape. Until you actually need to extract a file, the backup volume does not need to be mounted.

`f` *dump_file*
: Use *dump_file* as the device to restore from.

`v` Display pathnames as they are being restored (verbose mode).

Using the `ufsrestore` Command

Example:

1. Load the tape on the tape drive.
2. Become superuser.
3. Display the tape's contents to verify whether the file is on the tape, and to identify the correct path name of the file to be restored:

```
# ufsrestore tvf /dev/rmt/0
 2          .
 3          ./lost+found
 5120       ./export
 10240      ./home
 15360      ./opt
 20480      ./usr
 25600      ./var
 30720      ./var/sadm
 35840      ./var/sadm/install
 40960      ./var/sadm/install/admin
 41070      ./var/sadm/install/admin/default
 46080      ./var/sadm/install/logs
```

4. Once you have verified that the file you are looking for is on the tape, extract the individual file:

```
# cd /var/tmp
# ufsrestore xvf /dev/rmt/0 ./etc/passwd
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
Set directory mode, owner, and times.
set owner/mode for '.'? [yn] n
```

Enter the volume number which contains the desired file. Tape volumes start from `1`.

Using the `ufsrestore` Command

Example (continued):

Files from the dump volume are usually restored into the `/var/tmp` directory to avoid overwriting existing files by accident.

Performing a Remote Backup

The `ufsdump` and `ufsrestore` commands can be used to perform a backup or restore on a remote tape device.

The following two steps are needed to perform a remote backup or restore.

- You must have root access privileges on the system with the tape device.
- Specify the *server:tape_device* in the `ufsdump` or `ufsrestore` command line.

Example:

```
# ufsdump 0uf mars:/dev/rmt/0 /export/home
 DUMP: Date of this level 0 dump: Fri Mar 5 09:46:26 1993
 DUMP: Date of last level 0 dump: the epoch
 DUMP: Dumping /dev/rdsk/c0t3d0s7 (/export/home) to /dev/rmt/0 on host mars
 DUMP: mapping (Pass I) [regular f iles]
 DUMP: mapping (Pass II) [directories]
 DUMP: estimated 14278 blocks (6.97MB)
 DUMP: Writing 32 Kilobyte records
 DUMP: dumping (Pass III) [directories]
 DUMP: dumping (Pass IV) [regular f iles]
 DUMP: level 0 dump on Fri Mar 5 09:46:26 1993
 DUMP: Tape rewinding
 DUMP: 14278 blocks (6.97MB) on 1 volume
 DUMP: DUMP IS DONE
#
```

Interactive Restore

The `i` option of the `ufsrestore` command provides a command line interface for verifying what files are on the tape and gives you the ability to pick and choose which files to restore.

1. Change to a temporary directory and start the command.

```
# cd /var/tmp
# ufsrestore ivf /dev/rmt/0
```

2. Display the contents of the directory structure on the dump volume.

```
ufsrestore> ls
```

   Files added/marked for restore are displayed with an asterisk (*).

3. Change directory within the dump volume.

```
ufsrestore> cd directory
```

4. Add a file to the list of files to be extracted.

```
ufsrestore> add filename
```

5. Toggle verbose mode off/on to display inode numbers.

```
ufsrestore> verbose
```

6. Delete a file from the list of files to be extracted.

```
ufsrestore> delete filename
```

7. Extract the selected files from the dump volume.

```
ufsrestore> extract
```

8. Exit the interactive restore once the files are extracted.

```
ufsrestore> quit
```

Interactive Restore

Example:

```
# cd /var/tmp
# ufsrestore ivf /dev/rmt/0
Verify volume and initialize maps
Media block size is 126
Dump   date: Thu Dec 10 17:44:02 1992
Dumped from: the epoch
Level 0 dump of / on venus:/dev/dsk/c0t2d0s0
Label: none
Extract directories from tape
Initialize symbol table.
ufsrestore > ls
    2 *./                 39  devices/      30847  net/
    2 *../              5122  etc/          15360  opt/
  161  .Xauthority      5120  export/       25611  proc/
  160  .Xdefaults      10240  home/         15381  sbin/
  159  .rhosts            40  kadb          35863  tmp/
30854  .wastebasket/   25608  kernel/       30848  tmp_mnt/
   31  bin                35  lib              30  ufsboot
30872  cdrom/              3  lost+found/   20480  usr/
25610  dev/            20503  mnt/          25600  var/
ufsrestore > cd sbin
ufsrestore > add uname ifconfig
Make node ./sbin
ufsrestore > verbose
verbose mode off
ufsrestore > ls
./sbin:
 autopush     jsh          rc2          sh           sync
 bpgetfile    mount        rc3          shutdown     uadmin
 hostconfig   mountall     rc5          su           umount
*ifconfig     rc0          rc6          sulogin      umountall
 init         rc1          rcS          swapadd     *uname
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 ./sbin/ifconfig
extract file ./sbin/uname
Add links
Set directory mode, owner, and times.
set owner/mode for '.'? [yn] n
ufsrestore > quit
#
```

Restoring an Entire File System

There are several reasons why you might need to restore an entire file system:

- Adding a new disk drive

  The preliminary steps of adding a new disk were covered in a previous lesson. Once you have connected the disk, rebooted the system, and if necessary, repartitioned the disk, you would create the new file system and use the `ufsrestore` command to restore any existing data that you wanted on the new drive.

- Re-installing or upgrading the Solaris environment

  When you receive a new Solaris 2.*x* release, you may want to repartition your disk drives and rebuild the file systems. After the installation is complete, you would use the `ufsrestore` command to restore the contents of your old file systems.

- Reorganizing your disks and file systems

  You may decide after running your system for a while that you need to repartition a disk. For example, you may want to convert two rarely used small partitions into one large one to be used for long-term storage. You would have to backup the contents of both partitions, repartition the disk, create a new file system, and restore the contents of both old file systems.

- Recreating a damaged file system

  Sometimes a file system will become so badly damaged that you will have to restore it entirely. An example would be a head crash on your disk. You would have to replace the hardware before you could restore the data.

Restoring an Entire File System

Example:

The following example illustrates how to restore an existing file system onto a new larger disk partition (or disk).

1. Unmount the old file system and back up its contents.

```
# umount /opt
# fsck /opt
# ufsdump 0uf /dev/rmt/0 /opt
```

2. Repartition the new disk (if necessary), create the new file system, mount it on a temporary mount point, and restore the data.

```
# umount /opt
# newfs /dev/rdsk/c0t3d0s5
# fsck /dev/rdsk/c0t3d0s5
# mount /dev/dsk/c0t3d0s5 /mnt
# cd /mnt
# ufsrestore rvf /dev/rmt/0
# rm restoresymtable
```

3. Once the data is restored, unmount the file system, and check it with the `fcsk` program.

```
# cd /
# umount /mnt
# fsck /dev/rdsk/c0t3d0s5
```

4. Add an entry for the new file system in the `/etc/vfstab` file and mount it.

The `restoresymtable` file is created by the `ufsrestore` command and provides check-points for the restore. If you interrupt the command, you can restart it with the `R` option. The `ufsrestore` command does not remove this file when it is done.

Restoring the / (`root`) File System

Restoring the `root` file system is a longer procedure because the special files and command utilities that you need reside in the `root` file system.

The procedure is:

1. Load and the boot the release media to run level S.
2. Create the new file system, mount it, and restore the tape containing the old `root` file system.
3. Run the `installboot` program to create the *boot blocks* which reside in the first 15 sectors of the disk. The `bootblock` program contains a `ufs` file system reader that loads the `ufsboot` program into memory.
4. Unmount the new file system and run the `fsck` program on it.
5. Then reboot the system.

The same procedure would be used to restore the `/usr` file system except that you don't have to reinstall the boot blocks.

Restoring the / (root) File System

Example:

The following example illustrates how to restore the / (root) file system.

```
ok boot cdrom -sw
Boot device: /sbus/esp@0,800000/sd@6,0:c   File and args: -sw
SunOS Release 5.1 Version Generic [UNIX(R) System V Release 4.0]
INIT: SINGLE USER MODE
# tapes  ~~  adds /dev entries for attached tape drives
# ls /dev/rmt/0
/dev/rmt/0
# newfs /dev/rdsk/c0t3d0s0
newfs: construct a new file system /dev/rdsk/c0t3d0s0: (y/n)? y
/dev/rdsk/c0t3d0s0: 59760 sectors in 83 cylinders of 9 tracks, 80 sectors
30.6MB in 6 cyl groups (16 c/g, 5.90MB/g, 2688 i/g)
super-block backups (for fsck -F ufs -o b=#) at:
 32, 11632, 23232, 34832, 46432, 58032,
# fsck /dev/rdsk/c0t3d0s0
** /dev/rdsk/c0t3d0s0
** 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, 9 used, 27742 free (14 frags, 3466 blocks, 0.1% fragmentation)
# mount /dev/dsk/c0t3d0s0 /a
# cd /a
# ufsrestore rvf /dev/rmt/0
# rm restoresymtable
# cd /
# umount /a
# fsck /dev/rdsk/c0t3d0s0
** /dev/rdsk/c0t3d0s0
** Last Mounted on /a
** Phase 1 - Check Blocks and Sizes
** Phase 2 - Check Pathnames
** Phase 3 - Check Connectivity
** Phase 4 - Check Reference Counts
** Phase 5 - Check Cyl groups
1779 files, 12509 used, 15242 free (34 frags, 1901 blocks, 0.1%
fragmentation)
# cd /usr/lib/fs/ufs
# installboot bootblk /dev/rdsk/c0t3d0s0
# reboot
```

Summary

In this lesson, you learned that:

- Backups are an important safeguard against the accidental loss of your users' files, and are necessary to ensure a smooth transition of data when reinstalling or upgrading a system.
- Different types of backups include full, incremental, selective, and multivolume.
- Backup devices used on Sun systems are 1/2-inch reel tape, 1/4-inch cartridge tape, Exabyte 8-mm helical scan tape cartridge, and 4-mm DAT cartridge..
- The `ufsdump` and `ufsrestore` commands are used to backup and restore file systems.

Exercise 5-1

The purpose of this lab is to use the `ufsdump` and `ufsrestore` commands.

Procedure

Use the following steps to back up and restore the / (`root`) file system.

1. Bring the system to run level S. Then use the `ufsdump` command to dump your `root` file system to tape.

2. Use the `ufsrestore` command to make certain you have created a usable tape of your / (`root`) file system. Use the `tf` options to view the tape contents.

3. Use the restore procedure described in this lesson to restore your / (`root`) file system.

   a. Halt the system and boot from the installation CD-ROM.

      ```
      ok boot cdrom -sw
      ```

   b. Use the `tapes` command to create the tape device entry.

   c. Use the `newfs` command to create the new / (`root`) file system.

   d. Run the `fsck` command to verify the new file system.

   e. Mount the new file system on the `/a` directory.

   f. Change directory to `/a`.

   g. Use the `ufsrestore` command to restore the / (`root`) file system.

   h. Remove the `restoresymtable` file.

   i. Change directory to / (`root`).

   j. Unmount the new file system.

   k. Run the `fsck` command to verify the / (`root`) file system data.

   l. Change directory to `/usr/lib/fs/ufs`.

Exercise 5-1

m. Install the boot block.

n. Reboot the system.

Answer Key

A

Lesson 1: Disk Device Names

Exercise 1-1

Part I

1.

   a. Move a disk from one system to another.

   b. Add a new disk to a system.

   c. Access (or mount) a file system residing on a local disk.

   d. Back up a local file system.

1. Reference the devices with their logical device names.
2. `format`, `dmesg`, or `printconf`.
3. By their target address (unit number).

Part II

Follow the steps as described. Answers may vary for each system.

Lesson 2: Adding a New Disk

Exercise 2-1

Follow the steps as described. Answers may vary for each system.

Lesson 3: Mounting File Systems

Exercise 3-1

Follow the steps as described. Answers may vary for each system.

Lesson 4: Maintaing File Systems

Exercise 4-1

Part I

1. Answers may vary for each system.
2. Answers may vary for each system.
3. `find / -user uucp -print`
4. Use the `quot -a /` command. The result should be: `lp 626` (approximately).

Part II

1. To check the consistency of the file systems and make repairs, if necessary.
2. An improper shutdown such as a power failure.
3.
    a. Incorrect link count

    b. Incorrect directory sizes

    c. Bad inode format

    d. Blocks not accounted for anywhere

Lesson 5: Backup and Recovery

Exercise 5-1

Follow the steps as described. Answers may vary for each system.

1. `shutdown -y`
   `ufsdump 0uf /dev/rmt/0 /`

2. `ufsrestore tf /dev/rmt/0`

3.

   a. Press Control-D and type 0 (for run level), and press Return
      `boot cdrom -sw`

   b. `tapes`

   c. `newfs /dev/rdsk/cwtxdysz`

   d. `fsck /dev/rdsk/cwtxdysz`

   e. `mount /dev/dsk/cwtxdysz /a`

   f. `cd /a`

   g. `ufsrestore rvf /dev/rmt/0`

   h. `rm restoresymtable`

   i. `cd /`

   j. `umount /a`

   k. `fsck /dev/rdsk/cwtxdysz`

   l. `cd /usr/lib/fs/ufs`

   m. `installboot bootblk /dev/rdsk/cwtxdysz`

   n. `reboot`

Additional Backup Commands B

Reference

SunOS 5.0 Routine System Administration Guide, Chapter 8, "Backing Up Files and File Systems," and Chapter 9, "Restoring Files and File Systems"

Introduction

The following section summarizes additional Solaris 2.*x* commands that are used for backing up and restoring files.

The `tar` Command

The `tar` (tape archive) command allows you to backup single or multiple files in a directory hierarchy.

Command format:

`tar` *options* [*arguments*] *filename* ...

Options:

`c` Create a new *tarfile* using the file names specified on the command line.

`t` List the table of contents of the *tarfile.*

`x` Extract the specified files from the *tarfile.* If no filename arguments are specified, the entire archive is extracted.

`f` Use the next argument as the name of the *tarfile* rather than `/dev/rmt/0`. You can also set the environment variable TAPE. If *tarfile* is –, the `tar` command reads `stdin` and writes `stdout`.

`v` Print the file names as they are restored (verbose mode).

`B` Perform multiple reads so exactly enough bytes are read to fill a block (Necessary when using `tar` across the network).

`p` Restore the files with the permissions on tape.

The `tar` command is unaware of file systems; however, if you specify a directory as a `tar` argument, it copies the entire hierarchy below the directory.

The cpio Command

The cpio (copy in/copy out) command creates an archive of single or multiple files by taking a list of names from standard input and writing the archive to standard output, which is usually redirected to a file or a tape device. It creates directory hierarchies, if necessary. The cpio command is usually used with the ls or find commands to generate archives.

Command format:

[*command* |] cpio *options* [> *filename* ...]

Options:

One of the o, i, or, p options must be specified.

o	Create an archive by reading a list of path names generated from standard input and copies those files to standard output.
i	Extracts the archive specified by standard input which is assumed to the product of a previous cpio -o command.
p	Reads from standard input to obtain a list of file names. This option is useful for disk-to-disk copying.
B	Sets block input/output record to 5120 bytes, but does not apply when using the -p option. The default size is 512 bytes.
c	Read or write header information in ASCII-character format for portability to other platforms.
H	Read or write head information in bar, crc, odc, tar, or ustar format. To be used with the -c option.
v	Print a list of file names from the archive (verbose mode).
t	Print a list of file names. When used with the -v option, the output looks like the ls -l command.

The cpio Command

Examples:

Use the following command to create an archive of the current directory contents:

```
$ find . -print | cpio -ovcB > /dev/rmt/0
```

Extract the README file from the cpio archive created above:

```
$ cpio -ivcB README < /dev/rmt0
```

Use the find command with cpio to create an archive of files that have changed within the last week only:

```
$ find . -mtime -7 -print | cpio -ovcB > /dev/rmt/0
```

Use the find command with cpio to archive called file.list with files that begin with the name file:

```
$ find . -name 'file*' -print | cpio -ovcB > file.list
```

List the file names from the file.list archive:

```
$ cpio -ivt < file.list
-rw-r--r--   1 lister  visitors  314  Dec 13 13:54 1992, f ilea
-rw-r--r--   1 lister  visitors  628  Dec 13 13:54 1992, f ileb
-rw-r--r--   1 lister  visitors  936  Dec 13 13:54 1992, f ilec
8 blocks
$
```

The dd Command

The dd command is used to convert and copy files with various data formats.

Command format:

dd [*option=value*]

Options to the dd command are given in *option=value* pairs where the value identifies the argument.

Options:

if=*file* Specify input file name. Default is standard input.

of=*file* Specify output file name. Default is standard output.

bs=*n* Set both input and output block size. Default for both is 512 bytes.

The dd command is useful when combined with other commands to create or extract tapes from a remote tape device.

Examples:

Create a tape archive on a remote tape drive:

```
# tar cvf - scripts | rsh enterprise dd of=/dev/rmt/0
a scripts/cleanup 1 blocks
a scripts/cleanup.lists 1 blocks
a scripts/cleanup.distriblog 1 blocks
a scripts/cleanup.pulldir 2 blocks
16+0 records in
16+0 records out
#
```

The dd Command

Examples (continued):

Extract files from a tape archive on a remote tape drive:

```
# rsh enterprise dd if=/dev/rmt/0 | tar xvBpf - scripts
x scripts/cleanup 1 blocks
x scripts/cleanup.lists 1 blocks
x scripts/cleanup.distriblog 1 blocks
x scripts/cleanup.pulldir 2 blocks
16+0 records in
16+0 records out
#
```