Chapter 2

Performing Disk Administration Procedures

This chapter describes administration procedures for disks and their device files.

The major sections in this chapter are:

- "Listing the Disks on a System With hinv" on page 20
- "Formatting and Initializing a Disk With fx" on page 21
- "Adding Files to the Volume Header With dvhtool" on page 22
- "Removing Files in the Volume Header With dvhtool" on page 24
- "Displaying a Disk's Partitions With prtvtoc" on page 25
- "Repartitioning a Disk With xdkm" on page 27
- "Repartitioning a Disk With fx" on page 27
- "Creating Device Files With MAKEDEV" on page 36
- "Creating Device Files With mknod" on page 37
- "Creating Mnemonic Names for Device Files With ln" on page 38
- "Creating a System Disk From the PROM Monitor" on page 38
- "Creating a New System Disk From IRIX" on page 44
- "Creating a New System Disk by Cloning" on page 48
- "Adding a New Option Disk" on page 50

Administration procedures for filesystems and logical volumes are described in later chapters of this guide.

Listing the Disks on a System With *hinv*

You can list the disks connected to a system by giving this *hinv* command from IRIX:

```
hinv -c disk
```

The output lists the disk controllers and disks present on a system, for example:

```
Integral SCSI controller 0: Version WD33C93B, revision D
Disk drive: unit 2 on SCSI controller 0
Disk drive: unit 1 on SCSI controller 0
```

This output shows a single integral SCSI controller whose number is 0 and two disk drives. These disks are at drive addresses 1 and 2. In *hinv* output, drive addresses are called units. They are also sometimes called unit numbers. Each disk is uniquely identified by the combination of its controller number and drive address.

If you are in the PROM Monitor, you can also give the *hinv* command from the Command Monitor:

```
>> hinv
```

Output for SCSI disks looks like this:

```
SCSI Disk: scsi(0)disk(1)
SCSI Disk: scsi(0)disk(2)
```

In this output, the controller number is the "scsi" number and the drive address is the "disk" number. The type of controller isn't listed. As a rule of thumb, workstations have integral controllers and servers may have integral SCSI controllers or non-integral controllers that are SCSI or VME. On some Challenge systems, the output of *hinv* in the PROM monitor shows only disks on the boot IOP (I/O processor).

The controller number and drive addresses of disks are specified, using a variety of syntax, as arguments to the IRIX disk and filesystem commands, such as *fx*, *prtvtoc*, *dvhtool*, and *mkfs*. For example, for a disk on controller 0 at drive address 1:

- To specify the disk on an *fx* command line, the command line is:

  ```
  fx "dksc(0,1)"
  ```

- To specify the disk (actually, its volume header) on a *prtvtoc* command line, either of these two commands can be used:

  ```
  prtvtoc /dev/rdsk/dks0d1vh
  ```

  ```
  prtvtoc dks0d1vh
  ```

- To specify the disk 1 (actually, its volume header) on a *dvhtool* command line, the command is:

  ```
  dvhtool /dev/rdsk/dks0d1vh
  ```

- To specify partition 7 of the second disk above on a *mkfs* command line for an EFS filesystem, the command is:

  ```
  mkfs –t efs /dev/rdsk/dks0d1s7
  ```

Tip: You can use the Disk Manager in the System Toolchest to get information about the disks on a system. For instructions, see the section "Checking Disk Setup Information" in Chapter 6 of the *Personal System Administration Guide*.

Formatting and Initializing a Disk With *fx*

When you format a disk, you write timing marks and divide the disk into tracks and sectors that can be addressed by the disk controller. SCSI disks are shipped pre-formatted; formatting a SCSI disk is rarely required. Formatting is done by *fx*; see the fx(1M) reference page for details.

Caution: Formatting a disk results in the loss of all data on the disk. It is recommended only for experienced IRIX system administrators.

Formatting a disk destroys information about bad areas on the disk (called *bad blocks*). Identifying and handling bad blocks is also done by *fx*; see the fx(1M) reference page for details.

Caution: Using *fx* for bad block handling usually results in the loss of all data on the block. It is recommended only for experienced IRIX system administrators.

Initializing a disk consists of creating a volume header for a disk. Disks supplied by Silicon Graphics are shipped with a volume header, and initialization isn't necessary. Disks from third-party vendors or disks whose volume headers have been destroyed must be initialized to create a volume header. Initializing disks is done by *fx*. No explicit commands are necessary; *fx* automatically notices if no volume header is present and creates one. (See the section "Repartitioning a Disk With fx" in this chapter for information on invoking *fx*.) When *fx* creates a volume header, a prompt asks if you want to write the volume header; reply yes.

Tip: You can use the Disk Information window of the Disk Manager in the System Toolchest to perform disk initialization and other tasks. For more information, see the section "Formatting, Verifying, and Remaking Filesystems on a Fixed Disk" in Chapter 6 of the *Personal System Administration Guide*.

Adding Files to the Volume Header With *dvhtool*

As explained in the section "Volume Headers" in Chapter 1, the volume header of system disks must contain a copy of the program *sash*. The procedure in this section explains how to put *sash* or other programs into a volume header. Before performing this procedure, review the discussion of *dvhtool* in the section "Volume Headers" in Chapter 1.

When you add programs to the volume header of a disk, there are two sources for those programs. One is the */stand* directory of the system and the other is the */stand* directory on an IRIX software release CD. The */stand* directory on a CD (usually */CDROM/stand* after the CD is mounted) contains copies of *sash*, *fx*, and *ide* that are processor-specific.

As superuser, perform this procedure to add programs to a volume header:

1. Invoke *dvhtool* with the raw device name of the volume header of the disk as an argument, for example:

   ```
   # dvhtool /dev/rdsk/dks0d2vh
   ```

   (See the section "Device Names" in Chapter 1 for information on constructing the device name.)

2. Display the volume directory portion of the volume header by using the **vd** (volume directory) and **l** (list) commands:

```
Command? (read, vd, pt, dp, write, bootfile, or quit): vd
(d FILE, a UNIX_FILE FILE, c UNIX_FILE FILE, g FILE UNIX_FILE or l)?
        l

Current contents:
        File name        Length     Block #
        sgilabel            512           2
        sash             159232           3
```

3. For each program that you want to copy to the volume header, use the **a** (add) command. For example, to copy *sash* from the */stand* directory to *sash* in the volume header, use this command:

```
(d FILE, a UNIX_FILE FILE, c UNIX_FILE FILE, g FILE UNIX_FILE or l)?
        a /stand/sash sash
```

   As another example, to copy *sash* from a CD to an IP20 or IP22 system (an Indy™), use this command:

```
(d FILE, a UNIX_FILE FILE, c UNIX_FILE FILE, g FILE UNIX_FILE or l)?
        a /CDROM/stand/sashARCS sash
```

   CDs contain multiple processor-specific versions of *sash*; Table 1-3 lists the version of *sash* for each processor type.

4. Confirm your changes by listing the contents of the volume with the **l** (list) command:

```
(d FILE, a UNIX_FILE FILE, c UNIX_FILE FILE, g FILE UNIX_FILE or l)?
        l

Current contents:
        File name        Length     Block #
        sgilabel            512           2
        sash             159232           3
```

5. Make the changes permanent by writing the changes to the volume header using the **quit** command to exit this “submenu” and the **write** command:

```
(d FILE, a UNIX_FILE FILE, c UNIX_FILE FILE, g FILE UNIX_FILE or l)?
        quit

Command? (read, vd, pt, dp, write, bootfile, or quit): write
```

6. Quit *dvhtool* by giving the **quit** command:

```
Command? (read, vd, pt, dp, write, bootfile, or quit): quit
```

Removing Files in the Volume Header With *dvhtool*

Caution: The procedure in this section can result in the loss of data if it is not performed properly. It is recommended only for experienced IRIX system administrators.

The procedure below can be used to remove logical volume labels (for example *xlvlab*) and files (for example *sash*) from the volume header of a disk. Before performing this procedure, review the discussion of *dvhtool* in the section "Volume Headers" in Chapter 1.

1. Using *hinv,* determine the controller and drive addresses of the disk that has the volume header you want to change. In this procedure, the example commands and output assume that the disk is on controller 0, drive address 2. Substitute the controller and drive addresses of your disk.
2. As superuser, invoke *dvhtool* with the raw device name of the volume header of the disk, for example:

   ```
   # dvhtool /dev/rdsk/dks0d2vh
   ```

   (See the section "Device Names" in Chapter 1 for information on constructing the device name.)
3. Display the volume directory portion of the volume header by answering two prompts:

```
Command? (read, vd, pt, dp, write, bootfile, or quit): vd
(d FILE, a UNIX_FILE FILE, c UNIX_FILE FILE, g FILE UNIX_FILE or l)?
        l

Current contents:
        File name        Length     Block #
        sgilabel            512           2
        xlvlab            10752           3
        lvlab2              512          26
```

4. Use the **d** command to delete the file you want to delete, for example *xlvlab*:

```
(d FILE, a UNIX_FILE FILE, c UNIX_FILE FILE, g FILE UNIX_FILE or l)?
        d xlvlab
```

5. To delete additional files, continue to use the **d** command, for example:

```
(d FILE, a UNIX_FILE FILE, c UNIX_FILE FILE, g FILE UNIX_FILE or l)?
        d lvlab2
```

6. List the volume directory again to confirm that the files are gone:

```
(d FILE, a UNIX_FILE FILE, c UNIX_FILE FILE, g FILE UNIX_FILE or l)?
        l

Current contents:
        File name        Length     Block #
        sgilabel            512           2
```

7. Exit this “menu” and write the changes to the volume header:

```
(d FILE, a UNIX_FILE FILE, c UNIX_FILE FILE, g FILE UNIX_FILE or l)?
        q

Command? (read, vd, pt, dp, write, bootfile, or quit): write
```

8. Quit *dvhtool*:

```
Command? (read, vd, pt, dp, write, bootfile, or quit): quit
```

Displaying a Disk’s Partitions With *prtvtoc*

Use the *prtvtoc* command to get information about the size and partitions of a disk. Only the superuser can use this command. The command is:

prtvtoc *device*

device is optional; when it is omitted, *prtvtoc* displays information for the system disk. *device* is the raw device name (see the section “Device Names” in Chapter 1) of the disk volume header. The */dev/rdsk* portion of the device name can be omitted if desired. For example, for a SCSI disk that is drive address 1 on controller 0, *device* is dks0d1vh. (See the section “Device Names” in Chapter 1 for more information on device names.)

An example of the output of *prtvtoc* is:

```
Printing label for root disk

* /dev/rdsk/dks0d1vh (bootfile "/unix")
*     512 bytes/sector
*      85 sectors/track
*       9 tracks/cylinder
*       3 spare blocks/cylinder
*    2726 cylinders
*       4 cylinders occupied by header
*    2722 accessible cylinders
```

```
*
* No space unallocated to partitions

Partition  Type  Fs   Start: sec   (cyl)    Size: sec    (cyl)  Mount Directory
 0          efs  yes        3048  (   4)       51054   (  67)   /
 1          raw            54102  (  71)       81534   ( 107)
 6          efs  yes      135636  ( 178)     1941576   (2548)   /usr
 8       volhdr                0  (   0)        3048   (   4)
10       volume                0  (   0)     2077212   (2726)
```

The first section of the output shows the device parameters that can be used to figure out the capacity of the disk (remember that 1 kilobyte = 1024 bytes and 1 megabyte = 1048576 bytes):

512 bytes/block * 85 blocks/track * 9 tracks/cylinder * 2722 cylinders
= 1,066,152,960 bytes
= 1,041,165 kilobytes
= 1,016 megabytes

The partition table at the end of the output lists the partitions, their type (name or filesystem type), whether they contain a filesystem, their location on the disk (start and size in blocks and cylinders), and mount directory for filesystems. The partitions in this output are shown graphically in Figure 1-4.

Another example of the output of *prtvtoc*, showing fractional numbers of cylinders per partition, is:

```
# prtvtoc /dev/rdsk/dks0d2vh
* /dev/rdsk/dks0d2vh (bootfile "/unix")
*     512 bytes/sector
*     115 sectors/track
*      20 tracks/cylinder
*      20 spare blocks/cylinder
*    3865 cylinders
*       2 cylinders occupied by header
*    3863 accessible cylinders
*
* No space unallocated to partitions
Partition  Type  Fs   Start: sec   (cyl)    Size: sec    (cyl)  Mount
Directory
   0          xfs  yes        4560  (   2)      8684310  (3808.9)  /usr/people
   1          raw          8688870 (3810.9)      125000  (  54.8)
   8       volhdr                0  (   0)         4560   (   2)
  10       volume                0  (   0)      8813870  (3865.7)
```

Repartitioning a Disk With *xdkm*

Disks can be repartitioned using the graphical user interface of the *xdkm* command. Information about *xdkm* is available from its online help.

Repartitioning a Disk With *fx*

Caution: The procedure in this section can result in the loss of data if it is not performed properly. It is recommended only for experienced IRIX system administrators.

Repartitioning disks is done from the command line by the *fx* command. There are two versions of this program, a standalone version and an IRIX version. The standalone version is invoked from the Command Monitor, which enables you to repartition the system disk. Option disks can be repartitioned using the IRIX version. Two of the following subsections describe how to invoke each version of *fx*:

- "Invoking fx From the Command Monitor"
- "Invoking fx From IRIX"

The standard partition layouts described in the section "System Disks, Option Disks, and Partition Layouts" in Chapter 1 are "built into" *fx*. You can partition a disk using one of the standard layouts or you can create custom partition layouts. Two subsections describe how to create standard and custom partition layouts:

- "Creating Standard Partition Layouts"
- "Creating Custom Partition Layouts"

The final subsection, "After Repartitioning," describes how to proceed after the repartitioning is complete.

To repartition a disk, start with the first subsection, "Before Repartitioning." Then choose one of the sections on invoking *fx*, choose one of the sections on creating partitions, and finish up with the section "After Repartitioning."

Before Repartitioning

Caution: Repartitioning a disk makes the data on the disk inaccessible (you must repartition back to the original partitions to get to it).

Before repartitioning a disk, if there is *any* valuable data on the disk to be repartitioned, make a backup of the files on the disk. If the disk is a system disk and you plan to copy the files from the backup to the disk after repartitioning, you must use either the System Manager or the *Backup* command. Only backups made with *Backup* or the System Manager will be available to the system from the System Recovery menu of the System Maintenance Menu. The System Manager is the preferred method of the two and is described completely in the *Personal System Administration Guide*. Other commands require a full system installation to operate correctly.

Invoking *fx* From the Command Monitor

The procedure in this section describes how to invoke the standalone version of *fx* from the Command Monitor. It is only necessary for the system disk. You can use the IRIX version of *fx* for other disks (see the next section "Invoking fx From IRIX").

1. Shut the system down into the System Maintenance Menu.
2. Bring up the Command Monitor by choosing the fifth item on the System Maintenance Menu.
3. Identify the copy of *fx* that you will boot. Some possible locations are: *fx* in the */stand* directory of the system disk or *fx* on an IRIX software distribution CD in a CD-ROM drive on the local system or on a remote system.

   A single copy of *fx* is in the */stand* directory, but IRIX software distribution CDs contain several processor-specific versions of *fx*. Booting *fx* from a CD on a local CD-ROM drive requires a processor-specific copy of *sash* on the CD, too.

   Table 2-1 shows the versions of *sash* and *fx* to use when you are using them from a source that provides several processor-specific versions.

Table 2-1 *sash* and *fx* Versions

Processor Type	***sash* Version**	***fx* Version**
IP17	sashIP17	fx.IP17
IP19, IP20, IP22	sashARCS	fx.ARCS
IP21, IP26	sash64	fx.64

4. Boot *fx* from the Command Monitor. The command to boot *fx* depends upon the location of the copy of you are booting.

- This command boots *fx* from the */stand* directory on the system disk:

  ```
  >> boot stand/fx --x
  ```

- This command boots *fx* from an IRIX software release CD in a local CD-ROM drive, where the CPU type of the system is IP19, IP20, or IP22 and the CD-ROM drive is at drive address 4 on controller 0:

  ```
  >> boot -f dksc(0,4,8)sashARCS dksc(0,4,7)stand/fx.ARCS --x
  ```

- This command boots *fx* from an IRIX software release CD in a CD-ROM drive mounted at */CDROM* on a remote system named dist, where the CPU type of the local system is IP21 or IP26:

  ```
  >> boot -f bootp()dist:/CDROM/stand/fx.64 --x
  ```

5. *fx* prompts you for each part of the disk name. The default answer is in parentheses and matches the system disk. The prompts are:

   ```
   fx: "device-name" = (dksc)
   fx: ctlr# = (0)
   fx: drive# = (1)
   fx: lun# = (0)
   ```

   The default device name is dksc, which indicates a SCSI disk on a SCSI controller. (See the fx(1M) reference page for other device names.) The next prompt asks you to specify the disk controller number and the next one the drive address (unit) of the disk. The final prompt asks for the lun (logical unit) number. The logical unit number is typically used by only a few SCSI devices such as RAIDs (an array of disks with built-in redundancy) to address disks within the device. For regular disks, use logical unit number 0.

   For each prompt, press the **<Enter>** key for the default value or enter another value, followed by **<Enter>**.

   Once you have answered the prompts, *fx* performs a disk controller test and you see the *fx* main menu:

```
---- please choose one (? for help. .. to quit this menu)----
[exi]t              [d]ebug/              [l]abel/
[b]adblock/         [exe]rcise/           [r]epartition/
fx>
```

   The *exit* option quits *fx*, while the other commands take you to submenus. (The slash [/] character after a menu option indicates that choosing that option leads to a submenu.) For complete information on all *fx* options, see the fx(1M) reference page.

Invoking *fx* From IRIX

The procedure in this section describes how to invoke *fx* from IRIX.

1. Make sure that the disk drive to be partitioned is not in use. That is, make sure that no filesystems are mounted and no programs are accessing the drive.
2. As superuser, give the *fx* command:

   ```
   # fx "controller_type(controller,address,logical_unit)"
   ```

   The variables are:

controller_type	The controller type. It is dksc for SCSI controllers. For other controller types, see the fx(1M) reference page.
controller	The controller number for the disk.
address	The drive address of the disk.
logical_unit	The logical unit number for the device. It is used by only a few SCSI devices such as RAIDs (an array of disks with built-in redundancy) to address disks within the device. The *logical_unit* is normally 0.

   If you give the *q*

   command without arguments, you are prompted for these values.

   fx first performs a controller test, then displays this menu:

```
---- please choose one (? for help. .. to quit this menu)----
[exi]t              [d]ebug/             [l]abel/
[b]adblock/         [exe]rcise/          [r]epartition/
fx>
```

   The *exit* option quits *fx*, while the other commands take you to submenus. (The slash [/] character after a menu option indicates that choosing that option leads to a submenu.) For complete information on all *fx* options, see the fx(1M) reference page.

Creating Standard Partition Layouts

This section shows the procedure for repartitioning a disk so that it has one of the standard partition layouts. The example used in this section is to change a disk from separate root and usr partitions to a combined root and usr partition.

1. From the *fx* main menu, choose the **repartition** option:

```
---- please choose one (? for help. .. to quit this menu)----
[exi]t              [d]ebug/             [l]abel/
[b]adblock/         [exe]rcise/          [r]epartition/
fx> repartition

----- partitions-----
part  type       cyls             blocks          Megabytes   (base+size)
  0: efs        4 + 67         3024 + 50652         1 + 25
  1: raw       71 + 108       53676 + 81648        26 + 40
  6: efs      179 + 2547     135324 + 1925532      66 + 940
  8: volhdr     0 + 4             0 + 3024          0 + 1
 10: volume     0 + 2726          0 + 2060856       0 + 1006

capacity is 2061108 blocks

----- please choose one (? for help, .. to quit this menu)-----
[ro]otdrive           [o]ptiondrive        [e]xpert
[u]srrootdrive        [re]size
```

You see the partition layout for the disk that you specified when *fx* was started, followed by the **repartition** menu. The **rootdrive**, **usrrootdrive**, and **optiondrive** options are used for standard partition layouts, the **resize** option is used for custom partition layouts, and the **expert** option, which appears only if the *fx* is invoked with the **-x** option. The **expert** option enables custom partitioning functions. These functions can severely damage the disk when performed incorrectly, so they are unavailable unless explicitly requested with **-x**.

2. To create a combined root and usr partition, choose the **rootdrive** option.

```
fx/repartition> rootdrive
```

3. A prompt appears that asks about the partition type. The possible types are shown in Table 2-1. For this example, choose efs:

```
fx/repartition/rootdrive: type of data partition = (xfs) efs
```

4. A warning appears; answer yes to the prompt after the warning:

```
Warning: you will need to re-install all software and restore user data
from backups after changing the partition layout.  Changing partitions
will cause all data on the drive to be lost.  Be sure you have the drive
backed up if it contains any user data.  Continue? yes

----- partitions-----
part  type       cyls             blocks          Megabytes   (base+size)
  0: efs        4 + 2614       3024 + 1976184       1 + 965
  1: raw     2618 + 108     1979208 + 81648       966 + 40
  8: volhdr     0 + 4             0 + 3024          0 + 1
 10: volume     0 + 2726          0 + 2060856       0 + 1006

capacity is 2061108 blocks

----- please choose one (? for help, .. to quit this menu)-----
[ro]otdrive        [u]srrootdrive     [o]ptiondrive      [re]size
```

The partition layout after repartitioning is displayed and the **repartition** submenu appears again.

5. To return to the *fx* main menu, enter `..` at the prompt:

```
fx/repartition> ..

----- please choose one (? for help, .. to quit this menu)-----
[exi]t               [d]ebug/             [l]abel/
[b]adblock/          [exe]rcise/          [r]epartition/
fx>
```

Creating Custom Partition Layouts

The following procedure describes how to repartition a disk so that it has a custom partition layout. As an example, this procedure repartitions a 380 MB SCSI drive to increase the size of the root partition.

1. At the *fx* main menu, choose the **repartition** command:

```
---- please choose one (? for help. .. to quit this menu)----
[exi]t              [d]ebug/             [l]abel/
[b]adblock/         [exe]rcise/          [r]epartition/
fx> repartition
```

```
----- partitions-----
part  type       cyls             blocks          Megabytes   (base+size)
  0: efs        7 + 80         2835 + 32400         1 + 16
  1: rawdata   87 + 202       35235 + 81810        17 + 40
  6: efs      289 + 1269     117045 + 513945       57 + 251
  7: efs        7 + 1551       2835 + 628155        1 + 307
  8: volhdr     0 + 7             0 + 2835          0 + 1
 10: entire     0 + 1550          0 + 630990        0 + 308

capacity is 631017 blocks

----- please choose one (? for help, .. to quit this menu)-----
[ro]otdrive        [u]srrootdrive     [o]ptiondrive      [re]size
```

You see the partition layout for the disk that you specified when *fx* was started, followed by the **repartition** menu. Look at the size column for partitions 0, 1, and 6. In this example, you have 32400 + 81810 + 513945 = 628155 blocks to use. Look at the start block numbers, and notice that partition 7 overlaps 0, 1, and 6. Partition 0 is the Root filesystem, and is mounted on the system's root directory (/). Partition 1 is your system's swap space. Partition 6 is the Usr filesystem, and it is mounted on the */usr* directory. In this example, you will take space from the Usr filesystem and expand the Root filesystem.

2. Choose the **resize** option to change the size of partitions on the disk and answer **`y`** to the warning message:

```
fx/repartition> resize

Warning: you will need to re-install all software and restore user data
from backups after changing the partition layout.  Changing partitions
will cause all data on the drive to be lost.  Be sure you have the drive
backed up if it contains any user data.  Continue? y

After changing the partition, the other partitions will
be adjusted around it to fit the change.  The result will be
displayed and you will be asked whether it is OK, before the
change is committed to disk.  Only the standard partitions may
be changed with this function.  Type ? at prompts for a list
of possible choices
```

3. The prompt after the warning message offers the swap space partition as the default partition to change, but in this example designate the root partition to be resized, so enter **root** at the prompt:

```
fx/repartition/resize: partition to change = (swap) root
current:  type efs       base:     7 cyls,    2835 blks,    1 Mb
              len:    80 cyls,   32400 blks,  16 Mb
```

4. The next prompt asks for the partitioning method (partition size units) with megabytes as the default. Other options are to use percentages of total disk space, numbers of disk blocks, or numbers of disk cylinders. Megabytes and percentages are the easiest methods to use to partition your disk. Press **<Enter>** to use megabytes as the method of repartitioning:

```
fx/repartition/resize: partitioning method = (megabytes (2^20 bytes)) <Enter>
```

5. The next prompt asks for the size of the root partition in megabytes. The default is the current size of the partition. For this example, increase the size to 20 MB:

```
fx/repartition/resize: size in megabytes (max 307) = (16) 20
----- partitions-----
part  type       cyls           blocks          Megabytes   (base+size)
  0: efs        7 + 101        2835 + 40960        1 + 20
  1: rawdata  108 + 180       43795 + 73250       21 + 36
  6: efs      289 + 1269     117045 + 513945      57 + 251
  8: volhdr     0 + 7             0 + 2835         0 + 1
 10: entire     0 + 1558          0 + 630990       0 + 308
```

The new partition map is displayed. Note that the 4 megabytes that you added to your root partition were taken from the swap partition. Ultimately, you want those megabytes to come from the usr partition, but for the moment, accept the new partition layout.

6. To accept the new partition layout, enter **yes** at the prompt:

```
Use the new partition layout? (no) yes
```

The new partition table is printed again, along with the total disk capacity. Then you are returned to the repartition menu.

7. Select **resize** again to transfer space from the usr partition to the swap area:

```
fx/repartition> resize
```

You see the same warning message again.

8. At the partition to change prompt, press **<Enter>** to change the size of the swap partition:

```
fx/repartition/resize: partition to change = (swap) <Enter>
current:  type raw       base:   108 cyls,   43795 blks,   21 Mb
                          len:   180 cyls,   73250 blks,   36 Mb
```

9. Press **<Enter>** again to use megabytes as the method of repartition:

```
fx/repartition/resize: partitioning method = (megabytes (2^20 bytes)) <Enter>
```

10. The next prompt requests the new size of the swap partition. Since you added 4 megabytes to expand the Root filesystem from 16 to 20 megabytes, enter **40** and press **<Enter>** at this prompt to expand the swap space to its original size. (If your system is chronically short of swap space, you can take this opportunity to add some space by entering a higher number.)

```
fx/repartition/resize: size in megabytes (max 307) = (36) 40
----- partitions-----
part  type       cyls             blocks          Megabytes   (base+size)
  0: efs        7 + 101        2835 + 40960         1 + 20
  1: rawdata  108 + 202       43795 + 81920        21 + 40
  6: efs      310 + 1247     125715 + 505275       61 + 247
  8: volhdr     0 + 7             0 + 2835          0 + 1
 10: entire     0 + 1558          0 + 630990        0 + 308
```

You see the new partition table. Note that the partition table now reflects that 4 megabytes have been taken from partition 6 (usr) and placed in the swap partition.

11. At the prompt, enter **yes** to accept the new partition layout:

```
Use the new partition layout? (no) yes
```

The new partition table and the repartition submenu are displayed again.

12. Enter `..` at the prompt to move back to the *fx* main menu:

```
fx/repartition> ..

----- please choose one (? for help, .. to quit this menu)-----
[exi]t              [d]ebug/            [l]abel/
[b]adblock/         [exe]rcise/         [r]epartition/
fx>
```

After Repartitioning

1. From the *fx* main menu, enter **`exit`** to quit *fx*.

   ```
   fx> exit
   ```

2. If you repartitioned the system disk, you must now install software on it in one of two ways:

   - Bring up the miniroot (choose "Install System Software" from the System Maintenance Menu), use the **mkfs** command on the Administrative Commands Menu to make filesystems on the disk partitions, and install an IRIX release and optional software.
   - Choose "System Recovery" from the System Maintenance Menu and use the Backup or System Manager backup tape you created earlier to return the original files to the disk.

3. If you repartitioned an option disk, use the *mkfs* command to create new filesystems on the disk partitions.
4. Restore user files from backup tapes as necessary.

Creating Device Files With *MAKEDEV*

If you need to create device files for a non-SCSI disk or a SCSI disk that is not on an integral SCSI controller, use the *MAKEDEV* command. The *MAKEDEV* command with no arguments creates a standard set of device files in the current directory, so normally it is executed from the */dev* directory. As superuser, give these commands:

```
# cd /dev
# ./MAKEDEV
```

By giving command line arguments, you can create some nonstandard devices with *MAKEDEV*. See the MAKEDEV(1M) reference page for information about creating nonstandard devices using *MAKEDEV*. Another way to create nonstandard devices with *MAKEDEV* is to edit the *MAKEDEV* script, in */dev/MAKEDEV*, or its auxiliary scripts, in */dev/MAKEDEV.d*, add devices, and run *MAKEDEV* as shown above.

Creating Device Files With *mknod*

You may need to create specific device files that are not created by *MAKEDEV*; for example, a device file for a partition that is not created by default. You can edit */dev/MAKEDEV* or files in */dev/MAKEDEV.d* as described in the section "Creating Device Files With MAKEDEV" in this chapter or use the *mknod* command to create a specific device special file in */dev.*

The three forms of the *mknod* command are:

mknod *name* **b** *major minor*

mknod *name* **c** *major minor*

mknod *name* **p**

The arguments of *mknod* are:

name	Specifies the *name* of the special file.
b	Specifies a block device.
c	Specifies a character device.
major	*major* specifies a device type that corresponds to an appropriate entry in the block or character device switch tables.
minor	The *minor* number indicates a unit of the device. It distinguishes peripheral devices from each other.
p	Specifies the special file as a first-in, first-out (FIFO) device. This is also known as a *named pipe*. Named pipes have nothing to do with disks; the use of this option is not described in this guide.

As an example, create a character (raw) device file for partition 3 of a SCSI disk that is on controller 0 at drive address 2 (partition 3 has been created by custom partitioning of the disk with *fx*). The value of *name* would be */dev/rdsk/dks0d2s3*:

/dev/	All device files are in this directory.
rdsk/	The directory for character (raw) device files for disks.
dks	It is a SCSI disk.
0d2s3	Controller 0, drive address 2, partition 3.

To determine the values of *major* and *minor*, start by listing the contents of the device file directory for this disk:

```
# ls -l /dev/rdsk/dks0d2*
crw-------    1 root     sys      128, 32 Nov 30 06:49 dks0d2s0
crw-------    1 root     sys      128, 33 Nov 30 06:49 dks0d2s1
crw-------    1 root     sys      128, 38 Nov 30 06:49 dks0d2s6
crw-------    1 root     sys      128, 39 Nov 30 06:49 dks0d2s7
crw-------    1 root     sys      128, 40 Nov 30 06:49 dks0d2vh
crw-------    1 root     sys      128, 42 Nov 30 06:49 dks0d2vol
```

The major device number for this disk is 128. Looking at the minor numbers, you can see that they are assigned based on the partition number. Partition 3 should be minor number 35.

The command to make a device file for the character device for this partition is:

```
# mknod /dev/rdsk/dks0d2s2 c 128 35
```

Creating Mnemonic Names for Device Files With *ln*

Device file names, for example */dev/dsk/dks0d1s0* and */dev/rdsk/dks0d2s7*, can be difficult to remember and type. *Mnemonic device files* can solve this problem. They are filenames in the */dev* directory that are symbolic links to the real device files. By default, IRIX has several of these mnemonic device file names. For example, */dev/root* is a mnemonic device file name for */dev/dsk/dks0d1s0* (or whatever partition contains the Root filesystem) and */dev/rswap* is a mnemonic device file name for */dev/rdsk/dks0d1s1* (or whatever partition is the swap partition). You can create additional mnemonic device file names using the *ln* command:

```
# ln device_file mnemonic_name
```

For more information on the *ln* command, see the ln(1) reference page.

Creating a System Disk From the PROM Monitor

This section describes how to install a system disk on a system that does not currently have a working system disk. It is used in these situations:

- The new disk has no formatting or partitioning information on it at all or the partitioning is incorrect.

- It is an option disk that you must turn into a system disk.

If the system already has a working disk, you can use the procedure in the section "Creating a New System Disk From IRIX" in this chapter instead.

To turn a disk into a system disk, you must have an IRIX system software release CD available and a CD-ROM drive attached to the system or available on the network. If you are using a CD-ROM drive attached to a system on the network, that system must be set up as an installation server. See the *IRIX Admin: Software Installation and Licensing* guide for instructions.

These instructions assume that the system disk is installed on controller 0 at drive address 1. This is the standard location for workstations; the controller number is system-specific on servers. Follow these steps:

1. Bring the system up into the System Maintenance Menu.
2. Bring up the Command Monitor by choosing the fifth item on the System Maintenance Menu.
3. Give the *hinv* command and use the CPU type and Table 2-1 to determine the version of standalone *fx* that you need to invoke. For example, a system with an IP19 processor is an ARCS processor, so the version of standalone *fx* needed is *stand/fx.ARCS*.
4. Determine the controller and drive address of the device that contains the copy of *fx* that you plan to use (a CD-ROM drive attached to the system or a CD-ROM drive on a workstation on the network). For example, for a local CD-ROM drive, if *hinv* reports that the CD-ROM drive on the system is scsi(0), cdrom(4), the controller is 0 and the drive address is 4. The remainder of this example uses that device, although your device may be different or may be located on a different workstation.
5. If you are installing over a network connection, get the IP address of the workstation with the CD-ROM drive.
6. Insert the CD containing the IRIX system software release into the CD-ROM drive.
7. Give a Command Monitor command to boot *fx*. For this example the command is:

```
>> boot -f dksc(0,4,8)sashARCS dksc(0,4,7)stand/fx.ARCS --x
72912+9440+3024+331696+23768d+3644+5808 entry: 0x89f9a950
112784+28720+19296+2817088+59600d+7076+10944 entry: 0x89cd74d0
SGI Version 5.3 ARCS   Oct 18, 1994
```

See Appendix A of the guide *IRIX Admin: Software Installation and Licensing* for a complete listing of appropriate commands to boot *fx* from CD-ROM on this or another workstation.

8. Respond to the prompts by pressing the **<Enter>** key. These responses select the system disk:

```
fx: "device-name" = (dksc)
fx: ctlr# = (0) <Enter>
fx: drive# = (1) <Enter>
...opening dksc(0,1,)
...controller test...OK
Scsi drive type == SGI     SEAGATE ST31200N8640

----- please choose one (? for help, .. to quit this menu)-----
[exi]t              [d]ebug/            [l]abel/            [a]uto
[b]adblock/         [exe]rcise/         [r]epartition/      [f]ormat
```

9. Display the partitioning of the disk by giving the repartition command:

```
fx> repartition

----- partitions-----
part  type       cyls             blocks          Megabytes   (base+size)
  7: efs        4 + 2722       3048 + 2074164       1 + 1013
  8: volhdr     0 + 4             0 + 3048          0 + 1
 10: volume     0 + 2726          0 + 2077212       0 + 1014

capacity is 2077833 blocks
```

Check the partition layout to see if the disk needs repartitioning. See the section "System Disks, Option Disks, and Partition Layouts" in Chapter 1 for information about standard partition layouts.

10. If the disk doesn't need repartitioning, skip to step 13.
11. Choose a disk partition layout. You can choose a standard system disk partition layout (described in the section "System Disks, Option Disks, and Partition Layouts" in Chapter 1) or a custom partition layout.
12. If you choose a standard system disk partition layout, follow the directions in the section "Creating Standard Partition Layouts" in this chapter. If you choose a custom partition layout, follow the instructions in the section "Creating Custom Partition Layouts" in this chapter.

13. In preparation for a future step, check the contents of the volume header by giving this command:

```
----- please choose one (? for help, .. to quit this menu)-----
[ro]otdrive           [o]ptiondrive         [e]xpert
[u]srrootdrive        [re]size
fx/repartition> label/show/directory

 0: sgilabel   block    3 size     512  2: sash       block 1914 size  159232
 1: ide        block    4 size  977920
```

    Verify that the volume header contains *sash*, a required file (it is listed as item 2 in this example).

14. Quit *fx* and the Command Monitor so that you return to the System Maintenance Menu:

```
----- please choose one (? for help, .. to quit this menu)-----
[para]meters       [part]itions       [b]ootinfo         [a]ll
[g]eometry         [s]giinfo          [d]irectory
fx/label/show> ../../exit
>> exit
```

15. Choose the second option, “Install System Software,” from the System Maintenance Menu.

    Because there is no filesystem on the root partition, error messages may appear. One example is the following message:

```
Mounting file systems:

/dev/dsk/dks0d1s0: Invalid argument
No valid file system found on: /dev/dsk/dks0d1s0
This is your system disk: without it we have nothing
on which to install software.
```

    Another possible message indicates a problem, but does mount the root partition and bring up *inst*:

```
Mounting file systems:

mount: /root/dev/usr on /root/usr: No such file or directory
mount: giving up on:
   /root/usr

Unable to mount all local efs, xfs file systems under /root
Copy of above errors left in /root/etc/fscklogs/miniroot
```

```
    /dev/miniroot            on  /
    /dev/dsk/dks0d1s0        on  /root

Invoking software installation.
```

16. If the system offers to make a filesystem, answer **`yes`** to the prompts:

```
Make new file system on /dev/dsk/dks0d1s0 [yes/no/sh/help]: yes

About to remake (mkfs) file system on: /dev/dsk/dks0d1s0
This will destroy all data on disk partition: /dev/dsk/dks0d1s0.

        Are you sure? [y/n] (n): yes

        Do you want an EFS or an XFS filesystem? [efs/xfs]: xfs

        Block size of filesystem 512 or 4096 bytes? 4096

Doing: mkfs -b size=512 /dev/dsk/dks0d1s0
meta-data=/dev/rdsk/dks0d1s0     isize=256    agcount=8, agsize=248166 blks
data     =                       bsize=4096   blocks=248165
log      =internal log           bsize=512    blocks=1000
realtime =none                   bsize=4096   blocks=0, rtextents=0
Mounting file systems:

NOTICE: Start mounting filesystem: /root
NOTICE: Ending clean XFS mount for filesystem: /root
    /dev/miniroot            on  /
    /dev/dsk/dks0d1s0        on  /root
```

17. If the system offers to put you into a shell, go into the shell and manually make the Root and, if appropriate, the Usr filesystem. For example:

```
Please manually correct your configuration and try again.

        Press Enter to invoke C Shell csh: <Enter>

# mkfs /dev/dsk/dks0d1s0
meta-data=/dev/dsk/dks0d1s0      isize=256    agcount=8, agsize=31021 blks
data     =                       bsize=4096   blocks=248165
log      =internal log           bsize=4096   blocks=1000
realtime =none                   bsize=4096   blocks=0, rtextents=0
# exit
```

18. If the *inst* main menu comes up and you did not make a Root filesystem in step 16 or step 17, make the Root and, if used, the Usr filesystems, and mount them. For example:

```
Inst> admin
...
Admin> mkfs /dev/dsk/dks0d1s0

Make new file system on /dev/dsk/dks0d1s0 [yes/no/sh/help]: yes

About to remake (mkfs) file system on: /dev/dsk/dks0d1s0
This will destroy all data on disk partition: /dev/dsk/dks0d1s0.

        Are you sure? [y/n] (n): yes

        Do you want an EFS or an XFS filesystem? [efs/xfs]: xfs

        Block size of filesystem 512 or 4096 bytes? 4096

Doing: mkfs -b size=512 /dev/dsk/dks0d1s0
meta-data=/dev/rdsk/dks0d1s0     isize=256    agcount=8, agsize=248166 blks
data     =                       bsize=4096   blocks=248165
log      =internal log           bsize=512    blocks=1000
realtime =none                   bsize=4096   blocks=0, rtextents=0
Mounting file systems:

NOTICE: Start mounting filesystem: /root
NOTICE: Ending clean XFS mount for filesystem: /root
    /dev/miniroot            on  /
    /dev/dsk/dks0d1s0        on  /root


Re-initializing installation history database
Reading installation history .. 100% Done.
Checking dependencies .. 100% Done.

Admin> return
```

19. Install IRIX software from the CD as usual.
20. Install option software and patches from other CDs, if desired.
21. If you don't need to modify the volume header to add *sash* (see step 13), you have finished creating the new system disk. You don't need to do the remaining steps in this procedure.

22. In preparation for adding programs to the volume header of the disk, start a shell:
    ```
    Inst> sh
    ```
23. Follow the instructions in the procedure in the section "Adding Files to the Volume Header With dvhtool" in this chapter to add *sash*, if necessary, to the volume header of the system disk. Remember that the */stand* directory is mounted at */root/stand*.
24. Exit from the shell:
    ```
    # exit
    ```
25. Quit *inst* and bring the system up as usual.
    ```
    Inst> quit
    ```

Creating a New System Disk From IRIX

This procedure describes how to turn an option disk into a system disk. The option disk doesn't need to have a filesystem or be mounted prior to starting the procedure.

Caution: The procedure in this section destroys all data on the option disk. If the option disk contains files that you want to save, back up all files on the option disk to tape or another disk before beginning this procedure.

You can use this procedure when you want to change to a larger system disk, for example from a 1 GB disk to a 2 GB disk, or when you want to create a system disk that you can move to another system. With this procedure, you create a "fresh" disk by installing software from an IRIX system software CD. (To create an exact copy of a system disk, use the section "Creating a New System Disk by Cloning" in this chapter instead.) Note that if you plan to create a system disk for another system, the systems must be identical because of hardware dependencies in IRIX.

You must perform this procedure as superuser. The procedure requires several system reboots, so other users shouldn't be using the system.

1. Using *hinv*, determine the controller and drive addresses of the disk to be turned into a system disk. In this procedure, the example commands and output assume that the disk is on controller 0 and drive address 2. Substitute your controller and drive address throughout these instructions.
2. To repartition the disk so that it can be used as a system disk, begin by invoking *fx*:
    ```
    # fx
    fx version 5.3, Dec 19, 1994
    ```

3. Answer the prompts with the correct controller number and drive address for the disk you are converting and 0 for the lun number, for example:

```
fx: "device-name" = (dksc) <Enter>
fx: ctlr# = (0) <Enter>
fx: drive# = (1) 2
fx: lun# = (0) <Enter>
...opening dksc(0,2,0)
...controller test...OK
Scsi drive type == SGI     SEAGATE ST31200N8640

----- please choose one (? for help, .. to quit this menu)-----
[exi]t               [d]ebug/             [l]abel/
[b]adblock/          [exe]rcise/          [r]epartition/
```

4. Choose the **repartition** command:

```
fx> repartition

----- partitions-----
part  type       cyls             blocks          Megabytes   (base+size)
  7: efs        4 + 2722       3024 + 2057832       1 + 1005
  8: volhdr     0 + 4             0 + 3024          0 + 1
 10: volume     0 + 2726          0 + 2060856       0 + 1006

capacity is 2061108 blocks
```

5. Choose **rootdrive** or **usrrootdrive**, depending upon whether you want a combined root and usr partition or separate root and usr partitions. (See the section "System Disks, Option Disks, and Partition Layouts" in Chapter 1 for advantages and disadvantages of each.) In this example, a combined root and usr disk, configured for XFS, is chosen:

```
----- please choose one (? for help, .. to quit this menu)-----
[ro]otdrive        [u]srrootdrive     [o]ptiondrive      [re]size

fx/repartition> rootdrive

fx/repartition/rootdrive: type of data partition = (xfs) <Enter>

----- partitions-----
part  type       cyls             blocks          Megabytes   (base+size)
  0: xfs        4 + 2614       3024 + 1976184       1 + 965
  1: raw     2618 + 108     1979208 + 81648       966 + 40
  8: volhdr     0 + 4             0 + 3024          0 + 1
 10: volume     0 + 2726          0 + 2060856       0 + 1006
```

```
capacity is 2061108 blocks
```

6. Quit *fx*:

```
----- please choose one (? for help, .. to quit this menu)-----
[ro]otdrive        [u]srrootdrive     [o]ptiondrive      [re]size
fx/repartition> ../exit
```

7. Use the procedure in the section “Adding Files to the Volume Header With dvhtool” in this chapter to examine the contents of the volume header of the disk to be converted and to add *sash* to its volume header if it isn’t there already.

8. Make a Root filesystem on the root partition of the disk you are converting. If the disk has a separate Usr partition, make a filesystem on that partition, too. For example, to make an XFS filesystem with 4 KB block size and a 1000 block internal log (the default values), give this command:

   ```
   # mkfs /dev/dsk/dks0d2s0
   ```

   As another example, to make an EFS filesystem, give this command:

   ```
   # mkfs -t efs /dev/rdsk/dks0d2s0
   ```

   For additional instructions on making an XFS filesystem, see the sections “Planning for XFS Filesystems” and “Making an XFS Filesystem” in Chapter 4. For additional instructions on making an EFS filesystem, see the section “Making an EFS Filesystem” in Chapter 4. There is no need to mount the filesystems after making them.

9. Insert a CD containing the IRIX release you plan to install into either your system’s CD-ROM drive or a CD-ROM drive on a remote system.

10. Shut down the system and bring up the miniroot from the CD. For instructions, see the guide *IRIX Admin: Software Installation and Licensing*.

11. Switch to the Administrative Commands Menu, unmount the root and usr (if used) partitions from the old system disk, and mount the root and usr (if used) partitions of the new disk in their place. For example, if the old system disk has root and usr partitions and the new system disk has only a root partition, the commands are:

    ```
    Inst> admin
    Admin> umount /root
    Admin> umount /root/usr
    Admin> mount /dev/dsk/dks0d2s0 /root
    Admin> return
    ```

12. Confirm that the root and usr (if used) partitions of the new system disk are mounted as */root* and */root/usr* (if used). This example shows the output for the example in step 11:

    ```
    Inst> sh df

    Filesystem              Type  blocks      use      avail  %use Mounted
    on
    /dev/miniroot            xfs    49000     32812     16188  67  /
    /dev/dsk/dks0d1s0        xfs  1984325       251   1984074   0  /root
    ```

    Caution: If the wrong partitions are mounted, *inst* installs system software onto the wrong partitions, which destroys the data on those partitions.

13. Install system software from this CD and options and patches from other CDs as usual. Instructions are in the guide *IRIX Admin: Software Installation and Licensing*.
14. Quit *inst* and bring the system back to IRIX (the system boots the old system disk).
15. To test the new system disk before replacing the old system disk or moving the disk to a different system, begin by shutting down the system to the PROM Monitor.
16. Bring up the Command Monitor by choosing the fifth item on the System Maintenance Menu.
17. Boot the system in single user mode from the new system disk by giving the commands below. It uses controller 0 and drive address 2; substitute the values for the new system disk in the first and second positions of each of the three triples of numbers in this example.

    ```
    >> setenv initstate=s
    >> boot -f dksc(0,2,8)sash dksc(0,2,0)unix root=dks0d2s0
    ```

18. Run *MAKEDEV* and *autoconfig*:

    ```
    # cd /dev
    # ./MAKEDEV
    # /etc/autoconfig -f
    ```

19. Restart the system in multiuser mode by choosing Restart System from the System menu of the Toolchest or with the *reboot* command.

The new system disk is ready to replace the system disk on this system or another system with the same hardware configuration.

Creating a New System Disk by Cloning

This procedure describes how to turn an option disk into an exact copy of a system disk. Use this procedure when you want to set up two or more systems with identical system disks. The systems must have identical processor and graphics types.

Caution: The procedure in this section destroys all data on the option disk. If the option disk contains files that you want to save, back up all files on the option disk to tape or another disk before beginning this procedure.

You must perform this procedure as superuser. To ensure that the system disk that you create is identical to the original system disk, the system should be in single user mode.

1. List the disk partitioning of the system disk, for example:

```
# prtvtoc /dev/rdsk/dks0d1vh
...
Partition  Type  Fs   Start: sec   (cyl)    Size: sec    (cyl)  Mount Directory
 0          efs  yes        3048  (   4)        51054   (  67)   /
 1          raw            54102  (  71)        81534   ( 107)
 6          efs  yes      135636  ( 178)      1941576   (2548)   /usr
 8       volhdr                0  (   0)         3048   (   4)
10       volume                0  (   0)      2077212   (2726)
```

2. List the disk partitioning of the option disk that is to be the clone, for example:

```
# prtvtoc /dev/rdsk/dks0d2vh
...
Partition  Type  Fs   Start: sec   (cyl)    Size: sec    (cyl)  Mount Directory
 0          efs             3024  (   4)        50652   (  67)
 1          raw            53676  (  71)        81648   ( 108)
 6          efs           135324  ( 179)      1925532   (2547)
 8       volhdr                0  (   0)         3024   (   4)
10       volume                0  (   0)      2060856   (2726)
```

3. Compare the disk partitioning of the two disks. They must have the same layout for the root and (if used) the usr partition. If they are not the same, repartition the option disk to match the system disk using the procedure in the section "Repartitioning a Disk With fx" in this chapter.

4. Use the procedure in the section "Adding Files to the Volume Header With dvhtool" in this chapter to check the contents of the volume header of the option disk and add programs, if necessary, by copying them from the system disk.

5. Make a new filesystem on the root partition of the option disk. For example, to make an XFS filesystem with a 4 KB block size and a 1000 block internal log (the default values), give this command:

   ```
   # mkfs /dev/dsk/dks0d2s0
   ```

   As another example, to make an EFS filesystem, give this command:

   ```
   # mkfs –t efs /dev/rdsk/dks0d2s0
   ```

   For additional instructions on making an XFS filesystem, see the sections "Making an XFS Filesystem" and "Making an EFS Filesystem" in Chapter 4. For additional instructions on making an EFS filesystem, see the section "Making an EFS Filesystem" in Chapter 4. There is no need to mount the filesystems after making them.

6. If there is a separate usr partition, make a new filesystem on the usr partition of the option disk.

7. Create a temporary mount point for the option disk filesystems, for example:

   ```
   # mkdir /clone
   ```

8. Mount the Root filesystem of the option disk and change directories to the mount point, for example:

   ```
   # mount /dev/dsk/dks0d2s0 /clone
   # cd /clone
   ```

9. Use *dump* (for EFS filesystems) or *xfsdump* (for XFS filesystems) to copy the Root filesystem on the system disk to the Root filesystem of the option disk. The *dump* command is:

   ```
   # dump 0f – / | restore xf –
   ```

   The *xfsdump* command is:

   ```
   # xfsdump –l 0 – / | xfsrestore – .
   ```

10. If the disks do not have a usr partition, skip to step 13.

11. In preparation for copying the Usr filesystem, mount the Usr filesystem instead of the Root filesystem:

   ```
   # cd ..
   # umount /clone
   # mount /dev/dsk/dks0d2s6 /clone
   # cd /clone
   ```

12. Use *dump* (for EFS filesystems) or *xfsdump* (for XFS filesystems) to copy the Usr filesystem on the system disk to the Usr filesystem of the option disk. The *dump* command is:

    ```
    # dump 0f - /usr | restore xf -
    ```

    The *xfsdump* command is:

    ```
    # xfsdump -l 0 - /usr | xfsrestore - .
    ```

13. Unmount the filesystem mounted at the temporary mount point and remove the mount point, for example:

    ```
    # cd ..
    # umount /clone
    # rmdir /clone
    ```

    The option disk is now an exact copy of the system disk. It can be moved to a system with the same hardware configuration.

Adding a New Option Disk

Adding a new option disk to a system involves the general steps below. Each step contains one or more references to the manual or section in this guide that contains specific instructions for the step.

1. Install the hardware. See the *Owner's Guide* for the system for information.
2. Initialize the volume header, if necessary. See the section "Formatting and Initializing a Disk With fx" in this chapter.
3. Partition the new disk, if necessary. It should be partitioned as an option disk. See the section "Repartitioning a Disk With fx" in this chapter for instructions.
4. In preparation for the next step, identify the type of controller that the new disk is attached to (integral SCSI controller, non-integral SCSI controller, or non-integral VME controller). See the section "Listing the Disks on a System With hinv" in this chapter for instructions.
5. Create device files, if necessary, in the */dev* directory on the system disk and make one or more filesystems on the disk. For disks on integral SCSI controllers, use the procedure in the next subsection, "Adding a Disk on an Integral SCSI Controller." For a disk on a non-integral SCSI or VME controller, use the procedure in the subsection called "Adding a Disk on a Non-Integral SCSI Controller or a VME Controller" instead.

Tip: You can use the Disk Manager in the System Toolchest to add a new option disk. For instructions, see the section “Setting Up a New Disk” in Chapter 6 of the *Personal System Administration Guide*. The section “Taking Advantage of a Second Disk” in that chapter provides ideas for making effective use of an option disk.

Adding a Disk on an Integral SCSI Controller

To add an option disk on an integral SCSI controller to a system, perform these steps:

1. Complete the steps in the section “Adding a New Option Disk” above.
2. Use the *Add_disk* command to perform the remaining steps to configure the disk:

   ```
   # Add_disk controller_number  drive_address lun_number
   ```

   If you are adding a second disk on controller 0 to your system, you do not have to specify the disk, controller number, or logical unit number; adding disk 2 on controller 0 is the default. If you are adding a third (or greater) disk, or if you are adding a disk on a controller other than controller 0, you must specify the disk and controller. If the disk device has a logical unit number different from zero, it must be specified.

   Add_disk checks for valid filesystems on the disk, and if any filesystems are present, you are warned and asked for permission before the existing filesystems are destroyed and a new filesystem is made.

   The *Add_disk* command performs these tasks:

   - Creates the character and raw device files for the new disk
   - Creates a filesystem on the disk
   - Creates the mount directory
   - Mounts the filesystem
   - Adds the mount order to the */etc/fstab* file

Adding a Disk on a Non-Integral SCSI Controller or a VME Controller

To add an option disk on a non-integral SCSI controller to a system or to add an option disk on a VME bus SCSI controller to a system, perform these steps:

1. Complete the steps in the section "Adding a New Option Disk" above.
2. Create the device files, if necessary. See the sections "Creating Device Files With MAKEDEV" and "Creating Device Files With mknod" in this chapter.
3. Make a filesystem. Use the instructions in one of these sections in Chapter 4: "Making an XFS Filesystem" or "Making an EFS Filesystem."