Chapter 4

Creating and Growing Filesystems

This chapter describes the procedures you must perform to create or grow (increase the size of) XFS and EFS filesystems or to convert from an EFS filesystem to an XFS filesystem.

The major sections in this chapter are:

Planning for XFS Filesystems

The following subsections discuss preparation for and choices you must make when creating an XFS filesystem. Each time you plan to make an XFS filesystem or convert a filesystem from EFS to XFS, review each section and make any necessary preparations.

Prerequisite Software

The subsystem *eoe.sw.xfs* is required to use XFS filesystems.

If you are converting the Root and Usr filesystems to XFS, you must have software distribution CDs or access to a remote distribution directory for IRIX Release 6.2 or later.

Choosing the Filesystem Block Size and Extent Size

XFS allows you to choose the logical block size for each filesystem. (Physical disk blocks remain 512 bytes. EFS has a fixed block size of 512 bytes.) If you use a real-time subvolume on an XLV logical volume, you must also choose the extent size. The extent size is the amount of space that is allocated to a file every time more space needs to be allocated to it.

For XFS filesystems on disk partitions and logical volumes and for the data subvolume of filesystems on XLV volumes, the block size guidelines are:

- The minimum block size is 512 bytes. Small block sizes increase allocation overhead which decreases filesystem performance, but in general, the recommended block size for filesystems under 100 MB and for filesystems with many small files is 512 bytes.
- The default block size is 4096 bytes (4K). This is the recommended block size for filesystems over 100 MB.
- The maximum block size is 65536 bytes (64K). Because large block sizes can waste space and lead to fragmentation, in general block sizes shouldn't be larger than 4096 bytes (4K).
- For the Root filesystem on systems with separate Root and Usr filesystems, the recommended block size is 512 bytes.
- For news servers, the recommended block size for the news filesystems is 2048 bytes.

Block sizes are specified in bytes in decimal (default), octal (prefixed by 0), or hexadecimal (prefixed by 0x or 0X). If the number has the suffix "k," it is multiplied by 1024. If the number has the suffix "m," it is multiplied by 1048576 (1024 * 1024).

For real-time subvolumes of XLV logical volumes, the block size is the same as the block size of the data subvolume. The guidelines for the extent size are:

- The extent size must be a multiple of the block size of the data subvolume.
- The minimum extent size is 4 KB.
- The maximum extent size is 1 GB.
- The default extent size is 64 KB.
- The extent size should be matched to the application and the stripe unit of the volume elements used in the real-time subvolume.

Choosing the Log Type and Size

Each XFS filesystem has a log that contains filesystem journaling records. This log requires dedicated disk space. This disk space doesn't show up in listings from the *df* command, nor can you access it with a filename.

The location of the disk space depends on the type of log you choose. The two types of logs are:

External — When an XFS filesystem is created on an XLV logical volume and log records are put into a log subvolume, the log is called an *external* log. The log subvolume is one or more disk partitions dedicated to the log exclusively.

Internal — When an XFS filesystem is created on a disk partition or XLV logical volume, or when it is created on an XLV logical volume that doesn't have a log subvolume, log records are put into a dedicated portion of the disk partition (or data subvolume) that contains user files. This type of log is called an *internal* log.

The guidelines for choosing the log type are:

- If you want the log and the data subvolume to be on different partitions or to use different subvolume configurations for them, use an external log.
- If you want the log subvolume to be *striped* independently from the data subvolume (see the section "Volume Elements" in Chapter 6 for an explanation of striping), you must use an external log.
- If you are making the XFS filesystem on a disk partition (rather than on an XLV logical volume), you must use an internal log.
- If you are making the XFS filesystem on an XLV logical volume that has no log subvolume, you must use an internal log.
- If you are making the XFS filesystem on an XLV logical volume that has a log subvolume, you must use an external log.

For more information about XLV and log subvolumes, see the section "XLV Logical Volumes" in Chapter 6.

The amount of disk space needed for the log is a function of how the filesystem is used. The amount of disk space required for log records is proportional to the transaction rate and the size of transactions on the filesystem, not the size of the filesystem. Larger block sizes result in larger transactions. Transactions from directory updates (for example, the *mkdir* and *rmdir* commands and the **create()** and **unlink()** system calls) cause more log data to be generated.

You must choose the amount of disk space to dedicate to the log (called the log size). The minimum log size is 512 blocks. The typical log size is 1000 blocks. For filesystems with very high transaction activity, a larger log size of 2000 blocks is recommended.

For external logs, the size of the log is the same as the size of the log subvolume. The log subvolume is one or more disk partitions. You may find that you need to repartition a disk to create a properly sized log subvolume (see the section "Disk Repartitioning" in this chapter). For external logs, the size of the log is set when you create the log subvolume with the *xlv_make* command.

For internal logs, the size of the log is specified when you create the filesystem with the *mkfs* command.

The log size is specified in bytes as described in the section "Choosing the Filesystem Block Size and Extent Size" in this chapter or as a multiple of the filesystem block size by using the suffix "b."

Checking for Adequate Free Disk Space

XFS filesystems may require more disk space than EFS filesystems for the same files. This extra disk space is required to accommodate the XFS log and as a result of block sizes larger than EFS's 512 bytes. However, XFS represents free space more compactly, on average, and the inodes are allocated dynamically by XFS, which can result in less disk space usage.

This procedure can be used to get a rough idea of the amount of free disk space that will remain after a filesystem is converted to XFS:

1. Get the size in kilobytes of the filesystem to be converted and round the result to the next megabyte. For example,

   ```
   df -k
   Filesystem           Type  kbytes     use   avail %use  Mounted on
   /dev/root             efs  969857  663306  306551  68%  /
   ```

   This filesystem is 969857 KB, which rounds up to 970 MB.

2. If you plan to use an internal log (see the section "Choosing the Log Type and Size" in this chapter), give this command to get an estimate of the disk space required for the files in the filesystem after conversion:

   % **xfs_estimate -i** *logsize* **-b** *blocksize mountpoint*

   logsize is the size of the log. *blocksize* is the block size you chose for user files in the section "Choosing the Filesystem Block Size and Extent Size" in this chapter. *mountpoint* is the directory that is the mount point for the filesystem. For example,

   ```
   % xfs_estimate -i 1m -b 4096 /
   / will take about 747 megabytes
   ```

   The output of this command tells you how much disk space the files in the filesystem (with a *blocksize* of 4096 bytes) and an internal log of size *logsize* will take after conversion to XFS.

3. If you plan to use an external log, give this command to get an estimate of the disk space required for the files in the filesystem after conversion:

   % **xfs_estimate -e 0 -b** *blocksize mountpoint*

   blocksize is the block size you chose for user files in the section "Choosing the Filesystem Block Size and Extent Size" in this chapter. *mountpoint* is the directory that is the mount point for the filesystem. For example,

   ```
   % xfs_estimate -e 0 -b 4096 /
   / will take about 746 megabytes
           with the external log using 0 blocks or about 1 megabytes
   ```

   The first line of output from *xfs_estimate* tells you how much disk space the files in the filesystem will take after conversion to XFS. In addition to this, you will need disk space on a different disk partition for the external log. You should ignore the second line of output.

4. Compare the size of the filesystem from step 1 with the size of the files from step 2 or step 3. For example,

```
970 MB - 747 MB = 223 MB free disk space
747 MB / 970 MB = 77% full
```

   Use this information to decide if there will be an adequate amount of free disk space if this filesystem is converted to XFS.

If the amount of free disk space after conversion is not adequate, some options to consider are:

- Implement the usual solutions for inadequate disk space: remove unnecessary files, archive files to tape, move files to another filesystem, add another disk, and so on.
- Repartition the disk to increase size of the disk partition for the filesystem.
- If there isn't sufficient disk space in the Root filesystem and you have separate Root and Usr filesystems, switch to combined Root and Usr filesystems on a single disk partition.
- If the filesystem is on an *lv* logical volume or an XLV logical volume, increase the size of the volume.
- Create an XLV logical volume with a log subvolume elsewhere, so that all of the disk space can be used for user files.

Disk Repartitioning

Many system administrators may find that they want or need to repartition disks when they switch to XFS filesystems and/or XLV logical volumes. Some of the reasons to consider repartitioning are:

- If the system disk has separate partitions for Root and Usr filesystems, the Root filesystem may be running out of space. Repartitioning is a way to increase the space in Root (at the expense of the size of Usr) or to solve the problem by combining Root and Usr into a single partition.
- System administration is a little easier on systems with combined Root and Usr filesystems.
- If you plan to use XLV logical volumes, you may want to put the XFS log into a small subvolume. This requires disk repartitioning to create a small partition for the log subvolume.

- If you plan to use XLV logical volumes, you may want to repartition to create disk partitions of equal size that can be striped or plexed.

Disk partitions are discussed in Chapter 1, "Disk Concepts," and using *fx* to repartition disks is explained in the section "Repartitioning a Disk With fx" in Chapter 2.

Dump and Restore Requirements

The filesystem conversion procedures in the sections "Converting Filesystems on the System Disk From EFS to XFS" and "Converting a Filesystem on an Option Disk From EFS to XFS" in this chapter require that you dump the filesystems you plan to convert to tape or to another disk with sufficient free disk space to contain the dump image. Dumping to disk is substantially faster than dumping to tape.

When you convert a system disk, you must use the *dump* and *restore* commands. When you convert a filesystem on an option disk, you can use any backup and restore commands.

If you dump to a tape drive, follow these guidelines:

- Have sufficient tapes available for dumping the filesystems to be converted.
- If you are converting filesystems on a system disk, the tape drive must be local.
- If you are converting filesystems on option disks, the tape drive can be local or remote.

The requirements for dumping to a different filesystem are:

- The filesystem being converted must have 2 GB or less in use (the maximum size of the dump image file on an EFS filesystem) unless it is being dumped to an XFS filesystem.
- The filesystem that will contain the dump must have sufficient disk space available to hold the filesystems to be converted.
- If you are converting filesystems on a system disk, the filesystem where you place the dump must be local to the system.
- If you are converting filesystems on option disks, the filesystem you dump to can be local or remote.

Making an XFS Filesystem

This section explains how to create an XFS filesystem on an empty disk partition or XLV logical volume. (For information about creating XLV logical volumes, see Chapter 7, "Creating and Administering XLV Logical Volumes.")

Tip: You can make an XFS filesystem on a disk partition or a logical volume using the graphical user interface of the *xfsm* command. For information, see its online help.

Caution: When you create a filesystem, all files already on the disk partition or logical volume are destroyed.

1. Review the subsections within the section "Planning for XFS Filesystems" in this chapter to verify that you are ready to begin this procedure.
2. Identify the device name of the partition or logical volume where you plan to create the filesystem. This is the value of *partition* in the examples below. For example, if you plan to use partition 7 (the entire disk) of a SCSI option disk on controller 0 and drive address 2, *partition* is */dev/dsk/dks0d2s7*. For more information on determining *partition*, see Table 1-4, the section "Introduction to Logical Volumes" in Chapter 6, and the dks(7M) reference page.
3. If the disk partition is already mounted, unmount it:

   ```
   # umount partition
   ```

   Any data that is on the disk partition is destroyed (to convert the data rather than destroy it, use the procedure in the section "Converting a Filesystem on an Option Disk From EFS to XFS" in this chapter instead).
4. If you are making a filesystem on a disk partition or on an XLV logical volume that doesn't have a log subvolume, use this *mkfs* command to create the new XFS filesystem:

   ```
   # mkfs -b size=blocksize -l size=logsize partition
   ```

   blocksize is the filesystem block size (see the section "Choosing the Filesystem Block Size and Extent Size" in this chapter) and *logsize* is the size of the area dedicated to log records (see the section "Choosing the Log Type and Size" in this chapter). The default values are 4 KB blocks and a 1000 block log.

   Example 4-1 shows the command line used to create an XFS filesystem and the system output. The filesystem has a 10 MB internal log and a block size of 1K bytes and is on the partition */dev/dsk/dks0d2s7*.

Example 4-1 *mkfs* Command for an XFS Filesystem With an Internal Log

```
# mkfs -b size=1k -l size=10m /dev/dsk/dks0d2s7
meta-data=/dev/dsk/dks0d2s7      isize=256    agcount=8, agsize=128615 blks
data     =                       bsize=1024   blocks=1028916
log      =internal log           bsize=1024   blocks=10240
realtime =none                   bsize=65536  blocks=0, rtextents=0
```

5. If you are making a filesystem on an XLV logical volume that has a log subvolume (for an external log), use this *mkfs* command to make the new XFS filesystem:

   `# mkfs -b size=`*blocksize volume*

   blocksize is the block size for filesystem (see the section "Choosing the Filesystem Block Size and Extent Size" in this chapter), and *volume* is the device name for the volume.

   Example 4-2 shows the command line used to create an XFS filesystem on a logical volume */dev/dsk/xlv/a* with a block size of 1K bytes and the system output.

Example 4-2 *mkfs* Command for an XFS Filesystem With an External Log

```
# mkfs -b size=1k /dev/dsk/xlv/a
meta-data=/dev/dsk/xlv/a         isize=256    agcount=8, agsize=245530 blks
data     =                       bsize=1024   blocks=1964240
log      =volume log             bsize=1024   blocks=25326
realtime =none                   bsize=65536  blocks=0, rtextents=0
```

Example 4-3 shows the command line used to create an XFS filesystem on a logical volume */dev/dsk/xlv/xlv_data1* that includes a log, data, and real-time subvolumes and the system output. The default block size of 4096 bytes is used and the real-time extent size is set to 128K bytes.

Example 4-3 *mkfs* Command for an XFS Filesystem With a Real-Time Subvolume

```
# mkfs_xfs -r extsize=128k /dev/rdsk/xlv/xlv_data1
meta-data=/dev/rdsk/xlv/xlv_data1 isize=256    agcount=8, agsize=4300 blks
data     =                       bsize=4096   blocks=34400
log      =volume log             bsize=4096   blocks=34400
realtime =volume rt              bsize=131072 blocks=2560, rtextents=80
```

6. To use the filesystem, you must mount it. For example:

   ```
   # mkdir mountdir
   # mount partition mountdir
   ```

For more information about mounting filesystems, see the section "Manually Mounting Filesystems" in Chapter 5.

7. To configure the system so that the new filesystem is automatically mounted when the system is booted, add this line to the file */etc/fstab*:

   partition mountdir **`xfs rw,raw=`***rawpartition* **`0 0`**

   where *rawpartition* is the raw version of *partition*. For example, if *partition* is */dev/dsk/dks0d2s7*, *rawpartition* is */dev/rdsk/dks0d2s7*.

   For more information about automatically mounting filesystems, see the section "Mounting Filesystems Automatically With the /etc/fstab File" in Chapter 5.

Making an EFS Filesystem

The procedure in this section explains how to make a filesystem on a disk partition or on a logical volume and mount it. (See From or Chapter 8, "Creating and Administering lv Logical Volumes," for information on creating logical volumes.) This procedure assumes that the disk or logical volume is empty. If it contains valuable data, the data must be backed up because it is destroyed during this procedure.

Tip: You can make an EFS filesystem on a disk partition using the Disk Manager in the System Toolchest. For information, see the section "Formatting, Verifying, and Remaking Filesystems on a Fixed Disk" in Chapter 6 of the *Personal System Administration Guide*.

Caution: When you create a filesystem, all files already on the disk partition or logical volume are destroyed.

1. Identify the device name of the partition or logical volume where you plan to create the filesystem. This is the value of *partition* in the examples below. For example, if you plan to use partition 7 (the entire disk) of a SCSI option disk on controller 0 and drive address 2, *partition* is */dev/dsk/dks0d2s7*. For more information on determining *partition*, see Table 1-4, the section "Introduction to Logical Volumes" in Chapter 6, and the dks(7M) reference page.

2. If the disk partition is already mounted, unmount it:

   `#` **`umount`** *partition*

   Any data that is on the disk partition is destroyed (to convert the data rather than destroy it, use the procedure in the section "Converting a Filesystem on an Option Disk From EFS to XFS" in this chapter instead).

3. Create a new filesystem with the *mkfs* command, for example,

   ```
   # mkfs -t efs /dev/rdsk/dks0d2s7
   ```

   The argument to *mkfs* is the block or character device for the disk partition or logical volume. You can use either the block device or the character device.

   In the above example, *mkfs* uses default values for the filesystem parameters. If you want to use parameters other than the default, you can specify these on the *mkfs* command line. See the mkfs_efs(1M) reference page for information about using command line parameters and proto files.

4. To use the filesystem, you must mount it. For example,

   ```
   # mkdir /rsrch
   # mount /dev/dsk/dks0d2s7 /rsrch
   ```

   For more information about mounting filesystems, see the section "Manually Mounting Filesystems" in Chapter 5.

5. To configure the system so that this filesystem is automatically mounted when the system is booted up, add an entry in the file */etc/fstab* for the new filesystem. For example,

   ```
   /dev/dsk/dks0d2s7 /rsrch efs rw,raw=/dev/rdsk/dks0d2s7 0 0
   ```

   For more information about automatically mounting filesystems, see the section "Mounting Filesystems Automatically With the /etc/fstab File" in Chapter 5.

Making a Filesystem From *inst*

Caution: When you create a filesystem, all files already on the disk partition or logical volume are destroyed.

mkfs can be used from within the *inst* command to make filesystems. To make the Root or Usr filesystem on a system disk, you must use *inst* from the miniroot. There are two ways to use *mkfs*:

- The **mkfs** command on the Administrative Command Menu. The **mkfs** command uses default values for the *mkfs* command options. It chooses an EFS filesystem or an XFS filesystem based on the answer to a prompt. With no argument, the **mkfs** command makes the Root filesystem and, if a usr partition is present, a Usr filesystem. Other filesystems can be made by giving a device file argument to **mkfs**.

- From a shell. Giving the *mkfs* command from a shell (give the command **sh**, not **shroot**) enables you to specify the *mkfs* command line, including options.

For more information about making filesystems from *inst*, see the guide *IRIX Admin: Software Installation and Licensing*.

Growing an XFS Filesystem Onto Another Disk

When growing an XFS filesystem onto another disk, there are two possibilities:

- The XFS filesystem is on a disk partition.
- The XFS filesystem is on an XLV logical volume.

If the XFS filesystem is on an XLV logical volume, the additional disk can be added to the logical volume as an additional volume element. Instructions for doing this are in the section "Adding a Volume Element to a Plex (Growing a Logical Volume)" in Chapter 7.

The following steps show how to grow a fictional */disk2* XFS filesystem onto an XLV logical volume created out of the */disk2* disk partition and a new disk. The procedure assumes that the new disk is installed on the system and partitioned.

Caution: All files on the additional disk are destroyed by this procedure.

1. Make a backup of the filesystem you are going to extend.
2. Unmount the */disk2* filesystem:

```
# umount /disk2
```

3. Use *xlv_make* to create an XLV logical volume out of the */disk2* partition and the new disk. The */disk2* partition must be the first volume element in the data subvolume. For example:

```
# xlv_make
xlv_make> vol xlv0
xlv0
xlv_make> data
xlv0.data
xlv_make> plex
xlv0.data.0
xlv_make> ve dks0d2s7
xlv0.data.0.0
xlv_make> ve dks0d3s7
```

```
xlv0.data.0.1
xlv_make> end
Object specification completed
xlv_make> exit
Newly created objects will be written to disk.
Is this what you want?(yes) yes
Invoking xlv_assemble
```

4. Mount the */disk2* filesystem:

```
# mount /dev/dsk/xlv/xlv0 /disk2
```

5. Grow the filesystem into the logical volume with the *xfs_growfs* command:

```
# xfs_growfs /disk2
```

6. Change the entry for */disk2* in the file */etc/fstab* to mount the logical volume rather than the disk partition:

```
/dev/dsk/xlv/xlv0 /disk2 xfs rw,raw=/dev/rdsk/xlv/xlv0 0 0
```

Growing an EFS Filesystem Onto Another Disk

The following steps show how to grow a fictional */work* EFS filesystem onto an *lv* logical volume created out of the */work* disk partition and a new disk. The procedure assumes that the new disk is installed on the system and partitioned.

Caution: All files on the new disk are destroyed by this procedure.

1. Make a backup of the filesystem you are going to extend.
2. Place an entry in the file */etc/lvtab* for the logical volume. The entry should look something like this:

```
lv0:Project Volume:devs=/dev/dsk/dks0d2s7,/dev/dsk/dks0d3s7
```

An */etc/lvtab* entry is made up of several colon-separated fields. In the above example:

lv0	The device name of the logical volume. It must be lv followed by a single digit.
Project Volume	The volume label. This field is optional, but may be useful for commands to verify the volume associated with the device.

devs=/dev/dsk/dks0d2s7,/dev/dsk/dks0d3s7
: The disk partitions that make up the logical volume. The first partition must be the existing partition.

This example shows a logical volume composed of two disk partitions, but it could be made up of several partitions. The only limit is the maximum size of a filesystem, 8 GB. For more information on */etc/lvtab* entries, see the section "Creating Entries in the /etc/lvtab File" in Chapter 8. When using a logical volume to extend an existing filesystem, the logical volume *cannot* be striped.

3. Change the entry for */work* in the file */etc/fstab* to read:

   ```
   /dev/dsk/lv0 /work efs rw,raw=/dev/rdsk/lv0 0 0
   ```

4. Unmount the */work* filesystem:

   ```
   # umount /work
   ```

5. Run the *mklv* command with the device name of the logical volume as an argument to create the logical volume:

   ```
   # mklv -f lv0
   ```

6. Run *lvck* to check the new logical volume:

   ```
   # lvck /dev/rdsk/lv0
   ```

7. Grow the filesystem into the logical volume with the *growfs* command:

   ```
   # growfs /dev/rdsk/lv0
   ```

8. Run *fsck* on the expanded filesystem:

   ```
   # fsck /dev/rdsk/lv0
   ```

9. Mount the logical volume:

   ```
   # mount /work
   ```

You can repeat this expansion process indefinitely. You can always add a new disk, add its name to the *lvtab* entry, and then rerun *mklv* and *growfs* to further expand the filesystem.

Converting Filesystems on the System Disk From EFS to XFS

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

This section explains the procedure for converting filesystems on the system disk from EFS to XFS. Some systems have two filesystems on the system disk, the Root filesystem (mounted at /) and the Usr filesystem (mounted at */usr*). Other systems have a single, combined Root and Usr filesystem mounted at /. This procedure covers both cases but assumes that neither *lv* nor XLV logical volumes are in use on the system disk. The basic procedure for converting a system disk is:

1. Load the miniroot.
2. Do a complete dump of filesystems on the system disk.
3. Repartition the system disk if necessary.
4. Create one or two new, empty XFS filesystems.
5. Restore the files from the filesystem dumps.
6. Reboot the system.

During this procedure, you can repartition the system disk if needed. For example, you can convert from separate Root and Usr filesystems to a single, combined filesystem, or you can resize partitions to make the root partition larger and the usr partition smaller. See the section "Disk Repartitioning" in this chapter for more information.

The early steps of this procedure ask you to identify the values of various variables, which are used later in the procedure. You may find it helpful to make a list of the variables and values for later reference. Be sure to perform only the steps that apply to your situation. Perform all steps as superuser.

Caution: It is very important to follow this procedure as documented without giving additional *inst* or shell commands. Unfortunately, deviations from this procedure, even changing to a different directory or going from the *inst* shell to an *inst* menu when not directed to, can have very severe consequences from which recovery is difficult.

1. Review the subsections within the section "Planning for XFS Filesystems" in this chapter to verify that you are ready to begin this procedure.
2. Verify that your backups are up to date. Because this procedure temporarily removes all files from your system disk, it is important that you have a complete set of backups that have been prepared using your normal backup procedures. You will make a complete dump of the system disk starting at step 11, but you should have your usual backups in addition to the backup made during this procedure.
3. Use *prtvtoc* to get the device name of the root disk partition, *rootpartition*. For example:

   ```
   # prtvtoc
   Printing label for root disk

   * /dev/rdsk/dks0d1s0 (bootfile "/unix")
   ...
   ```

   The `bootfile` line contains the raw device name of the root disk partition, which is */dev/rdsk/dks0d1s0* in this example. *rootpartition* is the block device name, which is */dev/dsk/dks0d1s0* in this example.
4. If the system disk has separate Root and Usr filesystems, use the output of *prtvtoc* in the previous step to figure out the device name of the usr partition. This is the value of the variable *usrpartition*, which is used later in this procedure. Look for the line that shows a mount directory of */usr*:

```
Partition  Type  Fs   Start: sec   (cyl)    Size: sec    (cyl)  Mount Directory
...
6          efs  yes      116725  ( 203)       727950   (1266)   /usr
```

   The usr partition number is shown in the first column of this line; it is 6 in this example. To determine the value of *usrpartition*, replace the final digit in *rootpartition* with the usr partition number. For this example, *usrpartition* is */dev/dsk/dks0d1s6*.
5. If you are using a tape drive as the backup device, use *hinv* to get the controller and unit numbers (*tapecntlr* and *tapeunit*) of the tape drive. For example:

   ```
   # hinv -c tape
   Tape drive: unit 2 on SCSI controller 0: DAT
   ```

   In this example, *tapecntlr* is 0 and *tapeunit* is 2.

6. If you are using a disk drive as your backup device, use *df* to get the device name (*backupdevice*) and mount point (*backupfs*) of the partition that contains the filesystem where you plan to put the backup. For example:

```
# df
Filesystem                 Type  blocks     use   avail %use  Mounted on
/dev/root                   efs 1992630  538378 1454252  27%  /
/dev/dsk/dks0d3s7           efs 3826812 1559740 2267072  41%  /disk3
/dev/dsk/dks0d2s7           efs 2004550      23 2004527   0%  /disk2
```

   The filesystem mounted at */disk2* has plenty of disk space for a backup of the system disk (/ uses 538,378 blocks, and */disk2* has 2,004,527 blocks available). The *backupdevice* for */disk2* is */dev/dsk/dks0d2s7* and the *backupfs* is */disk2*.

7. Create a temporary copy of */etc/fstab* called */etc/fstab.xfs* and edit it with your favorite editor. For example:

```
# cp /etc/fstab /etc/fstab.xfs
# vi /etc/fstab.xfs
```

   Make these changes in */etc/fstab.xfs*:

   - Replace `efs` with `xfs` in the line for the Root filesystem, /, if there is a line for the Root filesystem.
   - If there is no line for the Root filesystem, add this line:

     ```
     /dev/root   /   xfs rw,raw=/dev/rroot 0 0
     ```

   - If Root and Usr are separate filesystems and will remain so, replace `efs` with `xfs` in the line for the Usr filesystem.
   - If Root and Usr have been separate filesystems, but the disk will be repartitioned during the conversion procedure so that they are combined, remove the line for the Usr filesystem.

8. Shut down your workstation using the *shutdown* command or the “System Shutdown” item on the System toolchest. Answer prompts as appropriate to get to the five-item System Maintenance Menu.

9. Bring up the miniroot from system software CDs or a software distribution directory.

10. Switch to the shell prompt in *inst*:

```
Inst> sh
```

11. Create a full backup of the Root filesystem by giving this command:

    ```
    # /root/sbin/dump 0uCf tapesize dumpdevice rootpartition
    ```

    tapesize is the tape capacity (it's used for backup to disks, too) and *dumpdevice* is the appropriate device name for the tape drive or the name of the file that will contain the dump image. Table 4-1 gives the values of *tapesize* and *dumpdevice* for different tape drives and disk. <*tapecntlr*> and <*tapeunit*> in Table 4-1 are *tapecntlr* and *tapeunit* from step 5 in this section.

Table 4-1 *dump* Arguments for Filesystem Backup

Backup Device	*tapesize*	*dumpdevice*
Disk	2m	Use /root/*backupfs*/root.dump for the Root filesystem and /root/*backupfs*/usr.dump for the Usr filesystem
DAT tape	2m	/dev/rmt/tps<*tapecntlr*>d<*tapeunit*>nsv
DLT tape	10m	/dev/rmt/tps<*tapecntlr*>d<*tapeunit*>nsv
EXABYTE™ 8mm model 8200 tape	2m	/dev/rmt/tps<*tapecntlr*>d<*tapeunit*>nsv
EXABYTE 8mm model 8500 tape	4m	/dev/rmt/tps<*tapecntlr*>d<*tapeunit*>nsv
QIC cartridge tape	150k	/dev/rmt/tps<*tapecntlr*>d<*tapeunit*>ns

12. If Usr is a separate filesystem, insert a new tape (if you are using tape), and create a full backup of the Usr filesystem by giving this command:

    ```
    # /root/sbin/dump 0uCf tapesize dumpdevice usrpartition
    ```

    tapesize is the tape capacity (it's used for backup to disks, too) and *dumpdevice* is the appropriate device name for the tape drive or the name of the file that will contain the dump image. Table 4-1 gives the values of *tapesize* and *dumpdevice* for different tape drives and disk.

13. Exit out of the shell:

    ```
    # exit
    ...
    Inst>
    ```

14. If you do not need to repartition the system disk, skip to step 18.

15. To repartition the system disk, use the standalone version of *fx*. This version of *fx* is invoked from the Command Monitor, so you must bring up the Command Monitor. To do this, quit out of *inst*, reboot the system, shut down the system, then request the Command Monitor. An example of this procedure is:

```
Inst> quit
...
Ready to restart the system.  Restart? { (y)es, (n)o, (sh)ell, (h)elp }: yes
...
login: root
# halt
...
System Maintenance Menu
...
Option? 5
Command Monitor.  Type "exit" to return to the menu.
>>
```

On systems with a graphical System Maintenance Menu, choose the last option, `Enter Command Monitor`, instead of choosing option 5.

16. Boot *fx* and repartition the system disk so that it meets your needs. The example below shows how to use *fx* to switch from separate root and usr partitions to a single root partition.

```
>> boot stand/fx
84032+11488+3024+331696+26176d+4088+6240 entry: 0x89f97610
114208+29264+19536+2817088+60880d+7192+11056 entry: 0x89cd31c0
Currently in safe read-only mode.
Do you require extended mode with all options available? (no) <Enter>
SGI Version 5.3 ARCS   Dec 14, 1994
fx: "device-name" = (dksc) <Enter>
fx: ctlr# = (0) <Enter>
fx: drive# = (1) <Enter>
...opening dksc(0,1,0)
...controller test...OK
Scsi drive type == SGI     SEAGATE ST31200N8640

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

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

----- 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
fx> exit
```

17. Load the miniroot again, using the same procedure you used in step 9.
18. Make an XFS filesystem for Root:

```
Inst> admin mkfs /dev/dsk/dks0d1s0
Unmounting device "/dev/dsk/dks0d1s0" from directory "/root".

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): y

        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=4096 /dev/dsk/dks0d1s0
meta-data=/dev/rdsk/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
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.
```

19. Switch to the shell prompt in *inst*:

    ```
    Inst> sh
    ```

20. If you made the backup on disk, create a mount point for the filesystem that contains the backup and mount it:

    ```
    # mkdir /backupfs
    # mount backupdevice /backupfs
    ```

21. If you made the backup on tape, restore all files on the Root filesystem from the backup you made in step 11 by putting the correct tape in the tape drive and giving these commands:

    ```
    # cd /root
    # mt -t /dev/rmt/tpstapecntlrdtapeunit rewind
    # restore rf dumpdevice
    ```

    You may need to be patient while the restore is taking place; it normally doesn't generate any output and it can take a while.

22. If you made the backup on disk, restore all files on the Root filesystem from the backup you made in step 11 by giving these commands:

    ```
    # cd /root
    # restore rf /backupfs/root.dump
    ```

23. If you made a backup of the Usr filesystem in step 12 on tape, restore all files in the backup by putting the correct tape in the tape drive and giving these commands:

    ```
    # cd /root/usr
    # mt -t /dev/rmt/tpstapecntlrdtapeunit rewind
    # restore rf dumpdevice
    ```

24. If you made a backup of the Usr filesystem in step 12 on disk, restore all files in the backup by giving these commands:

    ```
    # cd /root/usr
    # restore rf /backupfs/usr.dump
    ```

25. Move the new version of */etc/fstab* that you created in step 7 into place (the first command, which is optional, saves the old version of */etc/fstab*):

    ```
    # mv /root/etc/fstab /root/etc/fstab.old
    # mv /root/etc/fstab.xfs /root/etc/fstab
    ```

26. Exit from the shell and *inst* and restart the system:

```
# exit
#
Calculating sizes .. 100% Done.

Inst> quit
...
Ready to restart the system.  Restart? { (y)es, (n)o, (sh)ell, (h)elp }: yes
Preparing to restart system ...

The system is being restarted.
```

Converting a Filesystem on an Option Disk From EFS to XFS

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.

This section explains how to convert an EFS filesystem on an option disk (a disk other than the system disk) to XFS. It assumes that neither *lv* nor XLV logical volumes are used. You must be superuser to perform this procedure.

1. Review the subsections within the section “Planning for XFS Filesystems” in this chapter to verify that you are ready to begin this procedure.
2. Verify that your backups are up to date. Because this procedure temporarily removes all files from the filesystem you convert, it is important that you have a complete set of backups that have been prepared using your normal backup procedures. You will make a complete backup of the system disk in step 4, but you should have your usual backups in addition to the backup made during this procedure.
3. Identify the device name of the partition, which is the variable *partition*, where you plan to create the filesystem. For example, if you plan to use partition 7 (the entire disk) of an option disk on controller 0 and drive address 2, *partition* is */dev/dsk/dks0d2s7*. For more information on determining *partition* (also known as a *special* file), see the dks(7M) reference page.

4. Back up all files on the disk partition to tape or disk because they will be destroyed by the conversion process. You can use any backup command (*Backup*, *bru*, *cpio*, *tar*, and so on) and back up to a local or remote tape drive or a local or remote disk. For example, the command for *dump* for local tape is:

   `# dump 0uCf` *tapesize dumpdevice partition*

   tapesize is the tape capacity (it's used for backup to disks, too) and *dumpdevice* is the device name for the tape drive. Table 4-1 gives the values of *tapesize* and *dumpdevice* for different local tape drives and disk. You can get the values of *tapecntlr* and *tapeunit* used in the table from the output of the command *hinv –c tape*.

5. Unmount the partition:

   `# umount` *partition*

6. Use the *mkfs* command to create the new XFS filesystem:

   `# mkfs –b size=`*blocksize* `–l size=`*logsize partition*

   blocksize is the filesystem block size (see the section "Choosing the Filesystem Block Size and Extent Size" in this chapter) and *logsize* is the size of the area dedicated to log records (see the section "Choosing the Log Type and Size" in this chapter). Example 4-1 shows an example of this command line and its output.

7. Mount the new filesystem with this command:

   `# mount` *partition mountdir*

8. In the file */etc/fstab*, in the entry for *partition*, replace efs with xfs. For example:

   partition mountdir `xfs rw,raw=`*rawpartition* `0 0`

   rawpartition is the raw version of *partition*.

9. Restore the files to the filesystem from the backup you made in step 4. For example, if you gave the *dump* command in step 4, the commands to restore the files from tape are:

   ```
   # cd mountdir
   # mt –t device rewind
   # restore rf dumpdevice
   ```

   The value of *device* is the same as *dumpdevice* without `nsv` or other letters at the end.

   You may need to be patient while the restore is taking place; it doesn't generate any output and it can take a while.