Chapter 5

Maintaining Filesystems

This chapter describes administration procedures for maintaining XFS and EFS filesystems that may need to be performed on a routine or as-needed basis. It is extremely important to maintain filesystems properly, in addition to backing up the data they contain. Failure to do so might result in loss of valuable system and user information.

The major sections in this chapter are:

Routine Filesystem Administration Tasks

To administer filesystems, you need to do the following:

- Monitor the amount of free space and free inodes available.
- If a filesystem is chronically short of free space, take steps to alleviate the problem, such as removing old files and imposing disk usage quotas.
- Periodically check EFS filesystems for data integrity using *fsck*. (XFS filesystems are checked with *xfs_check* only when a problem is suspected.)
- Back up filesystems.

Many routine administration jobs can be performed by shell scripts. Here are a few ideas:

- Use a shell script to investigate free blocks and free inodes, and report on filesystems whose free space dips below a given threshold.
- Use a shell script to automatically "clean up" files that grow (such as log files).
- Use a shell script to highlight cases of excessive disk use.

All of these scripts can be run automatically by the *cron* command and the output sent to you using electronic mail. Typically, these scripts use some combination of the *find*, *du*, *Mail*, and shell commands.

The process accounting system performs many similar functions. If the process accounting system does not meet your needs, examine the scripts in */usr/lib/acct*, such as *ckpacct* and *remove*, for ideas about how to build your own administration scripts.

Mounting and Unmounting Filesystems

As explained in the section "Filesystem Mounting and Unmounting" in Chapter 3, to be accessed by IRIX, filesystems must be mounted. The subsections below explain the use of the *mount* and *umount* commands and the file */etc/fstab* to mount and unmount filesystems.

Tip: You can mount and unmount XFS filesystems using the graphical user interface of the *xfsm* command. For information, see its online help.

Manually Mounting Filesystems

The *mount* command is used to manually mount filesystems. The basic forms of the *mount* command are:

`mount` *device_file mount_point_directory*

`mount` *host*`:`*directory mount_point_directory*

device_file is a block device file. *host*:*directory* is the hostname and pathname of a remote directory that has been exported on the remote host by using the *exportfs* command on the remote host (it requires NFS). *mount_point_directory* is the mount point directory. The mount point must already exist (you can create it with the *mkdir* command).

If you omit either the *device_file* or the *mount_point_directory* from the *mount* command line, *mount* checks the file */etc/fstab* to find the missing argument (see the section "Mounting Filesystems Automatically With the /etc/fstab File" for more information about */etc/fstab*).

For example, to mount a filesystem manually, use this command:

```
mount /dev/dsk/dks0d1s6 /usr
```

Another example, which uses a mnemonic device file name, is:

```
mount /dev/usr /usr
```

An example of a *mount* command for a filesystem that is listed in */etc/fstab* is:

```
mount /d2
```

Other useful *mount* commands are:

mount -a	Mount all filesystems listed in */etc/fstab*.
mount -h *host*	Mount all filesystems listed in */etc/fstab* that are remote-mounted from the system named *host*.

See the mount(1M) reference page for more information about the *mount* command.

Mounting Filesystems Automatically With the */etc/fstab* File

The */etc/fstab* file contains information about every filesystem and swap partition that is to be mounted automatically when the system is booted into multi-user mode. In addition, the */etc/fstab* file is used by the *mount* command when only the device block file or the mount point is given to the *mount* command. Filesystems that are not mounted with the *mount* command, such as the */proc* filesystem, are not listed in */etc/fstab*.

The procedure in this section explains how to add an entry for a filesystem to */etc/fstab*.

For each filesystem that is to be mounted every time the system is booted, a line similar to this appears in the file */etc/fstab*:

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

The fields in this line are defined as follows:

/dev/dsk/dks0d2s7
: The block device file of the partition where the filesystem is located.

/test
: The name of the directory where the filesystem will be mounted (the mount point).

efs
: The type of filesystem. In this case, the filesystem is an EFS filesystem.

rw, raw=
: These are some of many options available when mounting a filesystem (see the fstab(4) reference page for a complete list). In this instance, the filesystem is to be mounted read-write, so that *root* and other users can write to it. The raw= option gives the filesystem's raw device filename. It should be the last option in the options list.

0 0
: These two numbers represent the frequency of dump cycles and the *fsck* pass priority. These two numbers must be added after the last option in the options list (raw =). The fstab(4) reference page contains additional information.

If you have already mounted the filesystem as described in the section "Manually Mounting Filesystems," you can use the *mount* command to determine the appropriate */etc/fstab* entry. For example:

```
mount -p
```

This command displays all currently mounted filesystems, including the new filesystem in */etc/fstab* format. Copy the line that describes the new filesystem to */etc/fstab*.

The *mount* command reads */etc/fstab* sequentially; therefore, filesystems that are mounted beneath other filesystems must follow their parent partitions in */etc/fstab* in order for their mount points to exist.

The swap partition on the system disk (partition 1) is not listed in */etc/fstab*. However, additional swap partitions added to the system are listed. For swap partitions, the mount point field is not used. See the guide *IRIX Admin: System Configuration and Operation* and the swap(1M) reference page for more information.

See the fstab(4) reference page for more information about */etc/fstab*
entries.

Mounting a Remote Filesystem Automatically

If you have the optional NFS software, you can automatically mount any remote filesystem whenever it is accessed (for example, by changing directories to the filesystem with *cd*). The remote filesystem must be exported with the *exportfs* command.

For complete information about setting up automounting, including all the available options, see the automount(1M) and exportfs(1M) reference pages. These commands are discussed more completely in the *ONC3/NFS Administrator's Guide*.

Unmounting Filesystems

Filesystems are automatically unmounted when the system is shut down. To manually unmount filesystems, use the *umount* command. The three basic forms of the command are shown in Table 5-1. Local filesystems can be unmounted with either of the first two forms shown in the table; they are equivalent. Similarly, the first and third forms are equivalent for remote filesystems.

Table 5-1 Forms of the *umount* Command

Command	Comments
umount *mount_point_directory*	*mount_point_directory* is a directory pathname that is the mount point for the filesystem. This form can be used for local or remote filesystems.
umount *device_file*	*device_file* is a block device file name. This form is only for local filesystems.
umount *host*:*directory*	*host*:*directory* is a remote directory. This form is only for remote filesystems.
umount -a	Attempt to unmount all the filesystems currently mounted (listed in /etc/mtab) except / and */usr*. This command is not the complement of the **mount -a** command, which mounts all filesystems listed in */etc/fstab*.

For example, to unmount a local or remote filesystem mounted at */d2*, give this command:

```
umount /d2
```

To unmount the filesystem on the partition */dev/dsk/dks0d1s7*, give this command:

```
umount /dev/dsk/dks0d1s7
```

To unmount the remote-mounted (NFS) filesystem *depot:/usr/spool/news*, give this command:

```
umount depot:/usr/spool/news
```

To be unmounted, a filesystem must not be in use. If it is in use and you try to unmount it, you get a "Resource busy" message. These messages and their solutions are explained in the umount(1M) reference page.

Managing Disk Space

At some point, you are likely to find yourself short on disk space. In addition to using disk space intentionally for new files, you and other users may be creating and retaining files that you do not need. Some possible sources of these files are:

- People tend to forget about files they no longer use. Outdated files often stay on the system much longer than necessary.
- Some files, particularly log files such as */var/adm/SYSLOG*, grow as a result of normal system operations. Normally, *cron* rotates this file once per week so that it does not grow excessively large. (See */var/spool/cron/crontabs/root*.) However, you should check this file periodically to make sure it is being rotated properly, or when the amount of free disk space has grown small.
- The *lost+found* directory at the root of EFS filesystems may be full. If you log in as *root*, you can check this directory and determine if the files there can be removed.
- Some directories, notably */tmp*, */usr/tmp*, and */var/tmp*, accumulate files. These are often copies of files being manipulated by text editors and other programs. Sometimes these temporary files are not removed by the programs that created them.
- The directories */usr/tmp*, */var/tmp*, and */var/spool/uucppublic* are public directories; people often use them to store temporary copies of files they are transferring to and from other systems and sites. Unlike */tmp*, they are not cleaned out when the system is rebooted. The site administrator should be even more conscientious about monitoring disk use in these directories.
- Users move old files to the dumpster without realizing that such files are not fully deleted from the system.

- *vmcore* and *unix* files in */var/adm/crash* are accumulating without being removed.
- Binary core dumps, *core* files, from crashed application programs are not being removed.

Tip: The section “Freeing Up Disk Space” in Chapter 6 of the *Personal System Administration Guide* provides additional ideas for identifying unnecessary files.

The following subsections describe various techniques for monitoring disk space usage, locating unneeded files, and limiting disk usage by individual users.

Monitoring Free Space and Free Inodes

You can quickly check the amount of free space and free inodes with the *df* command. For example,

```
%  df
Filesystem                 Type  blocks     use    avail %use  Mounted on
/dev/root                   efs 1939714 1326891   612823  68%  /
```

The avail column shows the amount of free space in blocks.

To determine the number of free inodes, use this command:

```
% df -i
Filesystem      Type  blocks     use    avail %use    iuse  ifree %iuse  Mounted
/dev/root        efs 1939714 1326891  612823  68%   14491 195031    7%  /
```

You see a listing similar to the first *df* listing, except that it also lists the number of inodes in use, the number of inodes that are free (available), and the percentage of inodes in use. For XFS filesystems, the number of free inodes is the maximum number that could be allocated if needed. XFS allocates inodes as needed. On XFS filesystems inode usage is very high only on very full filesystems.

On EFS filesystems, when a filesystem is more than about 90-95% full, system performance may degrade, depending on the size of the disk. (The number of free disk blocks on a 97% full large disk is larger than the number of free disk blocks on a 97% full small disk.) You should monitor the amount of available space and take steps to keep an adequate amount available. XFS filesystem performance doesn’t degrade when XFS filesystems are very full.

Tip: The section "Monitoring Disk Space and Setting a Warning Level" in Chapter 6 of the *Personal System Administration Guide* describes how to use the Disk Manager in the System Toolchest to tell the system to issue warnings when disks reach capacities that you specify.

Monitoring Key Files and Directories

Almost any system that is used daily has several key files and directories that grow through normal use. Some examples are shown in Table 5-2.

Table 5-2 Files and Directories That Tend to Grow

File	Use
/etc/wtmp	history of system logins
/tmp	directory for temporary files (Root filesystem)
/var/adm/avail/availlog	log file for the availabilility monitor (see the availmon(5) reference page)
/var/adm/avail/notifylog	log file for the availabilility monitor (see the availmon(5) reference page)
/var/adm/sulog	history of *su* commands
/var/cron/log	history of actions of *cron*
/var/spool/lp/log	history of actions of *lp*
/var/spool/uucp	directory for *uucp* log files
/var/tmp	directory for temporary files

The frequency with which you should check growing files depends on how active your system is and how critical the disk space problem is. A good technique for keeping them down to a reasonable size uses a combination of the *tail* and *mv* commands:

```
# tail -50 /var/adm/sulog > /var/tmp/sulog
# mv /var/tmp/sulog /var/adm/sulog
```

This sequence puts the last 50 lines of */var/adm/sulog* into a temporary file, then moves the temporary file to */var/adm/sulog*. This reduces the file to the 50 most recent entries. It is often useful to have these commands performed automatically every week using *cron*. For more information on using *cron* to automate your regular tasks, see the cron(1M) reference page.

Cleaning Out Temporary Directories

The directory */tmp* and all of its subdirectories are automatically cleaned out every time the system is rebooted. You can control whether or not this happens with the *chkconfig* option **nocleantmp**. By default, **nocleantmp** is off, and thus */tmp* is cleaned.

The directory */var/tmp* is not automatically cleaned out when the system is rebooted. This is a fairly standard practice on IRIX systems. If you wish, you can configure IRIX to automatically clean out */var/tmp* whenever the system is rebooted. Changing this standard policy is a fairly extreme measure, and many people expect that files left in */var/tmp* are not removed when the system is rebooted. Do not make this change without warning users well in advance.

If you must change the policy, this is how to do it:

1. Notify everyone who uses the system that you are changing the standard policy regarding */var/tmp*, and that all files left in */var/tmp* will be removed when the system is rebooted. Send electronic mail and post a message in the */etc/motd* file.

   Give the users at least one week's notice, longer if possible.

2. Copy the file */etc/init.d/rmtmpfiles* to a new file in the same directory, for example, */etc/init.d/rmtmpfiles2*:

   ```
   # cd /etc/init.d
   # cp rmtmpfiles rmptmpfiles2
   ```

3. Open *rmtmpfiles2* for editing, for example:

   ```
   # vi rmtmpfiles2
   ```

4. Find a block of commands in the file that looks something like this:

   ```
   # make /var/tmp exist
   if [ ! -d /var/tmp ]
   then
           rm -f /var/tmp # remove the directory
           mkdir /var/tmp
   fi
   ```

5. Before the `fi` statement add the following lines:

```
else
        # clean out /var/tmp
        rm -f /var/tmp/*
```

The complete block of commands should look something like this:

```
# make /var/tmp exist
if [ ! -d /var/tmp ]
then
        rm -f /var/tmp # remove the directory
        mkdir /var/tmp
else
        # clean out /var/tmp
        rm -f /var/tmp/*
fi
```

6. Save the file and exit the editor.

7. Create a link to the new file in the directory */etc/rc2.d*, following the naming conventions described in */etc/init.d/README*. For example:

```
# cd ../rc2.d
# ln -s ../init.d/rmtmpfiles S59rmtmpfiles2
```

Locating Unused Files

Part of the job of cleaning up filesystems is locating and removing files that have not been used recently. The *find* command can locate files that have not been accessed recently.

The *find* command searches for files, starting at a directory named on the command line. It looks for files that match whatever criteria you wish, for example all regular files, all files that end in *.trash*, or any file older than a particular date. When it finds a file that matches the criteria, it performs whatever task you specify, such as removing the file, printing the name of the file, changing the file's permissions, and so forth.

For example:

```
# find /usr -local -type f -mtime +60 -print > /usr/tmp/deadfiles &
```

In the above example:

`/usr` specifies the pathname where *find* is to start.

`-local` restricts the search to files on the local system.

`-type f` tells *find* to look only for regular files and to ignore special files, directories, and pipes.

`-mtime +60` says you are interested only in files that have not been modified in 60 days.

`-print` means that when a file is found that matches the **-type** and **-mtime** expressions, you want the pathname to be printed.

`> /usr/tmp/deadfiles &`
directs the output to the temporary file */usr/tmp/deadfiles* and runs in the background. Redirecting the results of the search in a file is a good idea if you expect a large amount of output.

As another example, you can use the *find* command to find files over 7 days old in the temporary directories and remove them. Use the following commands:

```
# find /var/tmp -local -atime 7 -exec rm {} \;
# find /tmp -local -atime 7 -exec rm {} \;
```

This example shows how to use *find* to locate and remove all core files over a week old:

```
# find  / -local -name core -atime +7 -exec rm {} \;
```

See the cron(1M) reference page for information on using the *cron* command to automate the process of locating and possibly removing.

Identifying Accounts That Use Large Amounts of Disk Space

Four commands are useful for tracking down accounts that use large amounts of space: *du*, *find*, *quot*, and *diskusg*.

du displays disk use, in blocks, for files and directories. For example:

```
# du /usr/share/catman/u_man
5       /usr/share/catman/u_man/cat1/audio
266     /usr/share/catman/u_man/cat1/Xm
1956    /usr/share/catman/u_man/cat1/X11
72      /usr/share/catman/u_man/cat1/Inventor
413     /usr/share/catman/u_man/cat1/dmedia
752     /usr/share/catman/u_man/cat1/explorer
12714   /usr/share/catman/u_man/cat1
1       /usr/share/catman/u_man/cat3/audio
63      /usr/share/catman/u_man/cat3
12      /usr/share/catman/u_man/cat6/video
1077    /usr/share/catman/u_man/cat6
92      /usr/share/catman/u_man/cat2
425     /usr/share/catman/u_man/cat4
170     /usr/share/catman/u_man/cat5
13      /usr/share/catman/u_man/cat1m
14557   /usr/share/catman/u_man
```

This displays the block count for all directories in the directory */usr/share/catman/u_man*. By default the *du* command displays disk use in 512-byte blocks. To display disk use in 1024-byte blocks, use the **-k** option. For example:

```
# du -k /usr/people/ralph
```

The **-s** option produces a summary of the disk use in a particular directory. For example:

```
# du -s /usr/people/alice
```

For a complete description of *du* and its options, see the du(1M) reference page.

Use *find* to locate specific files that exceed a given size limit. For example:

```
# find /usr -local -size +10000 -print
```

This example produces a display of the pathnames of all files (and directories) in the Usr filesystem that are larger than 10,000 512-byte blocks.

The *quot* command reports the amount of disk usage per user on an EFS filesystem. It is part of the disk quotas subsystem, although you need not use quotas to use this command. You can use the output of this command to inform your users of their disk space usage. An example of the command is:

```
# /usr/etc/quot /
/dev/root (/):
  371179    root
  265712    ellis
   12606    aevans
    7927    demos
    5526    bin
    2744    lp
     682    uucp
     379    guest
     207    adm
       7    sys
```

The *diskusg* command is part of the process accounting subsystem and serves the same purpose as *quot*. *diskusg*, though, is typically used as part of general system accounting and can be used on both EFS and XFS filesystems. This command generates disk usage information on a per-user basis. For example,

```
# /usr/lib/acct/diskusg /dev/root
0       root    736795
2       bin     11035
3       uucp    1342
4       sys     9
5       adm     1011
9       lp      5418
126     ellis   528263
993     demos   15737
998     guest   740
5315    aevans  24836
```

diskusg prints one line for each user identified in the */etc/passwd* file. Each line contains the user's UID number and login name, and the total number of 512-byte blocks of disk space currently being used by the account.

The output of *diskusg* is normally the input to *acctdisk* (see the acct(1M) reference page), which generates total disk accounting records that can be merged with other accounting records. For more information on the accounting subsystem, consult the guide *IRIX Admin: Backup, Security, and Accounting* and the acct(4) reference page.

Running Out of Space in the Root Filesystem

For systems that have separate Root and Usr filesystems, running out of disk space on the Root filesystem can occur for several reasons:

- New software options that place files in the Root filesystem have been installed.
- A new IRIX release that requires more disk space in the Root filesystem has been installed.
- Files created while filesystems were unmounted have been unintentionally placed in the Root filesystem instead of their intended filesystem. For example, say that the Usr filesystem is unmounted and the file */usr/tempfile* is created. When the Usr filesystem is mounted at */usr*, the file */usr/tempfile* isn't accessible, but it is still using disk space.
- Applications that create files in */tmp* are creating many files or very large files that fill up the Root filesystem.

Several possible courses of action when the Root filesystem is too full are listed below. You may want to pursue several of them.

- Check for hidden files. Unmount filesystems other than the Root filesystem (you may find this easiest to do from the miniroot) and list the contents of each of the mount point directories.
- Check the */lost+found* directory. You may find that large files have accumulated there.
- Increase the size of the Root filesystem by combining the Root and Usr filesystems or by making the Root filesystem larger by taking disk space from the Usr filesystem.
- Identify applications that are creating files in */tmp* and cause the most problems, and configure them to use */usr/tmp* instead of */tmp* for temporary files. Most applications recognize the *TMPDIR* environment variable, which specifies the directory to use instead of the default. For example, with *csh*:

  ```
  % setenv TMPDIR /usr/tmp
  ```

  With *sh*:

  ```
  % TMPDIR=/usr/tmp ; export TMPDIR
  ```

- Make */tmp* a mounted filesystem. (See the section "Mount a Filesystem as a Subdirectory" in Chapter 3.) You can "carve" a */tmp* filesystem out of other filesystems if need be.

Imposing Disk Quotas

The use of disk quotas to limit users' use of disk space is discussed in the section "Disk Quotas" in Chapter 3. They can be used only on EFS filesystems. To impose soft disk quotas, follow these steps:

1. To enable the quotas subsystem, give the commands:

   ```
   # chkconfig quotas on
   # chkconfig quotacheck on
   ```

2. Create a file named *quotas* in the root directory of each filesystem that is to have a disk quota. This file should be zero length and should be writable only by *root*. To create the *quotas* file, give this command as *root* in the root directory of each of these filesystems:

   ```
   # touch quotas
   ```

3. Establish the quota amounts for individual users. The *edquota* command can be used to set the limits desired upon each user. For example, to set soft limits of 100 MB and 100 inodes on the user ID sedgwick, give the following command:

   ```
   # edquota sedgwick
   ```

   The screen clears, and you are placed in the *vi* editor to edit the user's disk quota. You see:

   ```
   fs /  kbytes(soft=0, hard=0)  inodes(soft=0, hard=0)
   ```

   The filesystem appears first, in this case the Root filesystem (/). The numeric values for disk space are in kilobytes, not megabytes, so to specify 100 megabytes, you must multiply the number by 1024. The number of inodes should be entered directly.

4. Edit the line to appear as follows:

   ```
   fs / kbytes(soft=102400, hard=0)  inodes(soft=100, hard=0)
   ```

5. Save the file and quit the editor when you have entered the correct values. If you leave the value at 0, no limit is imposed. Since you are setting only soft limits in this example, the hard values have not been set.

6. Use the **-p** option of *edquota* to assign the same quota to multiple users. Unless explicitly given a quota, users have no limits set on the amount of disk they can use or the number of files they can create.

7. Issue the *quotaon* command to put the quotas into effect. For quotas to be accurate, this command should be issued on a local filesystem immediately after the filesystem has been mounted. The *quotaon* command enables quotas for a particular filesystem, or with the **-a** option, enables quotas for all filesystems indicated in */etc/fstab* as using quotas. See the fstab(4) reference page for complete details on the */etc/fstab* file.

Quotas will be automatically enabled at boot time in the future. The script */etc/init.d/quotas* handles enabling of quotas and uses the *chkconfig* command to check the **quotas** configuration flag to decide whether or not to enable quotas. If you need to turn quotas off, use the *quotaoff* command.

Monitoring Disk Quotas

Periodically, check the records retained in the quota file for consistency with the actual number of blocks and files allocated to the user. Use the *quotacheck* command to verify compliance. It is not necessary to unmount the filesystem or disable the quota system to run this command, though on active filesystems, slightly inaccurate results may be seen. This command is run automatically at boot time by the */etc/init.d/quotas* script if the **quotacheck** flag has been turned on with *chkconfig*. *quotacheck* can take a considerable amount of time to execute, so it is convenient to have it done at boot time.

Copying XFS Filesystems With *xfs_copy*

The *xfs_copy* command can be used to copy an XFS filesystem with an internal log (XFS filesystems with external logs or real-time subvolumes cannot be copied with *xfs_copy*). One or more copies can be created on disk partitions, logical volumes, or files. Each copy has a unique filesystem identifier, which enables them to be run as separate filesystems on the same system. (Programs that do block-by-block copying, such as *dd*, do not create unique filesystem identifiers.) Multiple copies are created in parallel. For more information, see the xfs_copy(1M) reference page.

An example of the *xfs_copy* command is:

```
# xfs_copy /dev/dsk/dks0d3s7 /dev/dsk/dks5d2s7
... 10%  ... 20%  ... 30%  ... 40%  ... 50%  ... 60%  ... 70%  ...
80%  ... 90%  ... 100%
Done.
All copies completed.
```

Checking EFS Filesystem Consistency With *fsck*

Before checking an EFS filesystem other than the Root filesystem for consistency, the filesystem should be unmounted. (The Root filesystem can be checked while mounted.) Unmounting can be achieved by explicitly unmounting the filesystem, or by shutting the system down and bringing it up in single-user mode. (See the section "Unmounting Filesystems" in this chapter for information on unmounting filesystems and the single(1M) reference page for information on shutting the system down and bringing it up in single-user mode.) Checking unmounted filesystems is described in the section "Checking Unmounted Filesystems" below.

If you cannot shut down the system and cannot unmount the filesystem, but you need to perform the check immediately, you can run *fsck* in "no-write" mode. The *fsck* command checks the filesystem, but makes no changes and does not repair inconsistencies. The procedure is explained in the section "Checking Mounted Filesystems" below.

You may find it convenient to check multiple filesystems at once. This is also known as *parallel* checking. The *fsck* **-m** flag is used for parallel checking. For more information about this and other *fsck* options see the fsck(1M) reference page.

Checking Unmounted Filesystems

To check a single, unmounted filesystem, give this command as *root*:

```
# fsck filesystem
```

filesystem is the device file name of the filesystem's disk partition or logical volume, for example */dev/usr*, */dev/dsk/dks0d2s7, or /dev/dsk/lv2*; see the sections "Filesystem Names" in Chapter 3 and "Introduction to Logical Volumes" in Chapter 6 for more information.

As *fsck* runs, it proceeds through a series of steps, or *phases*. You may see an error-free check:

```
fsck: Checking /dev/usr
** Phase 1 - Check Blocks and Sizes
** Phase 2 - Check Pathnames
** Phase 3 - Check Connectivity
** Phase 4 - Check Reference Counts
** Phase 5 - Check Free List
7280 files 491832 blocks 38930 free
```

If there are no errors, you are finished checking the filesystem.

If errors are detected in the filesystem, *fsck* displays an error message. Appendix A, “Repairing EFS Filesystem ProblemsWith fsck,” explains how to proceed.

Checking Mounted Filesystems

If you cannot shut down the system and cannot unmount the filesystem, but you need to perform the check immediately, you can run *fsck* in “no-write” mode. The *fsck* command checks the filesystem, but makes no changes and does not repair inconsistencies.

For example, the following command invokes *fsck* in no-write mode:

```
# fsck -n /dev/usr
```

If any inconsistencies are found, they are not repaired. You must run *fsck* again without the **-n** flag to repair any problems. The benefit of this procedure is that you should be able to gauge the severity of the problems with your filesystem. The disadvantage of this procedure is that *fsck* may complain about inconsistencies that don’t really exist (because the filesystem is active).

Checking XFS Filesystem Consistency With *xfs_check*

The filesystem consistency checking command for XFS filesystems is *xfs_check*. (*fsck* is used only for EFS filesystems.) Unlike *fsck*, *xfs_check* is not invoked automatically on system startup; *xfs_check* should be used only if you suspect a filesystem consistency problem. Before running *xfs_check*, the filesystem to be checked should be unmounted.

The command line for *xfs_check* is:

```
# xfs_check device
```

device is the device file for a disk partition or logical volume that contains an XFS filesystem, for example */dev/dsk/xlv/xlv0*.

Unlike *fsck*, *xfs_check* does not repair any reported filesystem consistency problems; it only reports them. If *xfs_check* reports a filesystem consistency problem:

- If possible, contact the Silicon Graphics Technical Assistance Center for assistance (see the Release Notes for this product for more information).
- To attempt to recover from the problem, follow this procedure:
    1. Mount the filesystem using *mount –r* (read-only).
    2. Make a filesystem backup with *xfsdump*.
    3. Use *mkfs* to a make new filesystem on the same disk partition or XLV logical volume.
    4. Restore the files from the backup.