Chapter 2

Backup and Recovery Procedures

This chapter provides examples of how to use the various backup and recover tools described in Chapter 1.

All of the utilities discussed in this chapter support more options than can be shown here, but the examples combined with the discussions in Chapter 1 should provide enough information for you to choose and begin to use the tools best suited for your environment. For a complete description of the options available with a particular tool, refer to the reference page for that tool (for example, tar(1) for the *tar* command).

This chapter is divided into the following sections:

General Backup Procedure

Follow these steps when making a backup, no matter which backup utility you use:

1. Make sure the tape drive is clean. The hardware manual that came with your drive should state how, and how often, to clean the drive.

   Dirty tape heads can cause read and write errors. New tapes shed more oxide than older tapes, so you should clean your drive more frequently if you use a lot of new tapes.

2. Make sure you have enough backup media on hand. The *bru* utility has an option to check the size of a backup, so you can determine if you have enough media. You can also use such utilities as du(1) and df(1) to determine the size of directories and filesystems, respectively.

   Also, use good-quality media. Considering the value of your data, use the best quality media you can afford.

3. Run fsck(1M) first on EFS filesystems (if you are backing up an entire filesystem) to make sure you do not create a tape of a damaged filesystem. You must unmount a filesystem before checking it with *fsck*, so plan your backup schedule accordingly.

   This step is not necessary if you are backing up only a few files (for example, with *tar*).

4. The default tape device for any drives you may have is */dev/tape*. If you do not use the default device, you must specify a device in your backup command line.

5. Label your backups. If you plan to reuse the media, use pencil. Include the date, time, name of the system, the name of the utility, the exact command line used to make the backup (so you'll remember how to extract the files later), and a general indication of the contents. If more than one administrator performs backups at your site, include your name.

6. Verify the backup when you are finished. Some utilities (such as *dump* and *bru*) provide explicit options to verify a backup. With other programs, you can simply list the contents of the archive—this is usually sufficient to catch errors in the backup.

7. Write-protect your media after you make the backup.

8. Note the number of times you use each tape. It's sufficient to keep a running tally on the tape label.

See "Storing Backups" on page 9 for information on safely storing your backups.

System Backups

To make a backup of your system on any system with a graphical user interface, bring up the System menu on the System Toolchest and select the Backup and Restore menu item. From the Backup and Restore window, follow the prompts to perform your backup. A complete set of instructions for this procedure is available in the *Personal System Administration Guide*.

Backups made with the Backup and Restore window are the easiest to make and use, and are accessible from the Recover System option on the System Maintenance Menu if they are full system backups. When you make a full system backup, the command also makes a backup of the files in the disk volume header and saves the information in a file that is stored on tape. This file is used during system recovery to restore a damaged volume header.

To make a backup of your system using an IRIX command, use the Backup(1) command. Although the *Backup* command is a front-end interface to the bru(1) command, *Backup* also writes the disk volume header on the tape so that the Recover System option can reconstruct the boot blocks, which are not written to the tape using other backup commands. For more information, see the section “Backup and Restore” on page 23.

Recovery After System Corruption

If your root filesystem is damaged and your system cannot boot, you can restore your system from the Recover System option on the System Maintenance Menu. This is the menu that appears when you interrupt the boot sequence before the operating system takes over the system. To perform this recovery, you need two things:

- Access to a CD that contains the IRIX release on your system.
- A full system backup tape (beginning in the root directory (/) and containing all the files and directories on your system) created using the Backup and Restore window or the Backup command as described in the section “System Backups” on page 13.

If you do not have a full system backup made with the *Backup* command or Backup and Restore window—and your *root* or *usr* filesystems are so badly damaged that the operating system cannot boot—you have to reinstall your system software and then read your backup tapes (made with any backup tool you prefer) over the freshly installed software.

You may also be able to restore filesystems from the miniroot. For example, if your root filesystem has been corrupted, you may be able to boot the miniroot, unmount the root filesystem, and then use the miniroot versions of *restore*, *xfs_restore*, *Restore*, *bru*, *cpio*, or *tar* to restore your root filesystem. Refer to the following discussions of these commands for details on how to use them.

To recover from system corruption using the Recover System option on the System Maintenance Menu, follow these steps:

1. When you first start up your machine or press the Reset button on the system, this message appears:

   ```
   Starting up the system...
   ```

   Click the *Stop for Maintenance* button or press **<Esc>** to bring up the System Maintenance menu.

2. Click the Recover System icon in the System Maintenance menu, or type:

   4

   This System Recovery menu appears or you see a graphical equivalent:

   ```
                           System Recovery...

                   Press <Esc> to return to the menu.

   1) Remote Tape  2) Remote Directory  3) Local CD-ROM  4) Local Tape

   Enter 1-4 to select source type, <esc> to quit,
   or <enter> to start:
   ```

3. Enter the menu item number or click the appropriate drive icon for the IRIX release CD or software distribution directory you plan to use.

   Note: As of IRIX 6.2, the Remote Tape and Local Tape options on the System Recovery window are no longer usable because bootable (miniroot) software distribution tapes are no longer supported.

   - If you have a CD-ROM drive connected to your system, enter **3** or click the *Local CD-ROM* icon, then click *Accept* to start.

     You then see a notifier prompting you to insert the media into the drive. Insert the IRIX CD that came with your system, then click *Continue*.

   - If you don't have a CD-ROM drive, you can use a drive that is connected to another system on the network. At the System Recovery menu, enter **2** or click the *Remote Directory* icon.

When a notifier appears asking you for the remote hostname, type the system's name, a colon (**:**), and the full pathname of the CD-ROM drive, followed by **/dist**. For example, to access a CD-ROM drive on the system *mars*, you would type:

mars:/CDROM/dist

Click *Accept* on the notifier window, then click *Accept* on the System Recovery window.

On systems without graphics, you are prompted for the host as above, then you see this menu:

```
1) Remote Tape 2)[Remote Directory] 3) Local CD-ROM 4) Local Tape
      *a) Remote directory /CDROM/dist from server mars.

Enter 1-4 to select source type, a to select the source, <esc>
to quit,
or <enter> to start:
```

Press **<Enter>**.

- If you are using a remote software distribution directory, enter **2** or click the *Remote Directory* icon.

  When a notifier appears that asks you to enter the name of the remote host, type the system's name, a colon (**:**), and the full pathname of the software distribution directory. For example:

  mars:/dist/6.2

  Click *Accept* on the notifier window, then click *Accept* on the System Recovery window.

  On systems without graphics, you are prompted for the host as above, then you see this menu:

```
1) Remote Tape 2)[Remote Directory] 3) Local CD-ROM 4) Local Tape
      *a) Remote directory /dist/6.2 from server mars.

Enter 1-4 to select source type, a to select the source, <esc>
to quit,
or <enter> to start:
```

  Press **<Enter>**.

4. The system begins reading recovery and installation from the CD. It takes approximately five minutes to copy the information that it needs. After everything is copied from the CD or remote directory to the system disk you see messages including:

```
************************************************************
*                                                          *
*                    CRASH    RECOVERY                     *
*                                                          *
************************************************************

You may type  sh  to get a shell prompt at most questions

Checking for tape devices
```

   The next message asks for the location of the tape drive that you will use to read a system backup tape you created prior to the system crash using the Backup and Restore tool on the System menu of the System Toolchest or using the Backup(1) script.

5. If you have a local tape device, you see this message:

   `Restore will be from` *tapename*`.  OK? ([y]es, [n]o): [y]`

   tapename is the name of the local tape device. Answer **y** if this is the correct tape drive and **n** if is not.

6. If you have a remote (network) tape device, no tape device was found, or you answered "no" to the question in the previous step, you see this message:

   `Remote or local restore ([r]emote, [l]ocal): [l]`

   - If you answer "remote," you have chosen to restore from the network, and you are then asked to enter the following information: the hostname of the remote system, the name of the tape device on the remote system, the IP address of the remote system, and the IP address of your system. The IP address must consist of two to four numbers, separated by periods, such as 192.0.2.1
   - If you answer "local," you have chosen a tape device that is connected to your system, and you are then asked to enter the name of the tape device.

7. When you see the following message, insert your most recent full backup tape, then press **<Enter>**.

```
Insert the first Backup tape in the drive, then
press (<enter>, [q]uit (from recovery), [r]estart):
```

8. There is a pause while the program identifies the filesystems on the tape and attempts to mount those filesystems under */root*. Then you see this message:

   ```
   Erase all old filesystems and make new ones (y, n, sh): [n]
   ```

   You have three choices:

   - Answer **n** for no. After additional prompts confirming the filesystems to be read, the files on the tape are extracted. The version of each file on the tape replaces the version, if any, on the disk even if the version on the disk is newer.
   - Answer **y** for yes. After additional confirming prompts and prompts about filesystem types, the system erases all of the filesystems and copies everything from your backup tape to the disk.
   - Answer **sh** to escape to a shell. You are now in the miniroot environment and can investigate the damage to the system or attempt to save files that have been created or modified since the backup tape was created. After exiting the shell, you have the opportunity to remake filesystems and/or read the backup tape.
9. After reading the full backup tape, this prompt gives you the opportunity to read incremental backup tapes:

   ```
   Do you have incremental backup tapes to restore ([y]es, [n]o
   (none)): [n]
   ```

   Insert another tape and answer **y** if you have additional tape, answer **n** otherwise.
10. This prompt gives you the opportunity to reboot your system if recovery is complete, begin the crash recovery process again at the beginning, or re-read your first backup tape:

    ```
    Reboot, start over, or first tape again? ([r]eboot, [s]tart,
    [f]irst) [r]
    ```

    If you are ready to reboot, answer **r**, otherwise choose **start** or **first**.

Changing the Default Backup Device

At some point in the life of your workstation, you may choose to add a new storage media device. If you wish to change the default backup device to use your new hardware, the following instructions provide complete information. You can also use the graphical System Manager; it is the preferred tool for this operation and is described completely in the *Personal System Administration Guide*. Note, however, that no matter which method you use to select your preferred device, installing new system software or using the MAKEDEV(1M) command may reset the default Backup device. For more information on adding a storage media device, see "IRIX Admin: Peripheral Devices."

The method of changing the system default tape device is to relink */dev/nrtape* to the desired device. Use the following procedure:

1. Enter the commands:

   ```
   cd /dev
   ls -l nrtape
   ```

   You see something similar to this:

   ```
   crw-rw-rw-    2 root     sys      144,1251 Aug  4 01:29 nrtape
   ```

   The major device numbers is 144, and the minor device numbers is 65.

2. Examine the device numbers of all tape devices by entering the command:

   ```
   ls -l mt
   ```

   You see something similar to this:

   ```
   total 0
   crw-rw-rw-  2 root sys  144, 32 Mar 23 1993 tps0d4
   crw-rw-rw-  2 root sys  144, 33 Mar 23 1993 tps0d4nr
   crw-rw-rw-  2 root sys  144, 35 Mar 23 1993 tps0d4nrns
   crw-rw-rw-  2 root sys  144, 34 Mar 23 1993 tps0d4nrs
   crw-rw-rw-  3 root sys  144, 64 Mar 23 1993 tps0d2
   crw-rw-rw-  3 root sys  144, 65 Mar 23 1993 tps0d2nr
   ```

   The device at the bottom of this listing has the matching major and minor device numbers and therefore must be the device you're looking for. If more than one device has the correct major and minor numbers, then either device will do.

3. Remove the */dev/nrtape* link and create the new link with the same name. Use the commands:

   ```
   rm /dev/nrtape
   ln mt/tps0d4 /dev/nrtape
   ```

Repeat this procedure for the */dev/tape* device.

Most programs use */dev/nrtape* or */dev/tape* as the default tape device. If a program does not seem to be working correctly, first ensure that it is using the correct tape device.

bru

The *bru* utility provides a convenient single interface to perform functions that can only be accomplished using multiple utilities otherwise. For example, options to *bru* allow you to estimate the storage space a backup requires, compress the backup data, and examine backup archives. Refer to the bru(1) reference page for details on all *bru* defaults and options.

Be aware that although *bru* is available on a variety of UNIX systems, it is not as widely used as the other backup utilities. At a site that has workstations from a variety of vendors, not all of which provide *bru*, you may wish to use one of the other IRIX backup utilities for consistency. *tar* is the most widely accepted format. Refer to Table 1-1 for an overview of various backup utilities including *bru*. If your site has predominately IRIX workstations, and you don't need to move filesystem backups between different brands of computers, *bru* is probably a good choice.

Note that the System Manager backup menu uses the *Backup* interface to *bru* to back up workstations. If a *bru* backup is made that requires more than one tape, *bru* stops and prompts you to insert another tape before it continues.

Saving Data With bru

With *bru*, it is a simple matter to estimate the space a backup requires, back up filesystems or individual directories and files, and compress backups as described in this section.

Note: XFS and *bru*: The **-K** option has been added to the *bru* command for use with files larger than 2 GB. If the **-K** option is not used, *bru* skips any files that it cannot compress to less than 2 GB and issues a warning. Note that use of this option can create *bru* archives that are not usable on non-XFS systems. The **-K** option can only be used in combination with the **-Z** (use 12-bit LZW file compression) option.

Estimating Space Required for Backup

Use the **-e** option with *bru* for an estimate of how much space is required for an archive, for example:

```
bru –e /usr
```

returns how much space is required for the */usr* filesystem backup.

Backing Up a Filesystem With bru

The *bru* command is the shell command used by the System Manager to create backups. If you are using a server and do not have access to the graphical System Manager, use *bru* instead. Backups made with *bru* are readable by the System Maintenance Menu and Command Monitor. This command backs up the */usr* filesystem:

```
bru -c /usr
```

Backing Up Files With bru

To back up individual files with *bru*, enter:

```
bru -c files
```

You can specify one or more files. You can also read filenames from another file:

```
bru -c - < listfile
```

where the file *listfile* is a list of file names to be backed-up.

Saving Files by Modification Date

To save specific files that have changed since a particular time, you can use *bru* with the **-n** option. The following command backs up files on the */usr* filesystem that have been modified on or after November 26, 1990:

```
bru -c -n 26-Nov-90 /usr
```

Saving Files Using Data Compression

You can compress files as they are archived. Use the **-Z** flag:

```
bru -Z /usr
```

bru uses a 12-bit LZW file compression algorithm. Note that not all versions of *bru* support LZW compression. If you plan to transfer a *bru* archive to another vendor's workstation, make sure the other version of *bru* supports LZW data compression.

If you add the **-v** option, *bru* displays the compression ratio for each file (as a percentage). If you use **-t** and **-Z** to display the table of contents of an archive that contains compressed files, *bru* displays the current file names and compressed sizes, instead of the original filenames and sizes before creating the archive.

Incremental Backups With bru

You can use the incremental option *bru* to create incremental backups. For example:

1. Create a complete backup of the */usr* filesystem:

   ```
   bru -c
   ```

2. Each day, back up the files that have changed since the previous daily backup:

   ```
   bru -c -n 25-Nov-95 /usr
   bru -c -n 26-Nov-95 /usr
   bru -c -n 27-Nov-95 /usr
   bru -c -n 28-Nov-95 /usr
   bru -c -n 29-Nov-95 /usr
   bru -c -n 30-Nov-95 /usr
   bru -c -n 1-Dec-95 /usr
   ```

3. Every week, back up the files that have changed since the last weekly backup:

   ```
   bru -c -n 25-Nov-95 /usr
   ```

   Note that the dates listed in the command examples above are place holders. Use appropriate current dates in your command lines.

4. At the end of four weeks, perform a complete backup and start the process over.

This is a common incremental backup scheme.

Examining bru Archives

The "table of contents" flag, **-t**, displays the contents of a *bru* archive:

```
bru -t
```

You can combine this with the **-v** option for more information:

```
bru -tv
```

Use up to four "*v*" arguments for the most verbose output possible. Refer to bru(1) for more information.

Comparing Archived Files

You can compare files that are archived with the original files.

With *bru*, use the **-d** option. For example:

```
bru -d /usr
```

If you specify a single **-d**, *bru* reports when it discovers that a regular file's size or contents have changed since the archive was made.

If you use **-dd**, *bru* reports additional differences in modification dates, access modes, number of links for non-directory files, differences in the contents of symbolic links, owner IDs, and group IDs.

If you specify **-ddd**, *bru* reports additional differences in major and minor devices for special files and time of last access for regular files.

If you use **-dddd**, *bru* reports all differences except the time of the last status change, major and minor device numbers for non-special files, and size differences for directories. Usually, **-dddd** provides information that is meaningful only when verifying a full backup of a relatively static filesystem.

Inspecting an Archive for Consistency

The *bru* program provides an option, **-i**, to inspect an archive for internal consistency and data integrity. For example:

```
bru -i
```

If you add **-vv**, *bru* prints information from the archive header block:

```
bru -ivv
```

Neither *tar* nor *cpio* provides this sort of check. However, listing the contents of an archive is usually sufficient. Also, a reasonable check is to extract the files in the archive while sending the output to */dev/null*.

Restoring bru Archives

This section describes how to restore *bru* backups of filesystems and individual files.

Restoring a Filesystem With bru

Complete information on using the *bru* command and all its options is available in the bru(1) reference page. This command extracts the entire contents of a backup tape:

`bru -x`

Restoring Individual Files With bru

To restore an individual file, type:

`bru -x` *filename*

If the file already exists on the filesystem, *bru* compares its modification date with that of the copy on tape. If the version of the file in the filesystem is more recent than the one on tape, *bru* does not extract the archived file. Note that *filename* must exactly match what *bru -t* displays.

To overwrite a file no matter what the modification dates are, use the **-u** option. With **-u**, you must specify what kinds of files to overwrite:

- **b** for block special files
- **c** for character special files
- **d** for directories
- **l** for symbolic links
- **p** for fifos (named pipes)
- **r** for regular files

For example, to force updating of any regular files on the archive, enter:

`bru -xur`

Backup and Restore

The *Backup* and *Restore* utilities are front end interfaces to *bru*. They support remote hostname and tape device options, and *Backup* creates a volume header file listing that *Restore* uses for recovering the files and directories. For complete information, consult the Backup(1) and Restore(1) reference pages.

If you are planning to use the System Maintenance menu *Recovery* option, use *Backup* or the backup facility of the graphical System Manager, as those are the only formats accepted by the System Maintenance Menu. The System Manager is described in detail in the *Personal System Administration Guide*.

Estimating Space Required for Backup

Use the **-e** option with *bru* for an estimate of how much space is required for an archive, for example:

`bru -e /usr`

returns the amount of space required to back up the */usr* filesystem.

Saving Data with Backup

With *Backup*, you can back up files, directories, whole filesystems, and full systems on local or remote devices. Full system backups include the ability to recover a damaged volume header and also to back up only those files modified since a previous backup.

The syntax for the *Backup* command is:

`Backup` [**`-h`** *hostname*] [**`-t`** *device*] [**`-i`**] *directory_name* | *filename*

To back up an entire disk to the default tape device, enter:

`Backup /`

This *Backup* command archives the entire system. The current date is saved in the file */etc/lastbackup*.

Note: In order to use a *Backup* tape to restore your system from the System Maintenance Menu, you must make a full system backup. When you make a full system backup, the command also makes a backup of the names of the files in the disk volume header and saves the information in a file that is stored on tape. This file is used during system recovery to restore a damaged volume header.

You can make a backup relative to the last full system backup by entering:

`Backup -i /`

To back up a specific filesystem, enter the directory name of the filesystem. For example, to back up the *usr* filesystem, enter the following:

```
Backup /usr
```

To use a remote tape drive, use the **-h** *hostname* option:

```
Backup -h guest@alice.cbs.tv.com:/dev/tape /usr/people/ralph
```

This would back up the directory */usr/people/ralph* on the */dev/tape* device on the host alice.cbs.tv.com. You must have at least *guest* login privileges on the remote system in order to use a remote tape drive.

To back up a file, enter the filename. For example:

```
Backup people.tar.Z
```

Files (and directories) are stored relative to the current directory if the backup is made with a relative pathname as shown in this example. Relative pathnames are those that do not begin with a slash (/) character. Pathnames that begin with a slash are known as *absolute* pathnames. For example, */usr/bin/vi* is an absolute pathname. The leading slash indicates that the pathname begins at the root directory of the system. In contrast, *work/special.project/chapter1* is a relative pathname since the lack of a leading slash indicates that the path begins with a directory name in the current directory.

Restoring Backup Archives With Restore

The *Restore* command is a shell script that uses *bru* to extract files from a backup (see "bru" on page 19). You can also use *Restore* to read tapes made using the graphical System Manager (see the *Personal System Administration Guide*).

Restoring a Filesystem With Restore

You can recover multivolume backups with *Restore*. Enter:

```
Restore
```

and you are prompted to insert the tape into the drive.

To extract a single file, use this command:

Restore *file1*

With the **-h** option, you can specify the tape drive on a different host workstation:

`Restore –h guest@alice.cbs.tv.com` *file1*

You must have login privileges for the given account in order to extract data from a remote drive.

Files are restored into the current directory if the backup was made with relative pathnames. Relative pathnames are those that do not begin with a slash (/) character. Pathnames that begin with a slash are known as *absolute* pathnames. For example, */usr/bin/vi* is an absolute pathname. The leading slash indicates that the pathname begins at the root directory of the system. In contrast, *work/special.project/chapter1* is a relative pathname since the lack of a leading slash indicates that the path begins with a directory name in the current directory.

Existing files of the same pathname on the disk are overwritten during a restore operation even if they are more recent than the files on tape. You must be especially careful, then, if you are restoring files with absolute pathnames, because regardless of your current working directory, the file is restored where the pathname indicates.

For example, if the file you are restoring was backed up as */etc/passwd* and you are in the directory */tmp*, the file you restore overwrites the */etc/passwd* file. If the file you are restoring was backed-up as *passwd*, then restore the *passwd* file into */tmp*.

dump and restore

The *dump* and *restore* programs are standard filesystem backup utilities used on many UNIX systems. These commands are only used with EFS filesystems. Refer to "xfsdump and xfsrestore" on page 30 to dump and restore XFS filesystems. The *dump* program makes incremental backups of entire filesystems.

Use *restore* to retrieve files from a *dump* archive. With *restore*, you can restore an entire filesystem or specific files. It also has an interactive mode that lets you browse the contents of an archive, select specific files, and restore them.

Saving Data With dump

This section describes how to perform backups with the *dump* utility.

Saving a Filesystem With dump

The *dump* utility archives not only regular files, but also device files and special files such as links and named pipes. To recover files from an archive, you use the *restore* command.The date on which you last ran the *dump* program is stored in the file */etc/dumpdates* when you specify the **u** option to indicate an update.

This command backs up all files on the */usr* filesystem:

```
dump 0 /dev/usr
```

Note: The 0 in the example specifies the increment level. Refer to the next section for an explanation of level numbers.

Incremental Backups With dump

The *dump* utility is designed for incremental backups, and it archives not only regular files and directories, but also special files, links, and pipes.

To create an incremental backup, specify an increment number when you use *dump*. The *dump* program archives all files that have changed since the last appropriate increment and special files such as links and named pipes. To recover files from an archive, use the *restore* command.

The *dump* program is designed specifically to create incremental backups. It refers to the increments as *levels*, and each level is assigned a number:

- A level 0 backup archives all files in a filesystem.
- Backup levels 1–9 archive all files that have changed since the previous backup of the same or lesser level.

For example, this command backs up all files on the */usr* filesystem:

```
dump 0 /dev/usr
```

This command backs up those files that have changed since the previous level 0 dump:

```
dump 1 /dev/usr
```

This command archives those files that have changed since the previous level 1 dump:

```
dump 2 /dev/usr
```

If the next *dump* command you give specifies level 1, *dump* backs up the files that have changed since the last level 0, but not those that have changed since the last level 2.

This numbering system gives you enormous flexibility so you can create a backup schedule to fit your specific needs.

Restoring dump Archives With restore

This section describes how to use the *restore* command to restore files and filesystems backed up with the *dump* command.

Restoring a Filesystem With restore

Use *restore* to recover files and filesystems made with the *dump* program. There are two ways to use *restore*:

- interactively
- non-interactively

Use the interactive option to recover moderate numbers of files from a *dump* archive. With the interactive feature of *restore*, you can browse the contents of a tape to locate and extract specific files.

Use the non-interactive mode to recover an entire backup. For example, place the backup in the drive and enter:

```
restore –x
```

If your root filesystem is damaged and needs to be completely restored, you should probably reinstall the system, then rebuild it by extracting selected files from backup tapes. You can also restore the root filesystem by booting the miniroot, unmounting the root filesystem, and then using *restore* in the miniroot to restore the root filesystem.

Restoring Individual Files With restore

To recover individual files from a *dump* archive, follow these steps:

1. Place the tape in the tape drive. Make sure it is write-protected.
2. Enter:

   ```
   restore vi
   ```

You see something like this:

```
Verify tape and initialize maps
Tape block size is 32
Dump  date: Wed Feb 13 10:18:59 1991
Dumped from: the epoch
Level 0 dump of an unlisted filesystem on ralph:/dev/rusr
Label: none
Extract directories from tape
Initialize symbol table.
restore >>
```

3. You are now at the *restore*>> prompt. You can browse the tape with *cd* and *ls*:

   ```
   restore > ls
   ```

   You see something like this:

   ```
   2      *./          973     source        1502  net/
   2      *../         149     d2/           1445  os/
   10     .cshrc       155016  debug/        1437  proto3.5/
   1463   .gamma       69899   dev/          1494  revE
   1464   .gamtables   696     etc/          2122  stand/
   160    .kshrc       137     bin/          3     tmp/
   1540   .lastlogin   1311412 jake/         128   unix
   819    .login       424     lib/          128   unix.debug
   820    .profile     9       lost+found/   4     usr/
   ```

   To continue browsing, enter the following commands to the *restore*>> prompt:

   ```
   restore >> cd etc
   ```

   ```
   restore >> pwd
   ```

   ```
   /etc
   ```

4. Start building a list of files that you want to extract. Use the *add* command to add the names of the files you want to the extract list:

   ```
   restore >> add fstab
   ```

   ```
   restore >> add fsck
   ```

   If you enter *ls* at this point, you see a list of files, and *fsck* and *fstab* are marked with an asterisk to show they will be extracted.

   If you want to remove a file from the list of those to be extracted, use the *delete* command:

   ```
   restore > delete fstab
   ```

5. To restore the specified files, use the *extract* command:

```
restore > extract
Extract requested files
You have not read any tapes yet.
Unless you know which volume your file(s) are on you should
start with the last volume and work towards the first.
Specify next volume #: 1
Mount tape volume 1
then enter tape name (default: /dev/tape) <Return>
extract file ./etc/fsck
Add links
Set directory mode, owner, and times.
set owner/mode for '.'? [yn] n
restore > q
```

To recover only a few files, you may wish to use the non-interactive options of *restore*. For example, enter:

restore –x ./usr/people/ralph/bus.schedule ./etc/passwd

This recovers the files *bus.schedule* and *passwd* from the archive.

xfsdump and xfsrestore

This section describes how the *xfsdump* and *xfsrestore* utilities work and how to use them to back up and recover data on XFS filesystems. (The xfsdump(1M) and xfsrestore(1M) reference pages provide online information on these utilities.) Table 2-1 and Table 2-2 summarize when to use *xfsdump* and *xfsrestore* and when to use their EFS counterparts, dump(1M) and restore(1M).

Table 2-1 Filesystems and Dump Utilities

For a Filesystem of Type	Dump It Using
EFS	*dump*
XFS	*xfsdump*

Table 2-2 Filesystems and Restore Utilities

For a Dump Made Using	Restore It Using	On a Filesystem of Type
dump	*restore*	EFS or XFS
xfsdump	*xfsrestore*	EFS or XFS

Note than you can restore data in either EFS or XFS filesystems, but must use the restore utility that corresponds with the dump utility used to make the backup.

Features of xfsdump and xfsrestore

The *xfsdump* and *xfsrestore* utilities fully support XFS filesystems. With *xfsdump* and *xfsrestore*, you can back up and restore data using local or remote drives. You can back up filesystems, directories, and/or individual files, and then restore filesystems, directories, and files independently of how they were backed up. *xfsdump* also allows you to back up "live" (mounted, in-use) filesystems.

With *xfsdump* and *xfsrestore*, you can recover from intentional or accidental interruptions—this means you can interrupt a dump or restore at any time, and then resume it whenever desired. With *xfsrestore*, you can restore *xfsdump* data onto EFS filesystems. *(xfsdump* backs up mounted XFS filesystems only.) *xfsdump* and *xfsrestore* support incremental dumps, and multiple dumps can be placed on a single media object. The utilities can automatically divide a dump among multiple drives, and can restore a dump from multiple drives. This allows you to perform faster dumps and restores.

xfsdump and *xfsrestore* support XFS features including 64-bit inode numbers, file lengths, holes, and user-selectable extent sizes. They support multiple media types, all IRIX-supported file types (regular, directory, symbolic link, block and character special, FIFO, and socket), and retain hard links. *xfsdump* does not affect the state of the filesystem being dumped (for example, access times are retained). *xfsrestore* detects and bypasses media errors and recovers rapidly after encountering them. *xfsdump* does not cross mount points, local or remote.

xfsdump optionally prompts for additional media when the end of the current media is reached. Operator estimates of media capacity are not required and *xfsdump* also supports automated backups. *xfsdump* maintains an extensive online inventory of all dumps performed. Inventory contents can be viewed through various filters to quickly locate specific dump information. *xfsrestore* supports interactive operation, allowing selection of individual files or directories for recovery. It also permits selection from among backups performed at different times when multiple dumps are available. Dump contents may also be viewed noninteractively.

Media Layout

The following section introduces some terminology and then describes the way *xfsdump* formats data on the storage media for use by *xfsrestore*.

Terminology

This section introduces terminology used in the rest of this chapter.

While *xfsdump* and *xfsrestore* are often used with tape media, the utilities actually support multiple kinds of media, so in the following discussions, the term *media object* is used to refer to the media in a generic fashion. The term *dump* refers to the result of a single use of the *xfsdump* command to output data files to the selected media object(s). An instance of the use of *xfsdump* is referred to as a *dump session*.

The dump session sends a single *dump stream* to the media object(s). The dump stream may contain as little as a single file or as much as an entire filesystem. The dump stream is composed of *dump objects*, which are:

- one or more *data segments*
- an optional *dump inventory*
- a *stream terminator*

The data segment(s) contains the actual data, the dump inventory contains a list of the dump objects in the dump, and the stream terminator marks the end of the dump stream. When a dump stream is composed of multiple dump objects, each object is contained in a *media file*. Some output devices, for example standard output, do not support the concept of media files—the dump stream is only the data.

Possible Dump Layouts

The simplest dump, for example the dump of a small amount of data to a single tape, produces a data segment and a stream terminator as the only dump objects. If the optional inventory object is added, you have a dump such as that illustrated in Figure 2-1. (In the data layout diagrams in this section, the optional inventory object is always included.)

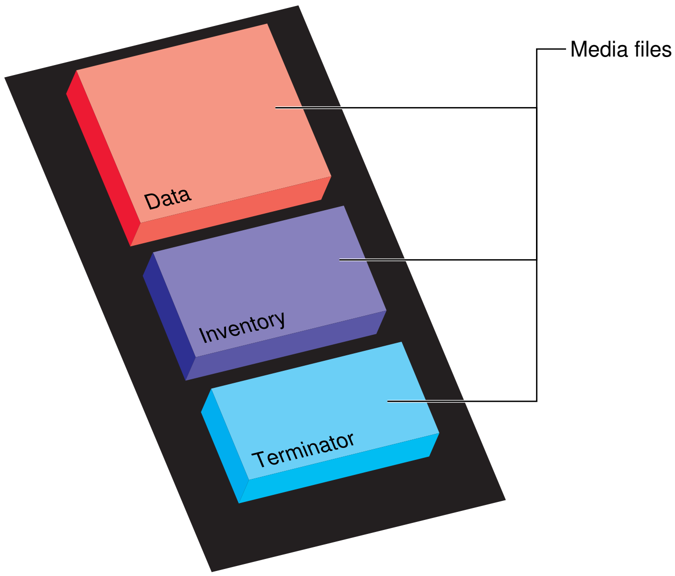

Figure 2-1 Single Dump on Single Media Object

You can also dump data streams that are larger than a single media object. The data stream can be broken between any two media files including data segment boundaries. (The inventory is never broken into segments.) In addition, if multiple drives are specified, the dump is automatically broken into multiple streams. The *xfsdump* utility prompts for a new media object when the end of the current media object is reached. Figure 2-2 illustrates the data layout of a single dump session that requires two media objects on each of two devices.

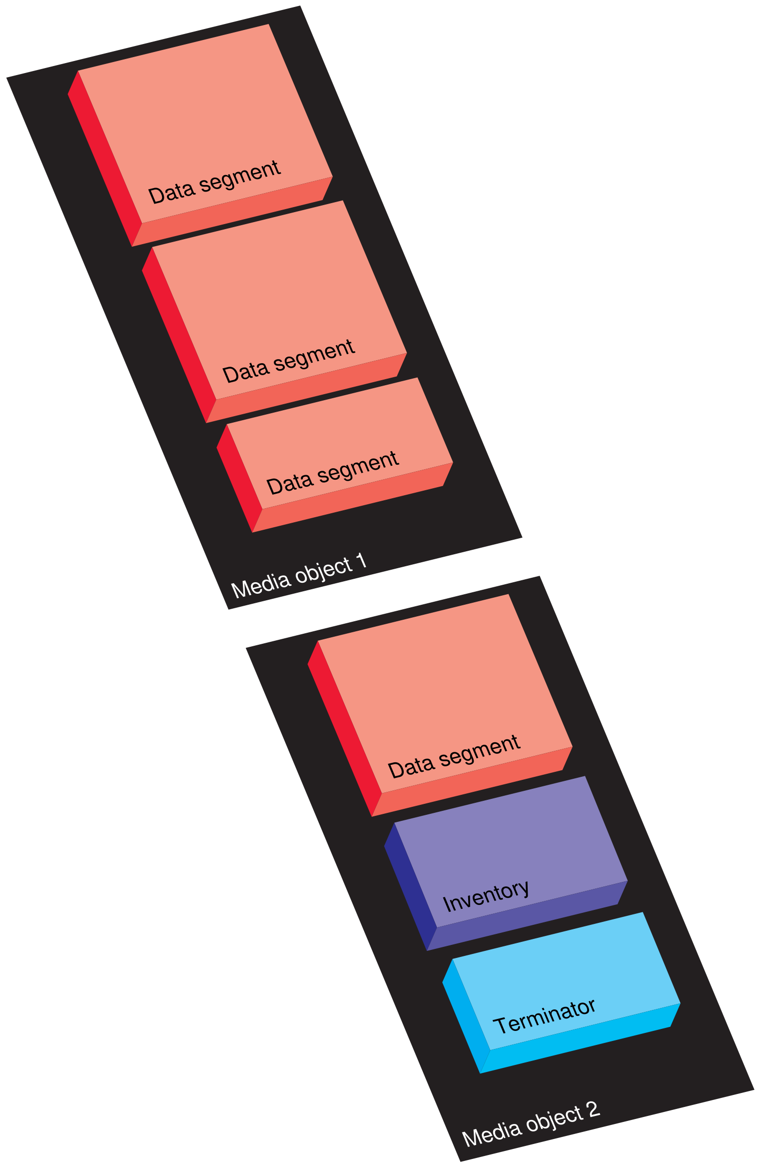

Figure 2-2 Single Dump on Multiple Media Objects

The *xfsdump* utility also accommodates multiple dumps on a single media object. When dumping to tape, for example, the tape is automatically advanced past the existing dump session(s) and the existing stream terminator is erased. The new dump data is then written, followed by the new stream terminator[1]. Figure 2-3 illustrates the layout of media files for two dumps on a single media object.

[1] For drives that do not permit termination to operate in this way, other means are used to achieve the same effective result.

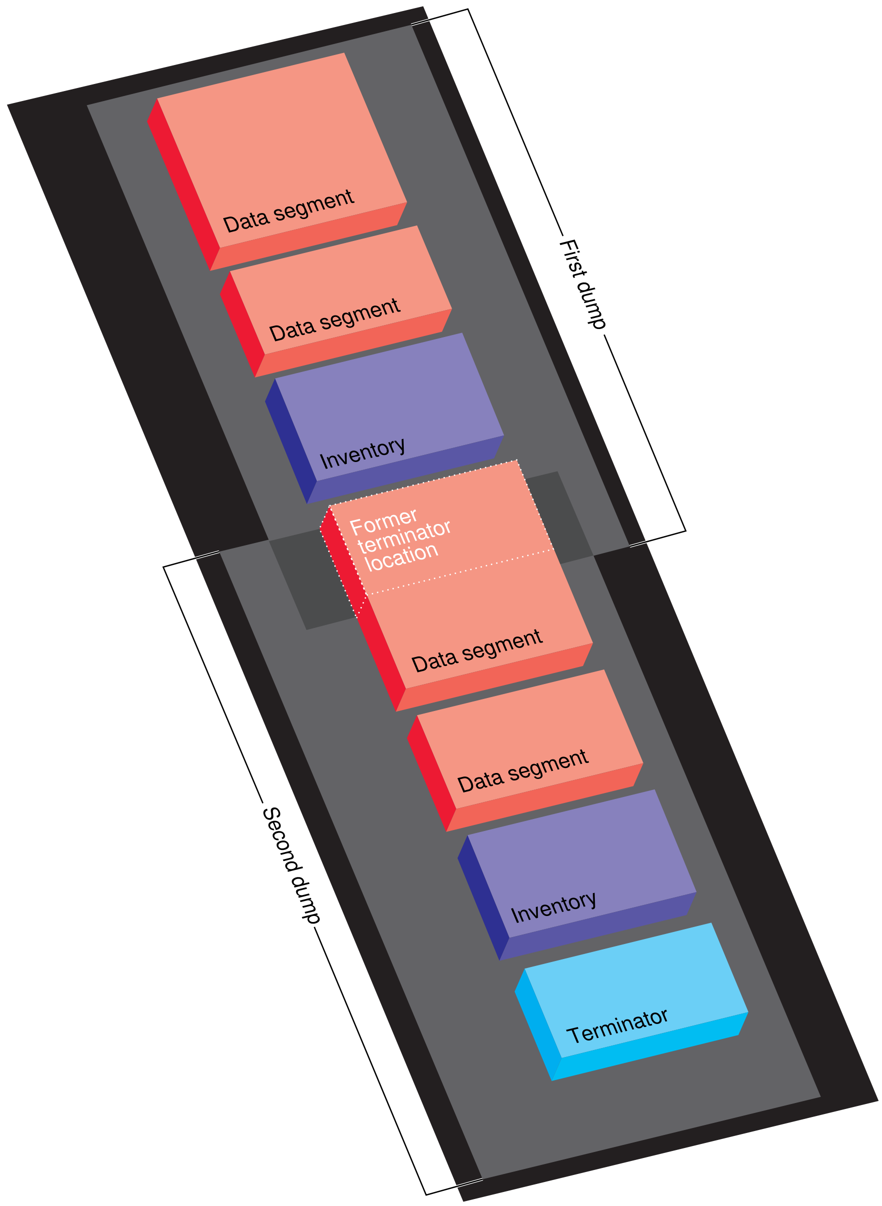

Figure 2-3 Multiple Dumps on Single Media Object

Figure 2-4 illustrates a case in which multiple dumps use multiple media objects. If media files already exist on the additional media object(s), the *xfsdump* utility finds the existing stream terminator, erases it, and begins writing the new dump data stream.

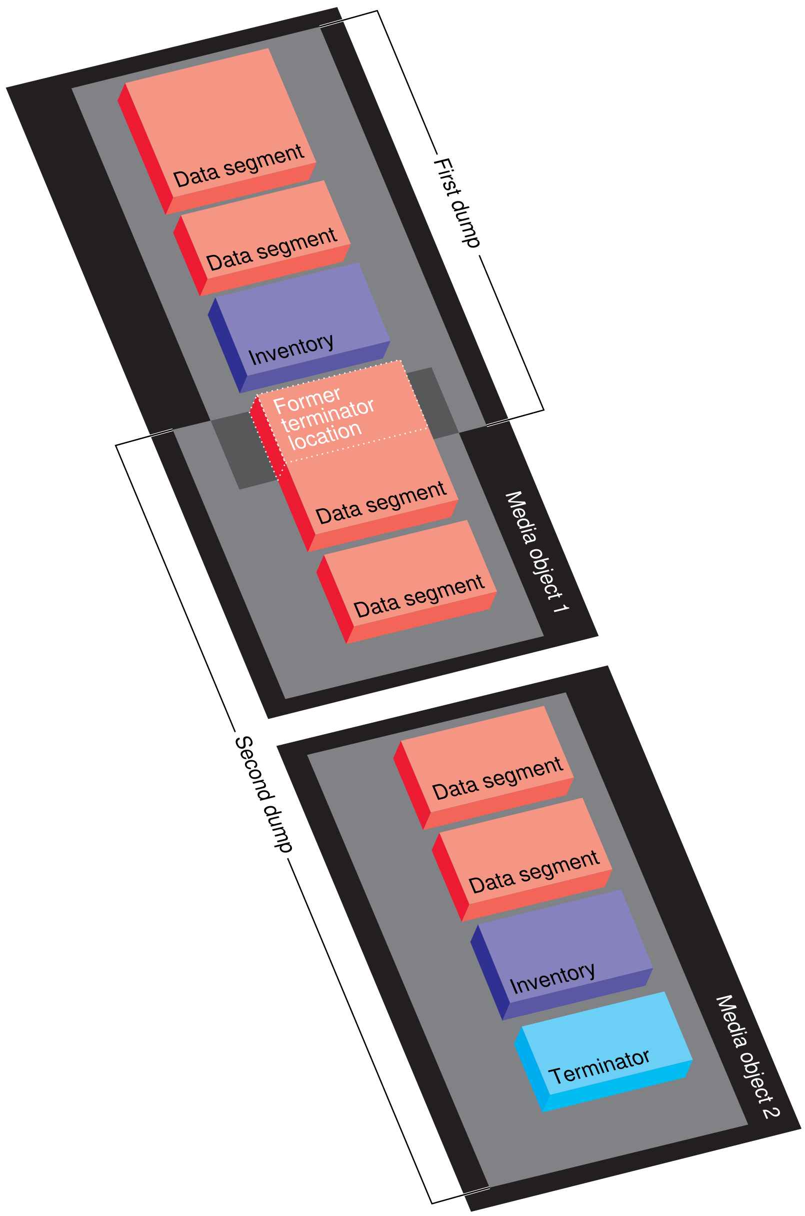

Figure 2-4 Multiple Dumps on Multiple Media Objects

Saving Data With xfsdump

This section discusses how to use the *xfsdump* command to back up data to local and remote devices. You can get a summary of *xfsdump* syntax with the **–h** option:

```
# xfsdump -h
xfsdump: version X.X
xfsdump: usage: xfsdump [ -f <destination> ... ]
               [ -h (help) ]
               [ -l <level> ]
               [ -p <seconds between progress reports> ]
               [ -s <subtree> ... ]
               [ -v <verbosity {silent, verbose, trace}> ]
               [ -A (don't dump extended file attributes) ]
               [ -B <base dump session id> ]
               [ -E (pre-erase media) ]
               [ -F (don't prompt) ]
               [ -I (display dump inventory) ]
               [ -J (inhibit inventory update) ]
               [ -L <session label> ]
               [ -M <media label> ... ]
               [ -O <options file> ]
               [ -R (resume) ]
               [ -T (don't timeout dialogs) ]
               [ -Y <I/O buffer ring length> ]
               [ - (stdout) ]
               [ <source (mntpnt|device)> ]
```

You must be the superuser to use *xfsdump*. Refer to the xfsdump(1M) reference page for details.

Specifying Local Media

You can use *xfsdump* to back up data to various media. For example, you can dump data to a tape or hard disk. The drive containing the media object may be connected to the local system or accessible over the network.

Following is an example of a level 0 dump to a local tape drive. Note that dump level does not need to be specified for a level 0 dump. (Refer to "Incremental Backups With dump" on page 27 for a discussion of dump levels.)

```
# xfsdump -f /dev/tape -L testers_11_21_94 -M test_1 /disk2
xfsdump: version 2.0 - type ^C for status and control
xfsdump: level 0 dump of cumulus:/disk2
xfsdump: dump date: Wed Oct 25 16:19:13 1995
xfsdump: session id: d2a6123b-b21d-1001-8938-08006906dc5c
xfsdump: session label: “testers_11_21_94”
xfsdump: ino map phase 1: skipping (no subtrees specified)
xfsdump: ino map phase 2: constructing initial dump list
xfsdump: ino map phase 3: skipping (no pruning necessary)
xfsdump: ino map phase 4: skipping (size estimated in phase 2)
xfsdump: ino map phase 5: skipping (only one dump stream)
xfsdump: ino map construction complete
xfsdump: preparing drive
xfsdump: creating dump session media file 0 (media 0, file 0)
xfsdump: dumping ino map
xfsdump: dumping directories
xfsdump: dumping non-directory files
xfsdump: ending media file
xfsdump: media file size 16777216 bytes
xfsdump: dumping session inventory
xfsdump: beginning inventory media file
xfsdump: media file 1 (media 0, file 1)
xfsdump: ending inventory media file
xfsdump: inventory media file size 4194304 bytes
xfsdump: writing stream terminator
xfsdump: beginning media stream terminator
xfsdump: media file 2 (media 0, file 2)
xfsdump: ending media stream terminator
xfsdump: media stream terminator size 2097152 bytes
xfsdump: I/O metrics: 3 by 2MB ring; 14/22 (64%) records streamed; 145889B/s
xfsdump: dump complete: 141 seconds elapsed
```

In this case, a session label (**–L** option) and a media label (**–M** option) are supplied, and the entire filesystem is dumped. Since no verbosity option is supplied, the default of *verbose* is used, resulting in the detailed screen output. The dump inventory is updated with the record of this backup because the **-J** option is not specified.

Following is an example of a backup of a subdirectory of a filesystem. In this example, the verbosity is set to *silent*, and the dump inventory is not updated (**–J** option):

```
# xfsdump -f /dev/tape -v silent -J -s people/fred /usr
```

Note that the subdirectory backed up (*/usr/people/fred*) was specified relative to the filesystem, so the specification did not include the name of the filesystem (in this case, */usr*). Since */usr* may be a very large filesystem and the **-v silent** option was used, this could take a long time during which there would be no screen output.

Specifying a Remote Tape Drive

To back up data to a remote tape drive, use the standard remote system syntax, specifying the system (by hostname if supported by a name server or IP address if not) followed by a colon (:), then the pathname of the special file.

Note: For remote backups, use the variable block size tape device if the device supports variable block size operation; otherwise, use the fixed block size device (see intro(7)).

The following example shows a subtree backup to a remote tape device:

```
# xfsdump -f magnolia:/dev/tape -L mag_10-95 -s engr /disk2
xfsdump: version 2.0 - type ^C for status and control
xfsdump: level 0 dump of cumulus:/disk2
xfsdump: dump date: Wed Oct 25 16:27:39 1995
xfsdump: session id: d2a6124b-b21d-1001-8938-08006906dc5c
xfsdump: session label: "mag_10-95"
xfsdump: ino map phase 1: parsing subtree selections
xfsdump: ino map phase 2: constructing initial dump list
xfsdump: ino map phase 3: pruning unneeded subtrees
xfsdump: ino map phase 4: estimating dump size
xfsdump: ino map phase 5: skipping (only one dump stream)
xfsdump: ino map construction complete
xfsdump: preparing drive
xfsdump: positioned at media file 0: dump 0, stream 0
xfsdump: positioned at media file 1: dump 0, stream 0
xfsdump: positioned at media file 2: dump 0, stream 0
xfsdump: stream terminator found
xfsdump: creating dump session media file 0 (media 0, file 2)
xfsdump: dumping ino map
xfsdump: dumping directories
xfsdump: dumping non-directory files
xfsdump: ending media file
xfsdump: media file size 6291456 bytes
xfsdump: dumping session inventory
xfsdump: beginning inventory media file
xfsdump: media file 1 (media 0, file 3)
xfsdump: ending inventory media file
```

```
xfsdump: inventory media file size 4194304 bytes
xfsdump: writing stream terminator
xfsdump: beginning media stream terminator
xfsdump: media file 2 (media 0, file 4)
xfsdump: ending media stream terminator
xfsdump: media stream terminator size 2097152 bytes
xfsdump: I/O metrics: 3 by 2MB ring; 12/22 (55%) records streamed; 99864B/s
xfsdump: dump complete: 149 seconds elapsed
```

In this case, */disk2/engr* is backed up to the variable block size tape device on the remote system *magnolia*. Existing dumps on the tape mounted on magnolia were skipped before recording the new data.

Note: The superuser account on the local system must be able to *rsh* to the remote system without a password. For more information, see hosts.equiv(4).

Backing Up to a File

You can back up data to a file instead of a device. In the following example, a file (*Makefile*) and a directory (*Source*) are backed up to a dump file (*monday_backup*) in */usr/tmp* on the local system:

```
# xfsdump -f /usr/tmp/monday_backup -v silent -J -s \
people/fred/Makefile -s people/fred/Source /usr
```

You may also dump to a file on a remote system, but note that the file must be in the remote system's */dev* directory. For example, the following command backs up the */usr/people/fred* subdirectory on the local system to the regular file */dev/fred_mon_12-2* on the remote system *theduke*:

```
# xfsdump -f theduke:/dev/fred_mon_12-2 -s people/fred /usr
```

Alternatively, you could dump to any remote file if that file is on an NFS-mounted filesystem. In any case, permission settings on the remote system must allow you to write to the file.

Refer to the section "Using xfsdump and xfsrestore to Copy Filesystems" on page 62 for information on using the standard input and standard output capabilities of *xfsdump* and *xfsrestore* to pipe data between filesystems or across the network.

Reusing Tapes

When you use a new tape as the media object of a dump session, *xfsdump* begins writing dump data at the beginning of the tape without prompting. If the tape already has dump data on it, *xfsdump* begins writing data after the last dump stream, again without prompting.

If, however, the tape contains data that is not from a dump session, *xfsdump* prompts you before continuing:

```
# xfsdump -f /dev/tape /test
xfsdump: version X.X - type ^C for status and control
xfsdump: dump date: Fri Dec 2 11:25:19 1994
xfsdump: level 0 dump
xfsdump: session id: d23cc072-b21d-1001-8f97-080069068eeb
xfsdump: preparing tape drive
xfsdump: this tape contains data that is not part of an XFS dump
xfsdump: do you want to overwrite this tape?
type y to overwrite, n to change tapes or abort (y/n):
```

You must answer **y** if you want to continue with the dump session, or **n** to quit. If you answer **y**, the dump session resumes and the tape is overwritten. If you do not respond to the prompt, the session eventually times out. Note that this means that an automatic backup, for example one initiated by a *crontab* entry, will not succeed—unless you specified the **-F** option with the *xfsdump* command, which forces it to overwrite the tape rather than prompt for approval.

Erasing Used Tapes

Erase preexisting data on tapes with the *mt erase* command. Make sure the tape is not write-protected.

For example, to prepare a used tape in the local default tape drive, enter:

```
# mt -f /dev/tape erase
```

Caution: This erases all data on the tape, including any dump sessions.

The tape can now be used by *xfsdump* without prompting for approval.

Incremental and Resumed Dumps

Incremental dumps are a way of backing up less data at a time but still preserving current versions of all your backed-up files, directories, and so on. Incremental backups are organized numerically by levels from 0 through 9. A level 0 dump always backs up the complete filesystem. A dump level of any other number backs up all files that have changed since a dump with a lower dump level number.

For example, if you perform a level 2 backup on a filesystem one day and your next dump is a level 3 backup, only those files that have changed since the level 2 backup are dumped with the level 3 backup. In this case, the level 2 backup is called the *base dump* for the level 3 backup. The base dump is the most recent backup of that filesystem with a lower dump level number.

Resumed dumps work in much the same way. When a dump is resumed after it has been interrupted, the remaining files that had been scheduled to be backed up during the interrupted dump session are backed up, and any files that changed during the interruption are also backed up. Note that you must restore an interrupted dump as if it is an incremental dump (see “Cumulative Restores With xfsrestore” on page 57).

In the following example, a new tape is used and the level 0 dump is the first dump written to it:

```
# xfsdump -f /dev/tape -l 0 -M Jun_94 -L week_1 -v silent /usr
```

A week later, a level 1 dump of the filesystem is performed on the same tape:

```
# xfsdump -f /dev/tape -l 1 -L week_2 /usr
```

The tape is forwarded past the existing dump data and the new data from the level 1 dump is written after it. (Note that it is not necessary to specify the media label for each successive dump on a media object.)

A week later, a level 2 dump is taken:

```
# xfsdump -f /dev/tape -l 2 -L week_3 /usr
```

and so on, for the four weeks of a month in this example, the fourth week being a level 3 dump (up to nine dump levels are supported). Refer to “Cumulative Restores With xfsrestore” on page 57 for information on the proper procedure for restoring incremental dumps.

You can interrupt a dump session and resume it later. To interrupt a dump session, type the interrupt character (typically **<CTRL-C>**). You receive a list of options which allow you to interrupt the session, change verbosity level, or resume the session.

In the following example, *xfsdump* is interrupted after dumping approximately 37% of a filesystem:

```
# xfsdump -f /dev/tape -M march95 -L week_1 -v silent /disk2

 ========================= status and control dialog ==========================

status at 16:49:16: 378/910 files dumped, 37.8% complete, 32 seconds elapsed

please select one of the following operations
1: interrupt this session
2: change verbosity
3: display metrics
4: other controls
5: continue (default) (timeout in 60 sec)
 -> 1

please confirm
1: interrupt this session
2: continue (default) (timeout in 60 sec)
 -> 1
interrupt request accepted

 --------------------------------- end dialog ---------------------------------

xfsdump: initiating session interrupt
xfsdump: dump interrupted prior to ino 1053172 offset 0
```

You can later continue the dump by including the **–R** option and a different session label:

```
# xfsdum -f /dev/tape -R -L week_1.contd -v silent /disk2p
```

Any files that were not backed up before the interruption, and any file changes that were made during the interruption, are backed up after the dump is resumed.

Note: Use of the **–R** option requires that the dump was made with a dump inventory taken, that is, the **–J** option was not used with *xfsdump*.

Examining xfsdump Archives

This section describes how to use the *xfsdump* command to view an *xfsdump* inventory.

The *xfsdump* inventory is maintained in the directory */var/xfsdump* created by *xfsdump*. You can view the dump inventory at any time with the *xfsdump –I* command. With no other arguments, *xfsdump –I* displays the entire dump inventory. (The *xfsdump -I* command does not require root privileges.)

The following output presents a section of a dump inventory.

```
# xfsdump -I | more
file system 0:
        fs id:          d23cb450-b21d-1001-8f97-080069068eeb
        session 0:
                mount point:    magnolia.abc.xyz.com:/test
                device:         magnolia.abc.xyz.com:/dev/rdsk/dks0d3s2
                time:           Mon Nov 28 11:44:04 1994
                session label:  ""
                session id:     d23cbf44-b21d-1001-8f97-080069068eeb
                level:          0
                resumed:        NO
                subtree:        NO
                streams:        1
                stream 0:
                        pathname:       /dev/tape
                        start:          ino 4121 offset 0
                        end:            ino 0 offset 0
                        interrupted:    YES
                        media files:    2
                        media file 0:
                                mfile index:    0
---more---
```

Notice that the dump inventory records are presented sequentially and are indented to illustrate the hierarchical order of the dump information.

You can view a subset of the dump inventory by specifying the level of depth (1, 2, or 3) that you want to view. For example, specifying depth=2 filters out a lot of the specific dump information as you can see by comparing the previous output with this:

```
# xfsdump -I depth=2
file system 0:
        fs id:          d23cb450-b21d-1001-8f97-080069068eeb
        session 0:
                mount point:    magnolia.abc.xyz.com:/test
                device:         magnolia.abc.xyz.com:/dev/rdsk/dks0d3s2
                time:           Mon Nov 28 11:44:04 1994
                session label:  ""
                session id:     d23cbf44-b21d-1001-8f97-080069068eeb
                level:          0
                resumed:        NO
                subtree:        NO
                streams:        1
        session 1:
                mount point:    magnolia.abc.xyz.com:/test
                device:         magnolia.abc.xyz.com:/dev/rdsk/dks0d3s2
               .
               .
               .
```

You can also view a filesystem-specific inventory by specifying the filesystem mount point with the *mnt* option. The following output shows an example of a dump inventory display in which the *depth* is set to *1,* and only a single filesystem is displayed:

```
# xfsdump -I depth=1,mnt=magnolia.abc.xyz.com:/test
filesystem 0:
        fs id:          d23cb450-b21d-1001-8f97-080069068eeb
```

Note that you can also look at a list of contents on the dump media itself by using the **–t** option with *xfsrestore*. (The *xfsrestore* utility is discussed in detail in the following section.) For example, to list the contents of the dump tape currently in the local tape drive, type:

```
# xfsrestore -f /dev/tape -t -v silent | more
xfsrestore: dump session found
xfsrestore: session label: "week_1"
xfsrestore: session id: d23cbcb4-b21d-1001-8f97-080069068eeb
xfsrestore: no media label
xfsrestore: media id: d23cbcb5-b21d-1001-8f97-080069068eeb
do you want to select this dump? (y/n): y
selected
one
A/five
people/fred/TOC
people/fred/ch3.doc
people/fred/ch3TOC.doc
people/fred/questions
A/four
people/fred/script_0
people/fred/script_1
people/fred/script_2
people/fred/script_3
people/fred/sub1/TOC
people/fred/sub1/ch3.doc
people/fred/sub1/ch3TOC.doc
people/fred/sub1/questions
people/fred/sub1/script_0
people/fred/sub1/script_1
people/fred/sub1/script_2
people/fred/sub1/script_3
people/fred/sub1/xdump1.doc
people/fred/sub1/xdump1.doc.backup
people/fred/sub1/xfsdump.doc
people/fred/sub1/xfsdump.doc.auto
people/fred/sub1/sub2/TOC
---more---
```

Restoring xfsdump Archives With xfsrestore

This section discusses the *xfsrestore* command, which you must use to view and extract data from the dump data created by *xfsdump*. You can get a summary of *xfsrestore* syntax with the **–h** option:

```
# xfsrestore -h
xfsrestore: version X.X
xfsrestore: usage: xfsrestore [ -a <alt. workspace dir> ... ]
                    [ -e (don't overwrite existing files) ]
                    [ -f <source> ... ]
                    [ -h (help) ]
                    [ -i (interactive) ]
                    [ -n <file> (restore only if newer than) ]
                    [ -o (restore owner/group even if not root) ]
                    [ -p <seconds between progress reports> ]
                    [ -r (cumulative restore) ]
                    [ -s <subtree> ... ]
                    [ -t (contents only) ]
                    [ -v <verbosity {silent, verbose, trace}> ]
                    [ -A (don't restore extended file attributes) ]
                    [ -C (check tape record checksums) ]
                    [ -D (restore DMAPI event settings) ]
                    [ -E (don't overwrite if changed) ]
                    [ -F (don't prompt) ]
                    [ -I (display dump inventory) ]
                    [ -J (inhibit inventory update) ]
                    [ -L <session label> ]
                    [ -N (timestamp messages) ]
                    [ -O <options file> ]
                    [ -P (pin down I/O buffers) ]
                    [ -Q (force interrupted session completion) ]
                    [ -R (resume) ]
                    [ -S <session id> ]
                    [ -T (don't timeout dialogs) ]
                    [ -U (unload media when change needed) ]
                    [ -V (show subsystem in messages) ]
                    [ -W (show verbosity in messages) ]
                    [ -X <excluded subtree> ... ]
                    [ -Y <I/O buffer ring length> ]
                    [ -Z (miniroot restrictions) ]
                    [ - (stdin) ]
                    [ <destination> ]
```

Use *xfsrestore* to restore data backed up with *xfsdump*. You can restore files, subdirectories, and filesystems—regardless of the way they were backed up. For example, if you back up an entire filesystem in a single dump, you can select individual files and subdirectories from within that filesystem to restore.

You can use *xfsrestore* interactively or noninteractively. With interactive mode, you can peruse the filesystem or files backed up, selecting those you want to restore. In noninteractive operation, a single command line can restore selected files and subdirectories, or an entire filesystem. You can restore data to its original filesystem location or any other location in an EFS or XFS filesystem.

By using successive invocations of *xfsrestore*, you can restore incremental dumps on a base dump. This restores data in the same sequence it was dumped.

Simple Restores With xfsrestore

A simple restore is a non-cumulative restore (for information on restoring incremental dumps, refer to "Cumulative Restores With xfsrestore" on page 57). An example of a simple, noninteractive use of *xfsrestore* is:

```
# xfsrestore -f /dev/tape /disk2
xfsrestore: version 2.0 - type ^C for status and control
xfsrestore: searching media for dump
xfsrestore: preparing drive
xfsrestore: examining media file 0

 =========================== dump selection dialog ============================

the following dump has been found on drive 0

hostname: cumulus
mount point: /disk2
volume: /dev/rdsk/dks0d2s0
session time: Wed Oct 25 16:59:00 1995
level: 0
session label: "tape1"
media label: "media1"
file system id: d2a602fc-b21d-1001-8938-08006906dc5c
session id: d2a61284-b21d-1001-8938-08006906dc5c
media id: d2a61285-b21d-1001-8938-08006906dc5c

restore this dump?
1: skip
2: restore (default)
 -> 2
this dump selected for restoral

 --------------------------------- end dialog ---------------------------------

xfsrestore: using online session inventory
xfsrestore: searching media for directory dump
xfsrestore: reading directories
xfsrestore: directory post-processing
xfsrestore: restoring non-directory files
xfsrestore: I/O metrics: 3 by 2MB ring; 9/13 (69%) records streamed; 204600B/s
xfsrestore: restore complete: 104 seconds elapsed
```

In this case, *xfsrestore* went to the first dump on the tape and asked if this was the dump to restore. If you had entered **1** for "skip," *xfsrestore* would have proceeded to the next dump on the tape (if there was one) and asked if this was the dump you wanted to restore.

You can request a specific dump if you used *xfsdump* with a session label. For example:

```
# xfsrestore -f /dev/tape -L Wed_11_23 /usr
xfsrestore: version X.X - type ^C for status and control
xfsrestore: preparing tape drive
xfsrestore: dump session found
xfsrestore: advancing tape to next media file
xfsrestore: dump session found
xfsrestore: restore of level 0 dump of magnolia.abc.xyz.com:/usr created Wed Nov
23 11:17:54 1994
xfsrestore: beginning media file
xfsrestore: reading ino map
xfsrestore: initializing the map tree
xfsrestore: reading the directory hierarchy
xfsrestore: restoring non-directory files
xfsrestore: ending media file
xfsrestore: restoring directory attributes
xfsrestore: restore complete: 200 seconds elapsed
```

In this way you recover a dump with a single command line and do not have to answer **y** or **n** to the prompt(s) asking you if the dump session found is the correct one. To be even more exact, use the **-S** option and specify the unique session ID of the particular dump session:

```
# xfsrestore -f /dev/tape -S \
d23cbf47-b21d-1001-8f97-080069068eeb /usr2/tmp
xfsrestore: version X.X - type ^C for status and control
xfsrestore: preparing tape drive
xfsrestore: dump session found
xfsrestore: advancing tape to next media file
xfsrestore: advancing tape to next media file
xfsrestore: dump session found
xfsrestore: restore of level 0 dump of magnolia.abc.xyz.com:/test resumed Mon
Nov 28 11:50:41 1994
xfsrestore: beginning media file
xfsrestore: media file 0 (media 0, file 2)
xfsrestore: reading ino map
xfsrestore: initializing the map tree
xfsrestore: reading the directory hierarchy
xfsrestore: restoring non-directory files
xfsrestore: ending media file
xfsrestore: restoring directory attributes
xfsrestore: restore complete: 229 seconds elapsed
```

You can find the session ID by viewing the dump inventory (see "Examining xfsdump Archives" on page 47). Session labels might be duplicated, but session IDs never are.

Restoring Individual Files with xfsrestore

On the *xfsrestore* command line, you can specify an individual file or subdirectory to restore. In this example, the file *people/fred/notes* is restored and placed in the */usr/tmp* directory (that is, the file is restored in */usr/tmp/people/fred/notes*):

```
# xfsrestore -f /dev/tape -L week_1 -s people/fred/notes \
/usr/tmp
```

You can also restore a file "in place" that is, restore it directly to where it came from in the original backup. Note, however, that if you do not use a **-e**, **-E**, or **-n** option, you overwrite any existing file(s) of the same name.

In the following example, the subdirectory *people/fred* is restored in the destination */usr*—this overwrites any files and subdirectories in */usr/people/fred* with the data on the dump tape:

```
# xfsrestore –f /dev/tape –L week_1 –s people/fred /usr
```

Network Restores with xfsrestore

You can use standard network references to specify devices and files on the network. For example, to use the tape drive on a network host named *magnolia* as the source for a restore, you can use the command:

```
# xfsrestore –f magnolia:/dev/tape –L 120694u2 /usr2
xfsrestore: version X.X – type ^C for status and control
xfsrestore: preparing tape drive
xfsrestore: dump session found
xfsrestore: advancing tape to next media file
xfsrestore: dump session found
xfsrestore: restore of level 0 dump of magnolia.abc.xyz.com:/usr2 created Tue
Dec 6 10:55:17 1994
xfsrestore: beginning media file
xfsrestore: media file 0 (media 0, file 1)
xfsrestore: reading ino map
xfsrestore: initializing the map tree
xfsrestore: reading the directory hierarchy
xfsrestore: restoring non–directory files
xfsrestore: ending media file
xfsrestore: restoring directory attributes
xfsrestore: restore complete: 203 seconds elapsed
```

In this case, the dump data is extracted from the tape on *magnolia*, and the destination is the directory */usr2* on the local system. Refer to the section “Using xfsdump and xfsrestore to Copy Filesystems” on page 62 for an example of using the standard input option of *xfsrestore*.

Interactive Restores With xfsrestore

Use the **–i** option of *xfsrestore* to perform interactive file restoration. With interactive restoration, you can use the commands *ls*, *pwd*, and *cd* to peruse the filesystem, and the *add* and *delete* commands to create a list of files and subdirectories you want to restore. Then you can enter the *extract* command to restore the files, or *quit* to exit the interactive restore session without restoring files. (The use of “wildcards” is not allowed with these commands.)

The following screen output shows an example of a simple interactive restoration.

```
# xfsrestore -f /dev/tape -i -v silent .
xfsrestore: dump session found
xfsrestore: no session label
xfsrestore: session id:    d23cbeda-b21d-1001-8f97-080069068eeb
xfsrestore: no media label
xfsrestore: media id:      d23cbedb-b21d-1001-8f97-080069068eeb
do you want to select this dump? (y/n): y
selected

 --- interactive subtree selection dialog ---

the following commands are available:
        pwd
        ls [ { <name>, ".." } ]
        cd [ { <name>, ".." } ]
        add [ <name> ]
        delete [ <name> ]
        extract
        quit
        help
 -> ls
            4122 people/
            4130 two
            4126 A/
            4121 one
 -> add two
 -> cd people
 -> ls
            4124 fred/
 -> add fred
 -> ls
    *       4124 fred/
 -> extract

 ---------------- end dialog ----------------
```

In the interactive restore session above, the subdirectory *people/fred* and the file *two* were restored relative to the current working directory ("."). Note that an asterisk (*) in your *ls* output indicates your selections.

Cumulative Restores With xfsrestore

Cumulative restores sequentially restore incremental dumps to re-create filesystems and are also used to restore interrupted dumps. To perform a cumulative restore of a filesystem, begin with the media object that contains the base level dump and recover it first, then recover the incremental dump with the next higher dump level number, then the next, and so on. Use the **–r** option to inform *xfsrestore* that you are performing a cumulative recovery.

In the following example, the level 0 base dump and succeeding higher level dumps are on */dev/tape*. First the level 0 dump is restored, then each higher level dump in succession:

```
# /usr/tmp/xfsrestore –f /dev/tape –r –v silent .

 =========================== dump selection dialog ============================

the following dump has been found on drive 0

hostname: cumulus
mount point: /disk2
volume: /dev/rdsk/dks0d2s0
session time: Wed Oct 25 14:37:47 1995
level: 0
session label: “week_1”
media label: “Jun_94”
file system id: d2a602fc–b21d–1001–8938–08006906dc5c
session id: d2a60b26–b21d–1001–8938–08006906dc5c
media id: d2a60b27–b21d–1001–8938–08006906dc5c

restore this dump?
1: skip
2: restore (default)
 –> <Enter>
this dump selected for restoral

 --------------------------------- end dialog ---------------------------------

#
```

Next, enter the same command again. The program goes to the next dump and again you select the default:

```
# xfsrestore -f /dev/tape -r -v silent .

 ========================== dump selection dialog ============================

the following dump has been found on drive 0

hostname: cumulus
mount point: /disk2
volume: /dev/rdsk/dks0d2s0
session time: Wed Oct 25 14:40:54 1995
level: 1
session label: "week_2"
media label: "Jun_94"
file system id: d2a602fc-b21d-1001-8938-08006906dc5c
session id: d2a60b2b-b21d-1001-8938-08006906dc5c
media id: d2a60b27-b21d-1001-8938-08006906dc5c

restore this dump?
1: skip
2: restore (default)
 -> <Enter>
this dump selected for restoral

 -------------------------------- end dialog --------------------------------
#
```

You then repeat this process until you have recovered the entire sequence of incremental dumps. The full and latest copy of the filesystem will then have been restored. In this case, it is restored relative to ".", that is, in the directory you are in when the sequence of *xfsrestore* commands is issued.

Restore an interrupted dump just as if it were an incremental dump. Use the **-r** option to inform *xfsrestore* that you are performing an incremental restore, and answer **y** and **n** appropriately to select the proper "increments" to restore (see "Cumulative Restores With xfsrestore" on page 57).

Note that if you try to restore an interrupted dump as if it were a non-interrupted, non-incremental dump, the portion of the dump that occurred before the interruption is restored, but not the remainder of the dump. You can determine if a dump is an interrupted dump by looking in the online inventory.

Here is an example of a dump inventory showing an interrupted dump session (the crucial fields are in bold type):

```
# xfsdump -I depth=3,mobjlabel=AugTape,mnt=indy4.xyz.com:/usr
file system 0:
        fs id:          d23cb450-b21d-1001-8f97-080069068eeb
        session 0:
                mount point:    indy4.xyz.com.com:/usr
                device:         indy4.xyz.com.com:/dev/rdsk/dks0d3s2
                time:           Tue Dec  6 15:01:26 1994
                session label:  "180894usr"
                session id:     d23cc0c3-b21d-1001-8f97-080069068eeb
                level:          0
                resumed:        NO
                subtree:        NO
                streams:        1
                stream 0:
                        pathname:       /dev/tape
                        start:          ino 4121 offset 0
                        end:            ino 0 offset 0
                        interrupted:    YES
                        media files:    2
        session 1:
                mount point:    indy4.xyz.com.com:/usr
                device:         indy4.xyz.com.com:/dev/rdsk/dks0d3s2
                time:           Tue Dec  6 15:48:37 1994
                session label:  "Resumed180894usr"
                session id:     d23cc0cc-b21d-1001-8f97-080069068eeb
                level:          0
                resumed:        YES
                subtree:        NO
                streams:        1
                stream 0:
                        pathname:       /dev/tape
                        start:          ino 4121 offset 0
                        end:            ino 0 offset 0
                        interrupted:    NO
                        media files:    2
.
.
.
```

From this it can be determined that session 0 was interrupted and then resumed and completed in session 1.

To restore the interrupted dump session in the example above, use the following sequence of commands:

```
# xfsrestore -f /dev/tape -r -L 180894usr .
# xfsrestore -f /dev/tape -r -L Resumed180894usr .
```

This restores the entire */usr* backup relative to the current directory. (You should remove the *housekeeping* directory from the destination directory when you are finished.)

Interrupting xfsrestore

In a manner similar to *xfsdump* interruptions, you can interrupt an *xfsrestore* session. This allows you to interrupt a restore session and then resume it later. To interrupt a restore session, type the interrupt character (typically **<CTRL-C>**). You receive a list of options, which include interrupting the session or continuing.

```
# xfsrestore -f /dev/tape -v silent /disk2

 =========================== dump selection dialog ============================

the following dump has been found on drive 0

hostname: cumulus
mount point: /disk2
volume: /dev/rdsk/dks0d2s0
session time: Wed Oct 25 17:20:16 1995
level: 0
session label: "week1"
media label: "newtape"
file system id: d2a602fc-b21d-1001-8938-08006906dc5c
session id: d2a6129e-b21d-1001-8938-08006906dc5c
media id: d2a6129f-b21d-1001-8938-08006906dc5c

restore this dump?
1: skip
2: restore (default)
 -> 2
this dump selected for restoral

 --------------------------------- end dialog ---------------------------------


 ========================= status and control dialog ==========================
```

```
status at 17:23:52: 131/910 files restored, 14.4% complete, 42 seconds elapsed

please select one of the following operations
1: interrupt this session
2: change verbosity
3: display metrics
4: other controls
5: continue (default) (timeout in 60 sec)
 -> 1

please confirm
1: interrupt this session
2: continue (default) (timeout in 60 sec)
 -> 1
interrupt request accepted

 -------------------------------- end dialog --------------------------------

xfsrestore: initiating session interrupt
```

Resume the *xfsrestore* session with the **–R** option:

`# `**`xfsrestore –f /dev/tape –R –v silent /disk2`**

Data recovery continues from the point of the interruption.

housekeeping and orphanage Directories

The *xfsrestore* utility can create two subdirectories in the destination called *housekeeping* and *orphanage*.

The *housekeeping* directory is a temporary directory used during cumulative recovery to pass information from one invocation of *xfsrestore* to the next. It must not be removed during the process of performing the cumulative recovery but should be removed after the cumulative recovery is completed.

The orphanage directory is created if a file or subdirectory is restored that is not referenced in the filesystem structure of the dump. For example, if you dump a very active filesystem, it is possible for new files to be in the non-directory portion of the dump, yet none of the directories dumped reference that file. A warning message is displayed, and the file is placed in the *orphanage* directory, named with its original inode number and generation count (for example, 123479.14.).

Using xfsdump and xfsrestore to Copy Filesystems

You can use *xfsdump* and *xfsrestore* to pipe data across filesystems or across the network with a single command line. By piping *xfsdump* standard output to *xfsrestore* standard input you create an exact copy of a filesystem.

For example, to make a copy of */usr/people/fred* in the */usr2* directory, enter:

```
# xfsdump -J -s people/fred - /usr | xfsrestore - /usr2
```

To copy */usr/people/fred* to the network host *magnolia*'s */usr/tmp* directory:

```
# xfsdump -J -s people/fred - /usr | rsh magnolia \
xfsrestore - /usr/tmp
```

This creates the directory */usr/tmp/people/fred* on *magnolia*.

Note: The superuser account on the local system must be able to *rsh* to the remote system without a password. For more information, see hosts.equiv(4).

tar

The *tar* utility backs up files and directories. You can copy files to tape, create *tar* files, compare files on tape to files on disk, read standard input, and pipe the output of *tar* to other processes. This command is widely used on UNIX systems worldwide.

Note: XFS and tar: The **-K** option is used with the tar(1) command for files larger than 2 GB. If the **-K** option is not used, *tar* skips any files larger than 2 GB and issues a warning. Note that use of this option can create *tar* archives that are not usable on non-XFS systems. The **-K** option cannot be used with the **-O** option, which creates *tar* archives formatted in an older, pre-POSIX format.

Saving Data With tar

This section describes how to backup files with the *tar* command, and how you can back up files that have been modified since a certain specific time, or relative to the last backup (incrementally).

Backing Up Files With tar

To back up individual files with *tar*, use the command:

tar c *file*

Saving Files by Modification Date

The *tar* command does not have the capability of saving files by modification date built in. However, you can use the *find* command to archive files that have not been modified in a particular number of days:

find /usr -mtime 5 -local -type f -o -type *othertypes* **-print | tar cv -**

The *find* command locates regular, local (non-NFS) files that have not been modified in five days. The *find* command sends its output to the *tar* command.

Incremental Backups With tar

Although *tar* does not have a built-in mechanisms for incremental backups, you can use other system commands to accomplish this task.

The following example uses the same incremental scheme presented in the preceding section to back up the */usr* filesystem. It uses the *find* command to determine which files to archive:

1. Go to the top of the filesystem that you want to back up. For example:

   cd /usr

2. Create a complete backup of the filesystem:

   tar cv .

3. Each day, back up the files that have changed since the previous daily backup:

   find /usr -mtime 1 -local -print | tar cvf -

4. Every week, back up the files that have changed since the last weekly backup:

   find /usr -mtime 7 -local -type f -print | tar cvf -

5. At the end of four weeks, perform a complete backup and start the process over.

Examining tar Archives

For *tar* archives, use the **v** keyword for verbose listing of the archive contents:

```
tar tv
```

You can compare files that are archived with the original files using *tar*:

```
tar C
```

You see messages about the status of the files. Each message begins with a key character (a letter or symbol) that signifies the status of the file in the archive versus the original file. These characters are shown in Table 2-3.

Table 2-3 tar Comparison Key Characters

Key	Meaning
=	The files compare
!	The files don't compare
?	Can't read the disk file
>	Disk file doesn't exist
L	Linked to an earlier file on the tape
S	Symbolic link
B	Block special file
C	Character special file
P	Named pipe

Restoring tar Archives

To recover individual files from a *tar* archive, specify the name of the files on the command line:

`tar xv` *file1 file2 directory/file3*

cpio

Like *tar*, *cpio* archives files and directories. With *cpio*, you can copy files to tape or disk, archive empty directories, swap byte order, create portable ASCII archives, and read from and write to standard output. *cpio* is also useful for copying files and directories when the cp(1) command is unable to do so. For example, you cannot use *cp* to copy a directory to a different filesystem.

Note: XFS and cpio: Use the **-K** option with the cpio(1) command for files larger than 2 GB. If the **-K** option is not used, *cpio* skips any files larger than 2 GB and issues a warning. Note that use of this option can create *cpio* archives that are not usable on non-XFS systems. The **-K** option can be used only with the **-o** (output) option. The **-K** option cannot be used the **-c** option (which creates *cpio* archives with ASCII headers), or with the **-H** option (used to specify various header formats).

Saving Data With cpio

This section describes how to back up files with the *cpio* command, and how you can back up files that have been modified since a certain specific time, or relative to the last backup (incrementally).

Backing Up Files With cpio

To back up files with *cpio*, use the command:

cat *filelist* **| cpio -o > /dev/tape**

Saving Files by Modification Date

The *cpio* command does not have the capability of saving files by modification date built in. However, you can use the *find* command to archive files that have not been modified in a particular number of days:

```
find /usr -mtime 5 -depth -print | cpio -oO /dev/tape
```

The *-depth* argument causes *find* to print the name of the directory after printing the files in that directory. This ensures that *cpio* has permission to place the files in the directory in case the directory is read-only.

Incremental Backups With cpio

Although *tar* and *cpio* do not have built-in mechanisms for incremental backups, you can use other system commands to accomplish this task.

The following example uses the same incremental scheme presented in the preceding section to back up the */usr* filesystem. It uses the *find* command to determine which files to archive:

1. Go to the top of the filesystem that you want to back up. For example:

   cd /usr

2. Create a complete backup of the filesystem:

   cpio –oLp .

3. Each day, back up the files that have changed since the previous daily backup:

   find /usr –mtime 1 –print | cpio –pdL

4. Every week, back up the files that have changed since the last weekly backup:

   find /usr –mtime 7 –type f –print | cpio –pdL

5. At the end of four weeks, perform a complete backup and start the process over.

Examining cpio Archives

For *cpio* archives, use the following command to obtain a verbose listing:

cpio –itvI /dev/tape

The *cpio* program does not have a built-in option to compare files. To compare the files on a *cpio* archive, you must extract the archive onto disk, then use a comparing program, such as gdiff(1), diff(1), cmp(1), or dircmp(1), or compare the checksum (sum(1)) of the extracted file with that of the original.

Restoring cpio Archives

To recover individual files from a *cpio* archive, specify the name of the file(s) on the command line:

`cpio -id` *file1 directory/file2* **`< /dev/tape`**

The **-i** option causes *cpio* to read input from the tape drive, and the **-d** option causes it to create the directory it is extracting, if it doesn't already exist.

dd

The *dd* program reads from a specified input file (*stdin* is the default), performs whatever conversions you specify, and writes the result to a specified output file (*stdout* is the default). It is not specifically a backup tool, but has many extremely useful features, including the ability to:

- skip specific blocks in an archive
- skip blocks of output
- specify input and output block size
- copy a specific number of blocks
- perform various data conversions such as byte swapping

Refer to the dd(1M) reference page for details on the use of the *dd* command.