XFS® for Linux® Administration

007-4273-003

CONTRIBUTORS
Written by Steven Levine
Production by Karen Jacobson

What's New in This Guide

New Features Documented

For the SGI ProPack v 2.4 for Linux release, small clarifications and technical corrections have been made throughout this document.

Record of Revision

001	January 2003 First printing, incorporating information for the SGI ProPack v 2.1 for Linux release
002	May 2003 Incorporates information for the SGI ProPack v 2.2 for Linux release
003	January 2004 Incorporates information for the SGI ProPack v 2.4 for Linux release

Contents

Figures xi

Examples xiii

About This Guide xv
What This Guide Contains xv
Conventions Used in This Guide xvi
Additional Resources xvi
Reader Comments xvii

1. **The XFS Filesystem** 1

2. **Planning an XFS Filesystem** 3
Choosing the Filesystem Block Size 3
Choosing the Filesystem Directory Block Size 4
Choosing the Log Type and Size 4
Choosing Allocation Groups and Stripe Units 6
Disk Repartitioning 7

3. **Creating Filesystems** 9
Making an XFS Filesystem 9
Growing an XFS Filesystem 11

4. **Filesystem Maintenance** 13
Filesystem Reorganization 13
Filesystem Corruption 13
Checking XFS Filesystem Consistency With xfs_check and xfs_repair 14
Checking Filesystem Consistency 14
Repairing Inconsistent Filesystems 16

Repairing XFS Filesystem Problems 17
Common Error Messages 17
Error Messages When Files Are in lost+found 18
What to Do If xfs_repair Cannot Repair a Filesystem 19
Mounting A Filesystem Without Log Recovery 20

5. Quotas 21
Using Disk Quotas on XFS Filesystems 22
Turning on Disk Quotas for Users on XFS Filesystems 22
Turning on Disk Quotas for Groups on XFS Filesystems 22
Setting Disk Quota Limits for Users on XFS Filesystems 23
Setting Disk Quota Limits for Groups on XFS Filesystems 24
Displaying Disk Quota Information on XFS Filesystems 24
Administering Disk Quotas on XFS Filesystems 26
Monitoring Disk Space Usage with Disk Quota Accounting 27
Checking Disk Space Usage on XFS Filesystems With quota 28

6. Backup and Recovery Procedures 29
Features of xfsdump and xfsrestore 29
Media Layout for xfsdump 30
Possible xfsdump Layouts 31
Saving Data with xfsdump 36
Specifying Local Media with xfsdump 37
Specifying a Remote Tape Drive with xfsdump 38
Backing Up to a File with xfsdump 39
Reusing Tapes with xfsdump 40
Erasing Used Tapes 41
About Incremental and Resumed Dumps 41
Performing an Incremental xfsdump 42
Performing a Resumed xfsdump 42
Examining xfsdump Archives 44

About xfsrestore 47
Performing Simple Restores with xfsrestore 49
Restoring Individual Files with xfsrestore 51
Performing Network Restores with xfsrestore 52
Performing Interactive Restores with xfsrestore 52
Performing Cumulative Restores with xfsrestore 54
Interrupting xfsrestore 58
About the housekeeping and orphanage Directories 59
Using xfsdump and xfsrestore to Copy Filesystems 60
A. XFS Linux and IRIX XFS 61
Index 63

Figures

Figure 6-1 Single Dump on Single Media Object 31
Figure 6-2 Single Dump on Multiple Media Objects 32
Figure 6-3 Multiple Dumps on Single Media Object 34
Figure 6-4 Multiple Dumps on Multiple Media Objects 35

Examples

Example 3-1 mkfs Command for an XFS Filesystem Using Defaults 10

Example 3-2 mkfs Command for an XFS Filesystem Specifying Block and Log Size of Internal Log 10

Example 3-3 Growing an XFS filesystem 12

About This Guide

XFS for Linux Administration is a guide to the administration of the XFS filesystem on the LInux operating system.This manual supports SGI ProPack v 2.4 for Linux.

What This Guide Contains

This guide is organized into chapters that give procedures for performing filesystem administration tasks. These chapters are:

- Chapter 1, “The XFS Filesystem,” which provides an overview of the features of the XFS filesystem
- Chapter 2, “Planning an XFS Filesystem,” which provides information on the choices you can make when creating an XFS filesystem
- Chapter 3, “Creating Filesystems,” which provides information on creating an XFS filesystem on an empty disk partition or logical volume
- Chapter 4, “Filesystem Maintenance,” which provides information on procedures you use for XFS filesystem maintenance
- Chapter 5, “Quotas,” which provides information on using disk quotas with XFS filesystems
- Chapter 6, “Backup and Recovery Procedures,” which provides information on using the `xfsdump` and `xfsrestore` utilities to back up and recover data on XFS filesystems
- Appendix A, which provides information on the differences between XFS Linux and IRIX XFS

Conventions Used in This Guide

These type conventions and symbols are used in this guide:

`command`	This fixed-space font denotes literal items (such as commands, files, routines, pathnames, signals, messages, programming language structures, and e-mail addresses) and items that appear on the screen.
variable	Italic typeface denotes variable entries and words or concepts being defined.
`user input`	This bold, fixed-space font denotes literal items that the user inters in interactive sessions. Output is shown in nonbold, fixed-space font.
[ ]	Brackets enclose optional portions of a command or directive line.
...	Ellipses indicate that a preceding element can be repeated.
`manpage`(*x*)	Man page section identifiers appear in parentheses after man page names.

When a procedure could result in the loss of files if not performed correctly or should be performed only by knowledgeable users, the procedure is preceded by a Caution. For example:

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

Additional Resources

For more information about XFS for Linux, see these sources:

- XFS man pages
- *SGI ProPack for Linux Start Here*

Reader Comments

If you have comments about the technical accuracy, content, or organization of this document, please tell us. Be sure to include the title and part number of the document with your comments. (Online, the document number is located in the front matter of the manual. In printed manuals, the document number can be found on the back cover.)

You can contact us in any of the following ways:

- Send e-mail to the following address:

  `techpubs@sgi.com`

- Use the Feedback option on the Technical Publications Library World Wide Web page:

  http://docs.sgi.com

- Contact your customer service representative and ask that an incident be filed in the SGI incident tracking system.

- Send mail to the following address:

  Technical Publications
  SGI
  1500 Crittenden Lane, M/S 535
  Mountain View, California, 94043-1351

We value your comments and will respond to them promptly.

The XFS Filesystem

The XFS filesystem provides the following major features;

- Full 64-bit file capabilities (files larger than 2 GB)
- Rapid and reliable recovery after system crashes because of journaling technology
- Efficient support of large, sparse files (files with “holes”)
- Integrated, full-function volume manager support
- Extremely high I/O performance that scales well on multiprocessing systems
- User-specified filesystem block sizes ranging from 512 bytes up to a maximum of the filesystem page size

At least 64 MB of memory is recommended for systems with XFS filesystems.

The maximum size of an XFS filesystem under SGI ProPack for Linux is 2^{64} bytes. The maximum size of an XFS file under SGI ProPack for Linux is 2^{63}-1 bytes.

XFS uses database journaling technology to provide high reliability and rapid recovery. Recovery after a system crash is completed within a few seconds, without the use of a filesystem checker such as the `fsck` command. Recovery time is independent of filesystem size.

XFS is designed to be a very high performance filesystem. XFS as a filesystem is capable of delivering near-raw I/O performance. While traditional filesystems suffer from reduced performance as they grow in size, with XFS there is no performance penalty.

You can create filesystems with block sizes ranging from 512 bytes to a maximum of the filesystem page size. The filesystem page size is a kernel compile option and may be set to 4K, 8K, or 16K.

Filesystem extents, which provide for contiguous data within a file, are created automatically for normal files and may be configured at file creation time using the `fcntl()` system call. Extents are multiples of a filesystem block.

Inodes are created as needed by XFS filesystems. You can specify the size of inodes with the `-i size=` option to the filesystem creation command, `mkfs`. You can also specify the maximum percentage of the space in a filesystem that can be occupied by inodes with the `-i maxpct=` option of the `mkfs` command.

XFS implements fully journaled extended attributes. An extended attribute is a name/value pair associated with a file. Attributes can be attached to all types of inodes: regular files, directories, symbolic links, device nodes, and so forth. Attribute values can contain up to 64KB of arbitrary binary data. XFS implements two attribute namespaces: a user namespace available to all users, protected by the normal file permissions; and a system namespace, accessible only to privileged users. The system namespace can be used for protected filesystem meta-data such as access control lists (ACLs) and hierarchical storage manager (HSM) file migration status. For more information see the, `attr`(1) reference page.

To dump XFS filesystems, the command `xfsdump` must be used instead of `dump`. Restoring from these dumps is done using `xfsrestore`. For more information about the relationships between `xfsdump`, `xfsrestore` on XFS filesystems, see Chapter 6, "Backup and Recovery Procedures."

Planning an XFS Filesystem

The following subsections discuss preparation for and choices you can make when creating an XFS filesystem.

Choosing the Filesystem Block Size

XFS allows you to choose the logical block size for each filesystem by using the `-b size=` option of the mkfs command. (Physical disk blocks remain 512 bytes.)

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

- The minimum block size is 512 bytes. Small block sizes increase allocation overhead which decreases filesystem performance, but in general, the recommended block size for filesystems under 100 MB and for filesystems with many small files is 512 bytes. The filesystem block size must be a power of two.
- The default block size is 4096 bytes (4K). This is the recommended block size for filesystems over 100 MB.
- The maximum block size is the page size of the kernel, which is 4K on x86 systems and is configurable on IA-64 systems. Because large block sizes can waste space, in general block sizes should not be larger than 4096 bytes (4K).
- For news servers, it is recommended that you use a filesystem block size of 512 bytes and a directory block size of 4096 bytes.

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

Choosing the Filesystem Directory Block Size

An XFS file system allows you to select a logical block size for the filesystem directory that is greater than the logical block size of the filesystem by using the `-n` option of the `mkfs` command. This allows you to choose a filesystem block size to match the distribution of data file sizes without adversely affecting directory operation performance. Using this option could improve performance for a filesystem with many small files, such as a news or mail filesystem. In this case, the filesystem logical block size could be small (512, 1K, or 2K bytes) and the logical block size for the filesystem directory could be large (4K or 8K bytes); this can improve the performance of directory lookups because the tree storing the index information has larger blocks and less depth.

You should consider setting a logical block size for a filesystem directory that is greater than the logical block size for the filesystem if you are supporting an application that reads directories (with the `readdir`(3C) or `getdents`(2) system calls) many times in relation to how much it creates and removes files. Using a small filesystem block size saves on disk space and on I/O throughput for the small files.

In a Linux XFS file system, the data needed to perform a `readdir` operation is segregated from the index information. Directory data blocks can be "read-ahead" in a `readdir`. Performing read-ahead improves the `readdir` performance dramatically. Because the data needed for a `readdir` operation and index information are separate in a directory block, the offset in a directory is limited to 32 bits.

Choosing the Log Type and Size

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

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

External — When you specify that log records are maintained in a dedicated log device, the log is called an *external* log. You use the `-l` option of the `mkfs` command to specify this device.

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

The guidelines for choosing the log type are as follows:

- If you want the data and log records to be on different partitions, use an external log.
- If you want the data and the log subvolume of a logical volume to be on different partitions or to use different subvolume configurations, use an external log.
- If you want the log subvolume of a logical volume to be striped independently from the data subvolume, you must use an external log.
- If you are making the XFS filesystem on a logical volume that has a log subvolume, you must specify this log subvolume as the log device with the `-l` option of the `mkfs` command in order.

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

You can choose the amount of disk space to dedicate to the log (called the log size). The minimum log size for a filesystem is enforced by the size of the largest transaction, which depends on the filesystem and directory block sizes. The maximum log size is 64k blocks or 128 MB, whichever is smaller (this will depend on the block size).

For internal logs, the size of the log is specified with the `-l size=` option when you create the filesystem with the `mkfs` command.The default log size grows with the size of the filesystem up to the maximum log size, 128 megabytes, on a 1 terabyte filesystem. The log size is specified in bytes as described in “Choosing the Filesystem Block Size” on page 3, or as a multiple of the filesystem block size by using the suffix “b.”

For a filesystem which is contained in a striped logical volume, the default internal log size is rounded up to a multiple of the stripe unit size. In this case, the user-specified size value must be a multiple of the stripe unit size.

For external logs, the default size of the log is the same as the size of the log device. You can specify the size of the log with the `-l size= option` of the `mkfs` command, but any additional space in the log device cannot be used. You may find that you need to repartition a disk to create a properly sized log subvolume.

For filesystems with a very high transaction activity, a large log size is recommended. You should avoid making your log too large, however, since a large log can increase filesystem mount time after a crash.

Choosing Allocation Groups and Stripe Units

The data section of an XFS filesystem is divided into allocation groups. You can select the number of allocation groups when you create an XFS filesystem or, alternatively, you can select the size of an allocation group. The larger the number of allocation groups, the more parallelism can be achieved when allocating blocks and inodes. You should avoid selecting a very large number of allocation groups or an allocation group size that will yield a very large number of allocation groups; a large number of allocation groups causes an unreasonable amount of CPU time to be used when the filesystem is close to full.

The minimum allocation group size is 16MB; the maximum size is just under 4 GB.

The default number of allocation groups is 8, unless the filesystem is smaller than 128 MB or larger than 8 GB. When the filesystem is smaller than 128 MB, the default number of allocation groups is less than 8, since the minimum allocation group size is 16MB. In this case, the data section, by default, will be divided into as many allocation groups as possible that are at least 16MB. When the filesystem is larger than 8GB, but smaller than 64GB, the default number of allocation groups is greater than 8, with each allocation group approximately 1 GB in size. When the filesystem is larger than 64GB, the default number of allocation groups is still greater than 8, but the allocation group size is 4GB.

XFS allows you to select the stripe unit for a RAID device or stripe volume. This ensures that data allocations, inode allocations, and the internal log will be aligned along stripe units when the end of file is extended and the file size is larger than 512KB. You specify stripe units in 512-byte block units or in bytes. See the `mkfs.xfs`(1M) man page for information on specifying stripe units.

When you specify a stripe unit, you also specify a stripe width. You specify a stripe width in 512-byte block units or in bytes. The stripe width must be a multiple of the stripe unit.

The stripe width will be the preferred I/O size returned in the `stat()` system call. See the `mkfs_xfs`(1M) man page for information on specifying stripe width.

When used in conjunction with the `-b` (block size) option of the `mkfs` command, you can use the `-d su=` and `-d sw=` options to specify the stripe unit and stripe width in filesystem blocks.

For a RAID device, the default stripe unit is 0, indicating that the feature is disabled. It is prudent of the sysadmin to configure the stripe unit and width sizes of RAID devices. This should be done to avoid unexpected performance anomalies caused by the filesystem doing non-optimal I/O operations to the RAID unit. For example, if a block write is not aligned on a RAID stripe unit boundary and is not a full stripe unit, the RAID will be forced to do a read/modify/write cycle to write the data. This can have a significant performance impact. By setting the stripe unit size properly, XFS will avoid unaligned accesses.

For a striped volume, the stripe unit that was specified when the volume was created is provided by default.

Disk Repartitioning

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

- If the system disk has separate partitions for root and `usr`, `var`, and `home` filesystems, the root filesystem may be running out of space. Repartitioning is a way to increase the space in root (at the expense of the size of `usr`, `var`, or `home`) or to solve the problem by combining root and `usr`, `var`, or `home` into a single partition.
- If you plan to use logical volumes, you may want to put the XFS log into a small subvolume. This requires disk repartitioning to create a small partition for the log subvolume.
- If you plan to use logical volumes, you may want to repartition to create disk partitions of equal size that can be striped or plexed.

Creating Filesystems

This chapter explains how to create an XFS filesystem on an empty disk partition or logical volume. It also provides an overview on growing existing XFS filesystems.

The examples in this section use the XSCSI device naming convention. For information on XSCSI device naming, see *SGI ProPack for Linux Start Here*.

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

Making an XFS Filesystem

Use the following procedure to make an XFS filesystem.

1. Review Chapter 2 to verify that you are ready to begin this procedure.
2. Identify the device name of the partition or logical volume where you plan to create the filesystem. This is the value of *partition* in the examples below.

   For example, if you plan to use partition 1 of a SCSI disk with target 3 connected to port 1 of a SCSI controller in slot 2 of PCI bus 2, the *partition* is `/dev/xscsi/pci02.02.0-1/target3/lun0/part1`.
3. If the disk partition is already mounted, unmount it:

   `# umount partition`

   Any data that is on the disk partition is destroyed.
4. If you are making a filesystem on a disk partition or on a logical volume that does not have a log subvolume and want to use the default values for block size and log size, use this `mkfs` command to create the new XFS filesystem:

   `# mkfs partition`

Example 3-1 shows the command line to create an XFS filesystem using the defaults and system output.

Example 3-1 mkfs Command for an XFS Filesystem Using Defaults

```
# mkfs.xfs /dev/xscsi/pci02.02.0-1/target3/lun0/part1
meta-data=/dev/xscsi/pci02.02.0-1/target3/lun0/part1 isize=256    agcount=18,
agsize=1048576 blks
data     =                       bsize=4096   blocks=17921788, imaxpct=25
         =                       sunit=0      swidth=0 blks, unwritten=0
naming   =version 2              bsize=4096
log      =internal log           bsize=4096   blocks=2187, version=1
         =                       sunit=0 blks
realtime =none                   extsz=65536  blocks=0, rtextents=0
```

5. If you are making a filesystem on a disk partition or on a logical volume that does not have a log subvolume and want to specify the block size and log size, use this `mkfs` command to create the new XFS filesystem:

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

   blocksize is the filesystem block size (see "Choosing the Filesystem Block Size" on page 3) and *logsize* is the size of the area dedicated to log records (see "Choosing the Log Type and Size" on page 4). The default values are 4 KB blocks and a 1000-block log.

   Example 3-2 shows the command line used to create an XFS filesystem and the system output. The filesystem has a 10 MB internal log and a block size of 1 KB and is on the partition `/dev/dsk/dks0d4s7`.

Example 3-2 mkfs Command for an XFS Filesystem Specifying Block and Log Size of Internal Log

```
# mkfs.xfs -b size=1k -l size=10m /dev/xscsi/pci02.02.0-1/target3/lun0/part1
meta-data=/dev/xscsi/pci02.02.0-1/target3/lun0/part1 isize=256    agcount=18,
agsize=4194304 blks
data     =                       bsize=1024   blocks=71687152, imaxpct=25
         =                       sunit=0      swidth=0 blks, unwritten=0
naming   =version 2              bsize=4096
log      =internal log           bsize=1024   blocks=10240, version=1
         =                       sunit=0 blks
realtime =none                   extsz=65536  blocks=0, rtextents=0
```

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

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

   blocksize is the block size for filesystem (see "Choosing the Filesystem Block Size" on page 3),and *volume* is the device name for the volume.

7. If you are making a filesystem with a directory block size that is larger than the filesystem block size, use this `mkfs` command to create the new XFS filesystem:

   ```
   # mkfs -b size=blocksize -n size=dirblocksize partition
   ```

   blocksize is the filesystem block size (see "Choosing the Filesystem Block Size" on page 3) and *dirblocksize* is the directory block size (see "Choosing the Filesystem Directory Block Size" on page 4).

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

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

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

   ```
   partition mountdir xfs defaults 0 0
   ```

Growing an XFS Filesystem

You grow an existing XFS filesystem by increasing the available disk space and growing the existing filesystem with the `xfs_growfs(1M)` command.The filesystem must be mounted to be grown The existing contents of the filesystem are undisturbed, and the added space becomes available for additional file storage.

Growing an XFS filesystem is supported on XVM volumes. You must first grow the XVM volume before growing the XFS filesystem. For information on XVM volumes, see the *XVM Volume Manager Administrator's Guide*.

The following example grows a filesystem mounted at `/mnt`:

Example 3-3 Growing an XFS filesystem

```
# xfs_growfs /mnt
meta-data=/mnt                   isize=256    agcount=30, agsize=262144 blks
data     =                       bsize=4096   blocks=7680000, imaxpct=25
         =                       sunit=0      swidth=0 blks, unwritten=0
naming   =version 2              bsize=4096
log      =internal               bsize=4096   blocks=1200 version=1
         =                       sunit=0 blks
realtime =none                   extsz=65536  blocks=0, rtextents=0
data blocks changed from 7680000 to 17921788
```

Filesystem Maintenance

The chapter provides information on the following topics:

- "Filesystem Reorganization" on page 13
- "Filesystem Corruption" on page 13
- "Checking XFS Filesystem Consistency With xfs_check and xfs_repair" on page 14
- "Repairing XFS Filesystem Problems" on page 17

Filesystem Reorganization

Filesystems can become fragmented over time. When a filesystem is fragmented, blocks of free space are small and files have many extents. The `xfs_fsr` command reorganizes filesystems so that the layout of the extents is improved. This improves overall performance. See the `xfs_fsr` reference page for information on the `xfs_fsr` command.

Filesystem Corruption

Most often, a filesystem is corrupted because the system experienced a panic. This can be caused by system software failure, hardware failure, or human error (for example, pulling the plug). Another possible source of filesystem corruption is overlapping partitions.

There is no foolproof way to predict hardware failure. The best way to avoid hardware failures is to conscientiously follow recommended diagnostic and maintenance procedures.

Human error is probably the greatest single cause of filesystem corruption. To avoid problems, follow these rules closely:

- Always shut down the system properly. Do not simply turn off power to the system. Use a standard system shutdown tool, such as the `shutdown` command.
- Never remove a filesystem physically (pull out a hard disk) without first turning off power.
- Never physically write-protect a mounted filesystem, unless it is mounted read-only.
- Do not mount filesystems on dual-hosted disks on two systems simultaneously.

The best way to insure against data loss is to make regular, careful backups.

In some cases, XFS filesystem corruption, even on the root file system, can be repaired with the command `xfs_repair`. For more information about `xfs_repair` see "Checking XFS Filesystem Consistency With xfs_check and xfs_repair" on page 14

Checking XFS Filesystem Consistency With xfs_check and xfs_repair

XFS filesystem consistency checking can be done using the `xfs_check` command and the dry-run mode of the `xfs_repair` command. The `xfs_repair` command is sometimes able to repair filesystem inconsistencies.

Note: If you suspect problems with the root file system, you should use a root disk or an alternate root disk to run `xfs_repair`.

Checking Filesystem Consistency

The filesystem consistency checking commands for XFS filesystems are `xfs_check` and `xfs_repair -n`. Unlike `fsck`, neither `xfs_check` nor `xfs_repair` are invoked automatically on system startup. They should be used only if you suspect a filesystem consistency problem.

Before running `xfs_check` or `xfs_repair -n`, the filesystem to be checked must be unmounted cleanly using normal system administration procedures (the `umount` command or system shutdown), not as a result of a crash or system reset. If the filesystem has not been unmounted cleanly, mount it and unmount it cleanly before running `xfs_check` or `xfs_repair -n`.

`xfs_repair -n` checks XFS filesystem consistency. `xfs_repair -n` performs a more complete check than `xfs_check`. The command line for `xfs_repair -n` is:

```
# xfs_repair -n device
```

device is the device file for a disk partition or logical volume that contains an XFS filesystem, for example:

```
# xfs_repair -n /dev/xscsi/pci02.02.0-1/target3/lun0/part1
```

The following example shows output with no consistency problems found:

```
Phase 1 - find and verify superblock...
Phase 2 - using internal log
        - scan filesystem freespace and inode maps...
        - found root inode chunk
Phase 3 - for each AG...
        - scan (but don't clear) agi unlinked lists...
        - process known inodes and perform inode discovery...
        - agno = 0
        - agno = 1
        ...
        - process newly discovered inodes...
Phase 4 - check for duplicate blocks...
        - setting up duplicate extent list...
        - check for inodes claiming duplicate blocks...
        - agno = 0
        - agno = 1
        ...
No modify flag set, skipping phase 5
Phase 6 - check inode connectivity...
        - traversing filesystem starting at / ...
        - traversal finished ...
        - traversing all unattached subtrees ...
        - traversals finished ...
        - moving disconnected inodes to lost+found ...
Phase 7 - verify link counts...
No modify flag set, skipping filesystem flush and exiting.
```

`xfs_check` also checks XFS filesystem consistency. It can be used on filesystems with Extended Attributes (see the `attr`(1) reference page). (`xfs_repair` performs only limited checking of Extended Attributes.) The command line for `xfs_check` is:

```
# xfs_check device
```

If no consistency problems were found, `xfs_check` returns without displaying any messages.

Repairing Inconsistent Filesystems

`xfs_repair` (without the `-n` option) checks XFS filesystem consistency and, if problems are detected, corrects them if possible. The filesystem to be checked and repaired must have been unmounted cleanly using normal system administration procedures (the `umount` command or system shutdown), not as a result of a crash or system reset. If the filesystem has not been unmounted cleanly, mount it and unmount it cleanly before running `xfs_repair`.

The command line for `xfs_repair` when you want it to repair any inconsistencies it finds is:

```
# xfs_repair device
```

device is the device file for a disk partition or logical volume that contains an XFS filesystem. It must not be mounted.

The following is an example of the output you see from running `xfs_repair` on a clean filesystem:

```
# xfs_repair /dev/xscsi/pci02.02.0-1/target3/lun0/part1
Phase 1 - find and verify superblock...
Phase 2 - using internal log
        - zero log...
        - scan filesystem freespace and inode maps...
        - found root inode chunk
Phase 3 - for each AG...
        - scan and clear agi unlinked lists...
        - process known inodes and perform inode discovery...
        - agno = 0
        - agno = 1
        ...
        - process newly discovered inodes...
Phase 4 - check for duplicate blocks...
        - setting up duplicate extent list...
        - clear lost+found (if it exists) ...
        - check for inodes claiming duplicate blocks...
        - agno = 0
        - agno = 1
        ...
```

```
Phase 5 - rebuild AG headers and trees...
        - reset superblock...
Phase 6 - check inode connectivity...
        - resetting contents of realtime bitmap and summary inodes
        - ensuring existence of lost+found directory
        - traversing filesystem starting at / ...
        - traversal finished ...
        - traversing all unattached subtrees ...
        - traversals finished ...
        - moving disconnected inodes to lost+found ...
Phase 7 - verify and correct link counts...
done
```

For information about using `xfs_repair` on an inconsistent filesystem, see "Repairing XFS Filesystem Problems" on page 17.

Repairing XFS Filesystem Problems

The `xfs_repair` command checks XFS filesystem consistency and sometimes repairs problems that are found. This section describes the messages that you may see from `xfs_repair` and what to do if `xfs_repair` is not able to repair a filesystem.

Common Error Messages

Some common error messages from `xfs_repair` and the repairs that it performs are the following:

`disconnected inode 242002, moving to lost+found`
: `xfs_repair` found an inode that is in use, but is not connected to the filesystem. The inode is moved to the filesystem's `lost+found` directory. Its name is its inode number, in this example 242002. If the disconnected inode is a directory, the directory's subtree is preserved—all its child inodes are automatically moved with it, so the entire directory subtree moves to `lost+found`.

`imap claims in-use inode 2444941 is free, correcting imap`
: The inode allocation map in the filesystem behaves as if inode 2444941 is free, but the inode itself looks like it is still in use. `xfs_repair` corrects the inode map to say that the inode is in use.

`entry references free inode 2444940 in shortform directory 2444922 junking entry “fb” in directory inode 2444922`

A directory entry points to an inode that `xfs_repair` has determined is actually free. `xfs_repair` junks the directory entry. The term *shortform* means a small directory. In larger directories, the entry deletion is usually a two-pass process. In this case, the second part of the message reads something like `marking bad entry`, `marking entry to be deleted`, or `will clear entry`.

`resetting inode 241996 nlinks from 5 to 3`

`xfs_repair` detected a mismatch between the number of directory entries pointing to the inode (links) and the number of links recorded in the inode. It corrected the number (from 5 to 3 in this case).

`cleared inode 2444926`

There was something wrong with the inode that was not correctable, so `xfs_repair` turned it into a zero-length free inode. This usually happens because the inode claims blocks that are used by something else or the inode itself is badly corrupted. Typically, the `cleared inode` message is preceded by one or more messages indicating why the inode needs to be cleared.

Error Messages When Files Are in lost+found

If `xfs_repair` has put files and directories in a filesystem’s `lost+found` directory and you do not remove them, the next time you run `xfs_repair` it temporarily disconnects the inodes for those files and directories. They are reconnected before `xfs_repair` terminates. As a result of the disconnected inodes in `lost+found`, you see output like this:

```
Phase 1 - find and verify superblock...
Phase 2 - zero log...
        - scan filesystem freespace and inode maps...
        - found root inode chunk
Phase 3 - for each AG...
        - scan and clear agi unlinked lists...
        - process known inodes and perform inode discovery...
        - agno = 0
        - agno = 1
        ...
        - process newly discovered inodes...
Phase 4 - check for duplicate blocks...
```

```
        - setting up duplicate extent list...
        - clear lost+found (if it exists) ...
        - clearing existing “lost+found” inode
        - deleting existing “lost+found” entry
        - check for inodes claiming duplicate blocks...
        - agno = 0
imap claims in-use inode 242000 is free, correcting imap
        - agno = 1
        - agno = 2
        ...
Phase 5 - rebuild AG headers and trees...
        - reset superblock counters...
Phase 6 - check inode connectivity...
        - ensuring existence of lost+found directory
        - traversing filesystem starting at / ...
        - traversal finished ...
        - traversing all unattached subtrees ...
        - traversals finished ...
        - moving disconnected inodes to lost+found ...
disconnected inode 242000, moving to lost+found
Phase 7 - verify and correct link counts...
done
```

In this example, inode 242000 was an inode that was moved to `lost+found` during a previous `xfs_repair` run. This run of `xfs_repair` found that the filesystem is consistent. If the `lost+found` directory had been empty, in phase 4 only the messages about clearing and deleting the `lost+found` directory would have appeared. The left-justified `imap claims and disconnected inode` messages appear (one pair of messages per inode) if there are inodes in the `lost+found` directory.

What to Do If xfs_repair Cannot Repair a Filesystem

If `xfs_repair` fails to repair the filesystem successfully, try giving the same `xfs_repair` command twice more; `xfs_repair` may be able to make more repairs on successive runs. If `xfs_repair` fails to fix the consistency problems in three tries, your next step depends upon where it failed:

- If `xfs_repair` failed in phase 1, you must restore lost files from backups.
- If `xfs_repair` failed in phase 2 or later, you may be able to restore files from the disk by backing up and restoring the files on the filesystem.

If `xfs_repair` failed in phase 2 or later, follow these steps:

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 logical volume.
4. Restore the files from the backup with `xfsrestore`.

See Chapter 6 for information about `xfsdump` and `xfsrestore`.

Mounting A Filesystem Without Log Recovery

If a filesystem is damaged to the extent that you are unable to mount the filesystem successfully in the standard fashion, you may be able to recover some of its data by mounting the filesystem with the `-o norecover` option of the `mount` command. This option mounts the filesystem without running log recovery. You must mount the filesystem as read-only when you use this option.

Quotas

If your system is constantly short of disk space and you cannot increase the amount of available space, you may be forced to implement disk quotas. Quotas allow you to limit the amount of space a user can occupy and the number of files (inodes) each user can own. The XFS filesystem supports disk quotas to automate this process. You can use this system to implement specific disk usage quotas for each user on your system. You can implement *hard* or *soft* limits; hard limits are enforced by the system, soft limits merely remind the user to trim disk usage. Disk usage limits are not enforced for root.

With soft limits, whenever a user logs in with a usage greater than the assigned soft limit, that user is warned (by the `login` command). When the user exceeds the soft limit, the timer is enabled. Any time the quota drops below the soft limits, the timer is disabled. If the timer is enabled longer than a time period set by the system administrator, the particular limit that has been exceeded is treated as if the hard limit has been reached, and no more disk space is allocated to the user. The only way to reset this condition is to reduce usage below the quota. Only root may set the time limits, and this is done on a per-filesystem basis.

Several options are available on XFS filesystems. You can impose limits on some users and not others, some filesystems and not others, and on total disk usage per user, or total number of files. In addition, on XFS filesystems there is no limit to the number of accounts and there is little performance penalty for large numbers of users.

On XFS filesystems, you can also impose limits according to group IDs as well as user IDs. For information on using disk quotas for user and group IDs, see "Using Disk Quotas on XFS Filesystems" on page 22.

Disk quotas on XFS filesystems can be used to do disk usage accounting. Disk usage accounting monitors disk usage, but does not enforce disk usage limits. See "Monitoring Disk Space Usage with Disk Quota Accounting" on page 27 for more information.

Disk quotas are described in more detail in the `quotas(4)` reference page. Procedures for imposing and monitoring disk quotas are described in "Using Disk Quotas on XFS Filesystems" on page 22.

Using Disk Quotas on XFS Filesystems

This section describes basic commands for administering disk quotas on XFS filesystems. Additional commands are described on the `quota`(1), `edquota`(1M), and `repquota`(1M) reference pages.

You can set disk quotas for individual users and you can set disk quotas for groups according to group ID.

For XFS filesystems, you must first turn on disk quotas on a filesystem, then set quotas on that filesystem for users and projects or groups.

Turning on Disk Quotas for Users on XFS Filesystems

You can turn on quotas for users in these ways:

- To turn on disk quotas automatically for users on a non-root filesystem, include the option `quota` in the `/etc/fstab` entry, for example:

  ```
  /dev/foo / xfs rw,quota 0 0
  ```

- To turn on disk quotas manually for users on a non-root filesystem, mount the filesystem with this command:

  ```
  # mount -o quota fsname rootdir
  ```

  fsname is the device name of the filesystem. *rootdir* is the directory where the filesystem is mounted.

- To turn on disk quotas for users on the root filesystem, you must pass the quota mount options into the kernel at boot time through the Linux `rootflags` boot option. The following example adds the `rootflags=quota` option to the append line in `elilo.conf`:

```
append="root=/dev/xscsi/pci00.01.0-1/tsrget0/lun0/part3 rootflags=quota
```

Turning on Disk Quotas for Groups on XFS Filesystems

You can turn on quotas for groups in these ways:

- To turn on disk quotas automatically for groups on a non-root filesystem, include the option g`quota` in the `/etc/fstab` entry, for example:

```
/dev/foo / xfs rw,gquota 0 0
```

- To turn on disk quotas manually for groups on a non-root filesystem, mount the filesystem with this command:

  # **mount -o gquota** *fsname rootdir*

  fsname is the device name of the filesystem. *rootdir* is the directory where the filesystem is mounted.

- To turn on disk quotas manually for groups on the root filesystem, give these commands:

- To turn on disk quotas for groups on the root filesystem, you must pass the quota mount options into the kernel at boot time through the Linux `rootflags` boot option. The following example adds the `rootflags=gquota` option to the append line in `elilo.conf`:

```
append="root=/dev/xscsi/pci00.01.0-1/tsrget0/lun0/part3 rootflags=gquota
```

Setting Disk Quota Limits for Users on XFS Filesystems

After turning on disk quotas on a filesystem, you can set limits for users on that filesystem using the commands below. You can preview the results of each of these commands by adding a `-n` option, which is the dry-run option.

- To specify limits for users interactively, give this command:

  # **edquota** *name* ...

  name is a user name or numeric user ID. The screen clears, and you are placed in the editor specified by the EDITOR environment variable (`vi` if `$EDITOR` is not set) to edit the disk quotas for the filesystem mounted at *rootdir* for the first *user* listed on the command line. You see:

  `fs` *`rootdir`* `kbytes (soft = 0, hard = 0) inodes (soft = 0, hard = 0)`

  The first pair of soft and hard numbers are the soft and hard limits for disk usage in kilobytes in the filesystem at *rootdir*. The second pair of soft and hard numbers are the soft and hard limits for the number of files that *user* can own in the filesystem.

  Edit the zeros to set the limits to sizes you choose. A limit of zero is not enforced. After you set the limits, save the file and quit the editor. If you specified more than one *user* on the command line, another instance of the editor appears with the line above. Edit this line to enter the limits for the second *user*. Continue until lines have been edited for all *user*s.

- To specify that users are to have the same limits as another user (*proto_name*),enter this command:

  # **edquota -p** *proto_name name* **...**

- To specify limits for a user non-interactively, enter this command:

  # **edquota -f** *rootdir* **-l **
  uid=*userid***,bsoft=***kvalue***,bhard=***kvalue***,isoft=***value***,ihard=***value*

  userid is a user name or numeric user ID. Each *kvalue* is a soft or hard limit for disk usage in kilobytes.Each *value* is a soft or hard limit for the number of files the user can own.

- To use the file (*quotafile*) created by command `repquota -e` as input to the `edquota` command, enter this command:

  # **edquota -i** *quotafile*

Setting Disk Quota Limits for Groups on XFS Filesystems

After turning on disk quotas on a filesystem, you can set limits for groups on that filesystem. You set limits for groups just as you do for users, by using the `edquota` command as described in "Setting Disk Quota Limits for Users on XFS Filesystems" on page 23.

To use the `edquota` command to set limits for a project, you include the `-g` option on the command line. When you use the `-g` option with `edquota`, any name specified on the command line is considered a group name. For example, to specify limits for groups interactively, give this command:

edquota -g *name* **...**

name is a group name or numeric group ID. For information on additional options of the `edquota` command, see the `edquota`(1M) man page.

Use of disk quota limits for groups and use of disk quota limits for projects are mutually exclusive.

Displaying Disk Quota Information on XFS Filesystems

Some commands that display information about disk quotas are as follows:

- To display a report that shows whether disk quotas are on or off for each filesystem, give this command as superuser:

```
# repquota -sa
/dev/root (/):
--------------
Status
        user  quota accounting       : on
        user  quota limit enforcement: on
        proj  quota accounting       : off
        proj  quota limit enforcement: off
        group quota accounting       : off
        group quota limit enforcement: off
Quota Storage
        user  quota inum 103156, blocks 8, extents 7
        proj  quota inum 103157, blocks 5, extents 4
Default Limits
        blocks time limit: 1.0 week
        files  time limit: 1.0 week
Cache
        dquots currently cached in memory: 14
--------------------
```

The sections of the output are as follows:

Status	Lists the status of disk space accounting (on or off) and enforcement of disk quotas (on or off) for this filesystem.
Quota Storage	Blocks and extents are the number of filesystem blocks and extents used to store disk quota information. The `inum` value is the inode number at which quota information is stored and is for internal use only.
Default Limits	The blocks and files time limits are the default lengths of time for this filesystem that users have to reduce their disk space usage or number of files below their soft limits. These time limits can be set on a per-user basis by the command `edquota -t`.
Cache	This section is for internal use only

- To get information about your disk quotas, enter this command:

```
# quota -v
Disk quotas for margo (uid 1606):
Filesystem    usage   quota   limit    timeleft  files   quota   limit   timeleft
/           138360       0       0             14971       0       0
/e         4156360   41200       0    1.6 days 222264       0       0
```

The columns in this output are:

`Filesystem`	Lists each of the filesystems that have quotas turned on.
`usage`	Lists the user's disk usage on each filesystem.
`quota`	The user's soft limit for disk usage or files on each filesystem.
`limit`	The user's hard limit for disk usage or files on each filesystem.
`timeleft`	For filesystems where the user's soft limit for disk usage or files is exceeded, gives the number of days until the user is prohibited from using additional disk space or creating more files.
`files`	The number of files owned by the user on each filesystem.

- To get information about your group disk quotas, enter this command:

```
# quota -g -v
Disk quotas for staff (gid 50):
Filesystem    usage   quota   limit    timeleft  files   quota  limit    timeleft
/part1            4       0       0                 2       0      0
```

Administering Disk Quotas on XFS Filesystems

If the filesystem being dumped contains quotas, `xfsdump` will use `repquota`(1M) to store the quotas in the following files in the root of the filesystem to be dumped:

`xfsdump_quotas`
: user quotas

`xfsdump_quotas_group`
: group quotas

These files will then be included in the dump. These files will appear only for those quotas which are enabled on the filesystem being dumped. Upon restoration, `edquota`(1M) can be used to reactivate the quotas for the filesystem. Note, however, that the `xfsdump_quotas` file will probably require modification to change the filesystem or UIDs if the filesystem has been restored to a different partition or system.

To create a file that lists the current quota limits of all the filesystems for users, enter this command as superuser:

repquota -a -e *quotafile*

To create a file that lists the current quota limits of all the filesystems for groups, enter this command as superuser:

```
# repquota -g -a -e quotafile
```

Monitoring Disk Space Usage with Disk Quota Accounting

The disk quotas system can be used to monitor disk space usage without enforcing disk usage limits. Disk quota accounting can be enabled by user or by group.

On XFS filesystems, use these commands to turn on disk usage accounting without enforcement, stop disk usage accounting, and report disk space usage:

- To turn on disk usage accounting automatically on a filesystem for user quotas, include the option `qnoenforce` in the `/etc/fstab` entry, for example:

  ```
  /dev/foo / xfs rw,qnoenforce 0 0
  ```

  To turn on disk usage accounting automatically on a filesystem for group quotas, include the option `gqnoenforce` in the `/etc/fstab` entry, for example:

  ```
  /dev/foo / xfs rw,gqnoenforce 0 0
  ```

- To turn on disk usage accounting manually for user quotas on a non-root filesystem, when mounting the filesystem, use this `mount` command:

  ```
  # mount -o qnoenforce fsname rootdir
  ```

  fsname is the device name of the filesystem. *rootdir* is the directory where the filesystem is mounted.

  To turn on disk usage accounting manually on a non-root filesystem for group quotas when mounting the filesystem, use this `mount` command:

  ```
  # mount -o gqnoenforce fsname rootdir
  ```

- To turn on disk usage accounting manually on the root filesystem for user quotas, execute the following commands. The `quotaon` command turns on disk accounting with enforcement, and the `quotaoff -o` command turns off the enforcement.

  ```
  # quotaon -v /
  # quotaoff -v -o enforce /
  # reboot
  ```

  To turn on disk usage accounting manually on the root filesystem for group quotas, give these commands:

```
# quotaon -v -o gquota /
# quotaoff -v -o gqenforce /
# reboot
```

- To stop disk usage accounting on a filesystem for user quotas, give this command:

  # **quotaoff** *fsname*

  To stop disk usage accounting on a filesystem for group quotas, give this command:# **quotaoff -o gquota** *fsname*

- To get information about disk usage, use the commands described in "Checking Disk Space Usage on XFS Filesystems With quota" on page 28.

Checking Disk Space Usage on XFS Filesystems With quota

The `quota` command reports the amount of disk usage per user, per group, or per project on a filesystem, as well as additional information about the disk quotas. On XFS filesystems, you must turn on quotas to use this feature, even if you are not going to enforce quota limits. For instructions on monitoring disk space usage without enforcing disk usage limits see "Monitoring Disk Space Usage with Disk Quota Accounting" on page 27.

For information on the output of the quota command, see "Displaying Disk Quota Information on XFS Filesystems" on page 24.

Backup and Recovery Procedures

This chapter 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.

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. `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 the following:

- XFS features including 64-bit inode numbers, file lengths, and holes
- multiple media types,
- regular, directory, symbolic link, block and character special, FIFO, and socket file types

`xfsdump` and `xfsrestore` 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.

Note: If you are using disk quotas on XFS filesystems, refer to Chapter 5 for more information.

Media Layout for xfsdump

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

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 xfsdump 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 like that illustrated in Figure 6-1. (In the data layout diagrams in this section, the optional inventory object is always included.)

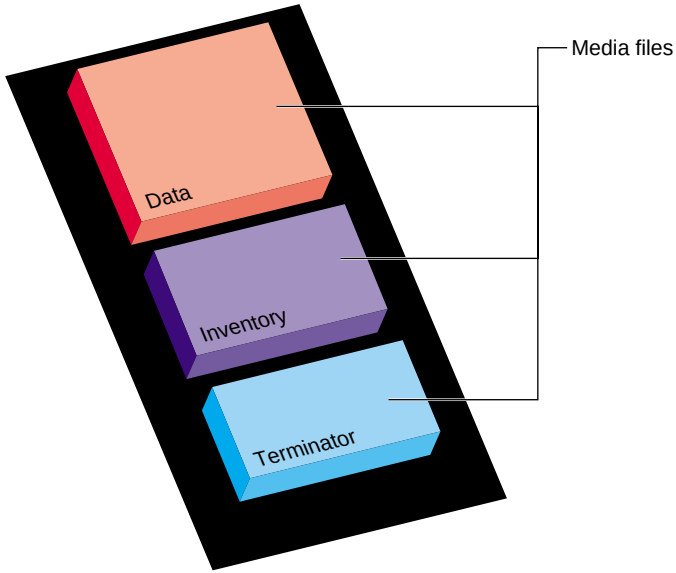

Figure 6-1 Single Dump on Single Media Object

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 you specify multiple drives, 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 6-2 illustrates the data layout of a single dump session that requires two media objects on each of two devices.

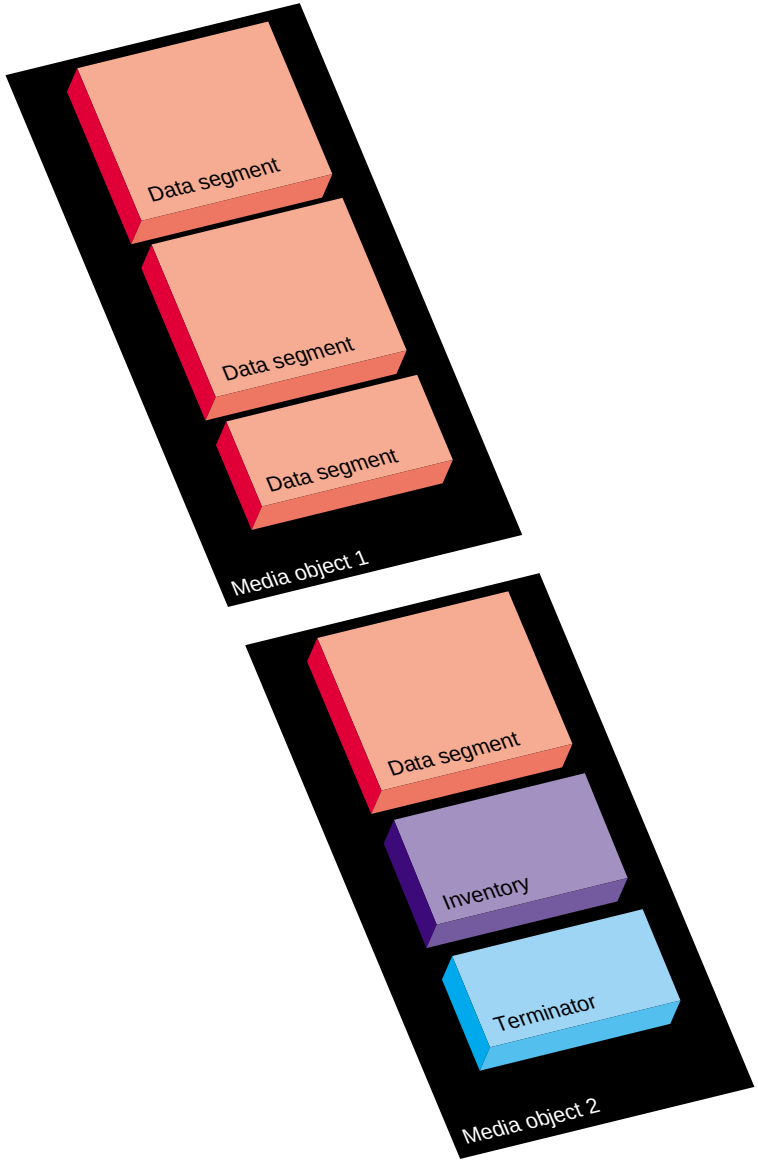

Figure 6-2 Single Dump on Multiple Media Objects

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 6-3 illustrates the layout of media files for two dumps on a single media object.

Figure 6-4 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.

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

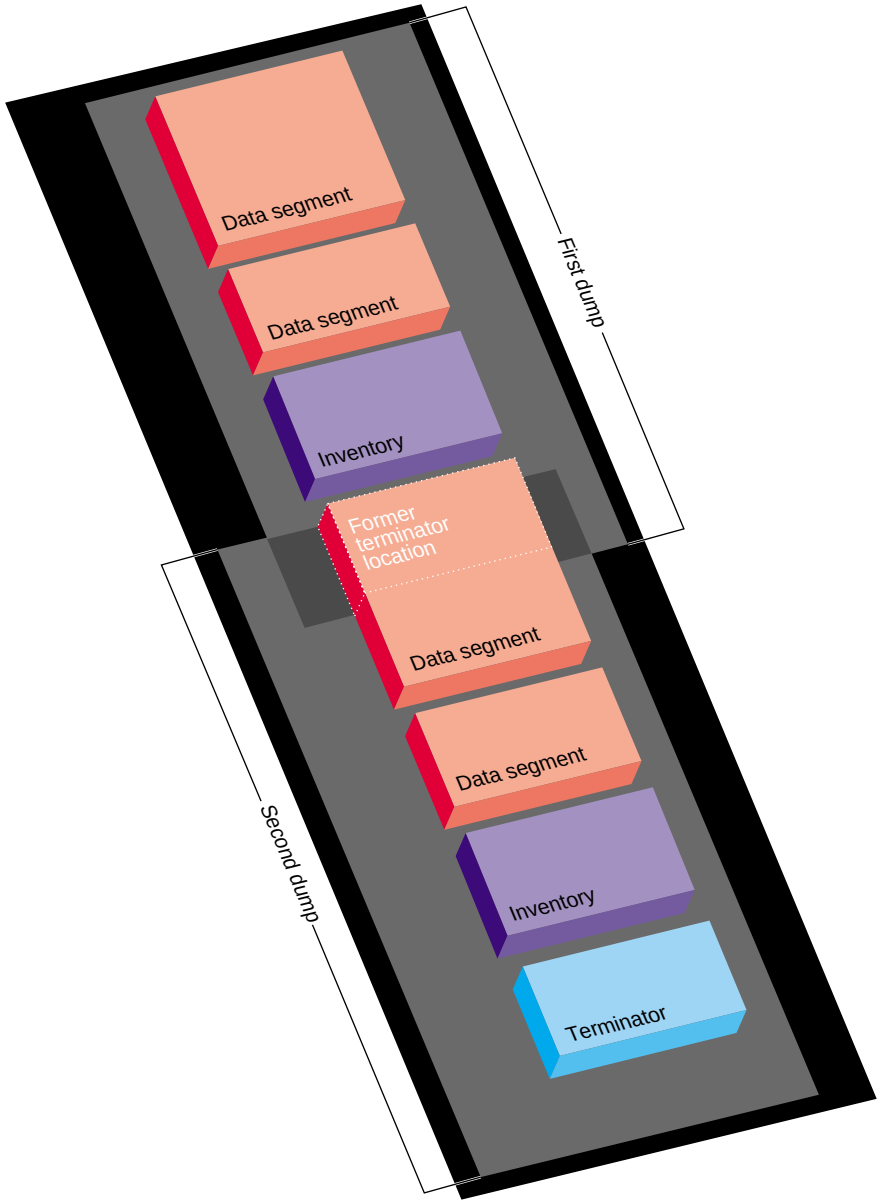

Figure 6-3 Multiple Dumps on Single Media Object

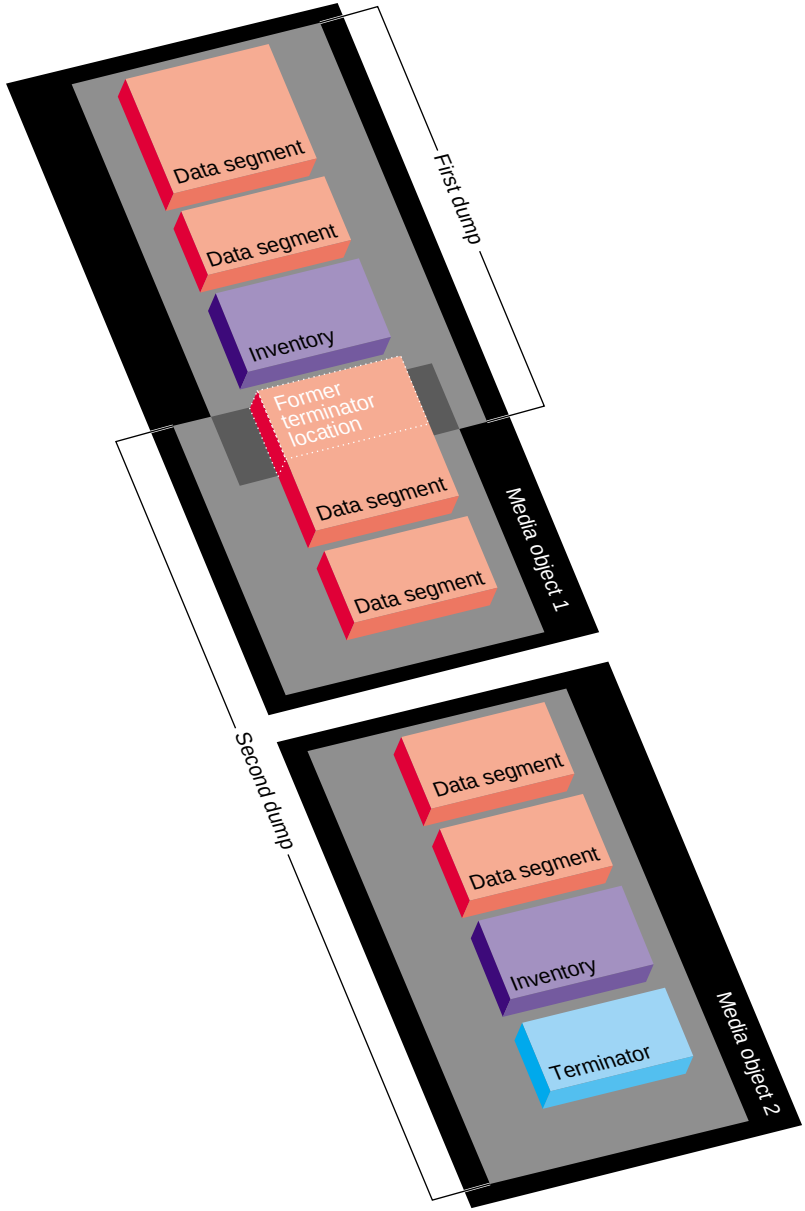

Figure 6-4 Multiple Dumps on Multiple Media Objects

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 [ -b <blocksize> (with minimal rmt option) ]
                        [ -c <media change alert program> ]
                        [ -f <destination> ... ]
                        [ -h (help) ]
                        [ -l <level> ]
                        [ -m <force usage of minimal rmt> ]
                        [ -o <overwrite tape > ]
                        [ -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 with xfsdump

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 ""About Incremental and Resumed Dumps" on page 41 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 with xfsdump

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 with xfsdump

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 60 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 with xfsdump

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.

About 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“Performing Cumulative Restores with xfsrestore” on page 54).

Performing an Incremental xfsdump

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

```
# 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 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):

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

Refer to "Performing Cumulative Restores with xfsrestore" on page 54 for information on the proper procedure for restoring incremental dumps.

Performing a Resumed xfsdump

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:

```
# xfsdump -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---
```

About 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 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.

Performing Simple Restores with xfsrestore

A simple restore is a non-cumulative restore (for information on restoring incremental dumps, refer to "Performing Cumulative Restores with xfsrestore" on page 54[]). 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 44). 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
```

Performing 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 60"for an example of using the standard input option of `xfsrestore`.

Performing 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.)

Note: Interactive restore is not allowed when the `xfsrestore` source is standard input (STDIN).

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.

Performing 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 "Performing Cumulative Restores with xfsrestore" on page 54).

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.

About the 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).

XFS Linux and IRIX XFS

The XFS filesystem under Linux differs from the XFS filesystem under IRIX in the following ways:

- The interface with the kernel (system calls) for Access Control Lists (ACLs) and Extended Attributes (EAs) is different in XFS/Linux compared with IRIX. However, the user level libraries for both ACLs and EAs are exactly the same in Linux and IRIX. Thus ACL or EA application code can be exactly the same.
- The unwritten extent feature is not yet operational in XFS Linux and should be turned off when the `mkfs` command is executed (which is the default setting). If necessary, you can turn this feature off using the following argument to the `mkfs` command:

  ```
  -d unwritten=0
  ```

- Linux `xfsdump/xfsrestore` does not support:
  - multiple tape devices using multiple `-f` options
  - IRIX dump option `-z` (prune large files)
  - DMAPI related options of `-a` and `-D`

  Linux `xfsdump/xfsrestore` does, however, have the added capabilities of unrestricted use of the `-b` option for blocksize specification and remote dumping/restoring between Linux and IRIX hosts.
- Linux XFS filesystems do note support project quotas.
- Under IRIX, external logs are specified using a volume manager, `xlv` or XVM. Under Linux, external logs can be specified with extensions to the `-l` option of `mkfs`.
- The `xlv` volume manager is not available with Linux XFS. Volume support in Linux XFS is through standard Linux volume managers `lvm` and `md` and through SGI's XVM logical volume manager.
- The `quotactl`(2) system call interface differs between IRIX and Linux, and the Linux quota tools differ in a number of ways to their IRIX counterparts

- Linux XFS does not support real-time subvolumes or Guaranteed-Rate I/O (GRIO).
- The `mkfs` command in Linux XFS uses the Version 2 directory format by default. Use of Version 1 is not currently supported.
- Linux XFS supports mounting by label and mounting by `uuid`.

Index

A

Access Control Lists (ACLs), 61
allocation groups, 6
attributes, filesystem, 2

B

backup and restore
  commands, 2
  using xfsdump and xfsrestore, 29-60
block size, 1, 3
  default, 3
  filesystem directory, 4
  maximum, 3
  minimum, 3
  news servers, 3
  range of sizes, 1

C

conventions, typographical, xvi

D

disk partitioning, 7
disk quotas, 21-28
  group, 22, 24
  hard limits, 21
  project, 61
  soft limits, 21
  turning on for users, 22
  user, 23
dump stream, 30
dump, incremental, 41-43
dump, resumed, 41-43

E

edquota command, 23, 24
erasing tape data, 41
extended attributes, 2
extents, XFS filesystem, 1
external filesystem log, 4

F

fcntl system call, 1
filesystem
  allocation groups, 6
  corruption, 13
  creation, 9
  extents, 1
  log size, 5
  mounting, 11
  reorganization, 13
  stripe units, 6
filesystem log

external, 4
internal, 5
size, 4
type, 4
font conventions, xvi
fstab file, 11, 22, 23, 27

G

gqnoenforce
fstab file, 27
option to mount command, 27
gquota
command, 23
fstab file, 23
growing XFS filesystems, 11
Guaranteed Rate I/O (GRIO), 62

H

hard limits, quotas, 21
hardware requirements, 3
housekeeping subdirectory, 59

I

incremental dumps, 41-43
inodes, 1, 2, 17, 18
internal filesystem log, 5
IRIX operating system, 61

J

journaling, 1

L

log
recovery, 20
size, 4, 5
type, 4
lost+found directory, 17, 18

M

media layout, xfsdump utility, 30
mkfs command, 9, 10, 11
mounting a filesystem, 11
mt erase command, 41

O

orphanage subdirectory, 59

P

panic, system, 13
partitioning, 7
prerequisite hardware, 3
project quotas, 61

Q

qnoenforce
fstab file, 27
option to mount command, 27
quota
command, 25, 28
fstab file, 22
quotaactl system call, 61

quotaoff command, 27
quotaon command, 27
quotas
  See disk quotas

R

readdr operation, 4
real-time subvolumes, 62
repquota command, 25, 26
resumed dumps, 41-43

S

soft limits, quotas, 21
stripe units, 6
system panic, 13

T

tape data, erasing, 41

U

unwritten extents, 61
uuid, mounting, 62

X

XFS filesystem
  attributes, 2
  block size, 1, 3
  creation, 9
  description, 1, 3
  extents, 1
  features, 1, 3
  growing, 11
  IRIX, 61
  log size, 4
  log type, 4
xfs_check command, 14
xfs_fsr command, 13
xfs_repair command, 14-20
  error messages, 17
xfsdump utility, 2, 29-60
  archives, 44
  copying filesystems, 60
  file backup, 39
  filesystem copy, 60
  inventory, 44
  local media, 37
  media layout, 30
  remote tape drive, 38
  syntax, 36
  tape reuse, 40
xfsdump_quotas file, 26
xfsdump_quotas_group file, 26
xfsrestore utility, 2, 29-60
  cumulative restore, 54
  files, 51
  housekeeping subdirectory, 59
  interactive restore, 52
  interruption, 58
  network restore, 52
  orphanage subdirectory, 59
  simple restore, 49
xlv volume manager, 61
XVM volume manager, 61