LSC File System Administrator's Guide

Sun Microsystems, Inc.
Part Number 816-1786-10

LSC Publication Number SG-0006
Revision 3.5.0-20

1270 Eagan Industrial Road, Suite 160, Eagan, Minnesota 55121-1231 U.S.A.

New Features

The *LSC File System Administrator's Guide*, publication SG-0006, was derived from the *SAM-FS System Administrator's Guide*, publication SG-0001, and the *QFS Administrator's Guide*, publication SG-0004. Revision 3.5.0-20 supports the QFS, SAM-FS, and SAM-QFS 3.5.0-20 releases running on the Solaris 2.6, 2.7, and 2.8 platforms.

The LSC 3.5.0-20 releases support the following new features in the QFS, SAM-FS, and SAM-QFS environments:

- Solaris 2.8 operating system. LSC products can now be installed on servers running the Solaris 2.8 operating system.
- SAN-QFS file system. The QFS file system can be used in conjunction with fiber-attached devices in a Storage Area Network (SAN). When the SAN-QFS license key from LSC is enabled, the QFS file system enables high-speed access to data using software such as Tivoli SANergy File Sharing.
- ACSLS 5.4. The SAM-FS and SAM-QFS environments now support the StorageTek ACSLS 5.4 release for network-attached StorageTek automated libraries.
- New devices. The SAM-FS and SAM-QFS environments now support the following devices:

Device	Comments
Exabyte Mammoth-2 drive	Equipment type `xm` in your LSC `mcf` file.
Fujitsu M8100 drive	Equipment type `fd` in your LSC `mcf` file.
Sony DTF-2 drive	Equipment type `so` in your LSC `mcf` file.
Sony network-attached automated library and drives	Sites with Sony network-attached automated libraries use the Sony DZC-8000S interface. Such automated libraries are equipment type `pe` in your LSC `mcf` file. The drives are equipment type `so` in your `mcf` file. Sites with automated libraries attached through this interface need to add the `LSCsony` (`samsony`) package at installation time.
StorageTek 9940 drive	Equipment type `sf` in your LSC mcf file.

- Documentation restructuring. The LSC manual set has been restructured in order to make the documentation more modular. Certain parts of the *SAM-FS System Administrator's Guide*, publication SG-0001, have been moved into new manuals or man pages.

  The following table indicates the topic that was affected by this restructuring, where it used to reside in the LSC documentation set prior to the 3.5.0-20 release, and where it resides in the documentation set as of the 3.5.0-20 release:

Topic	Pre-3.5.0-20 Location	3.5.0-20 Location
QFS installation and configuration.	*QFS Administrator's Guide*, publication SG-0004.	*QFS, SAM-FS, and SAM-QFS Installation and Configuration Guide*, publication SG-0007.

Topic	Pre-3.5.0-20 Location	3.5.0-20 Location
SAM-FS installation and configuration.	*SAM-FS System Administrator's Guide*, publication SG-0001, chapter 2.	*QFS, SAM-FS, and SAM-QFS Installation and Configuration Guide*, publication SG-0007.
QFS file system overview, reference, and operations information.	*QFS Administrator's Guide*, publication SG-0004.	*LSC File System Administrator's Guide*, publication SG-0006.
SAM-FS file system overview, reference, and operations information	*SAM-FS System Administrator's Guide*, publication SG-0001, chapter 5 and part of chapter 14.	*LSC File System Administrator's Guide*, publication SG-0006.
Application Programmer Interface (API) overview	*SAM-FS System Administrator's Guide*, publication SG-0001, chapter 12.	`intro_libsam`(3) man page.
Storage and archive management operational, reference, and disaster recovery information	*SAM-FS System Administrator's Guide*, publication SG-0001, chapters 3, 4, 6, 7, 8, 9, 10, 11, 13, and 14; appendixes A and B.	*SAM-FS and SAM-QFS Storage and Archive Management Guide*, publication SG-0008

Record of Revision

Revision	Description
3.5.0	October 2000. Original printing.
3.5.0-20	November 2000. Supports the LSC 3.5.0-20 releases.

Table of Contents

Appendix A - LSC Product Support A-1

Preface

This manual describes the file systems supported by LSC in the QFS, SAM-FS, and SAM-QFS 3.5.0-20 releases. This manual describes the file systems used in these products because they are technologically similar. Within this manual, differences are noted when necessary. For information on the storage and archive management (SAM) software, see the *SAM-FS and SAM-QFS Storage and Archive Management Guide*, publication SG-0008.

The LSC file systems are as follows:

- SAM-FS file system. The SAM-FS environment includes the standard LSC file system along with the LSC storage and archive manager, SAM. The SAM-FS environment's file system allows data to be migrated to automated libraries at device-rated speeds. The file system in the SAM-FS environment is a complete file system. The user is presented with a standard file system interface and can read and write files as though they were all on primary disk storage.
- QFS and SAM-QFS file systems. The QFS file system can be used as a standalone file system or it can be used in conjunction with the storage and archive manager, SAM. When used in conjunction with SAM, it is known as SAM-QFS. QFS shares most of the SAM-FS file system's features. The QFS file system, however, is designed for high performance and contains more features than the standard LSC file system supported within the SAM-FS environment.

This manual, the *LSC File System Administrator's Guide*, publication SG-0006, is written for system administrators responsible for setting up and maintaining LSC file systems. You, the system administrator, are assumed to be knowledgeable about Solaris operating system procedures, including installation, configuration, creating accounts, performing system backups, and other basic Solaris system administrator tasks.

This manual is organized as follows:

Section	Title
Chapter 1	Overview
Chapter 2	File system design
Chapter 3	Volume management and configuration
Chapter 4	File system operations
Chapter 5	Advanced topics
Chapter 6	Performance topics

Section	Title
Appendix A	LSC Product Support

In addition to the preceding sections, the glossary section defines terms used in LSC documentation.

Conventions

The following conventions and terms are used throughout this manual:

Convention	Meaning
`Courier`	The fixed-space courier font denotes literal items such as commands, files, path names, system prompts, system output, and messages. For example: `/etc/opt/LSCsamfs/mcf`
`Bold courier`	The **`bold courier`** font denotes text you enter at the shell prompt. For example: `server#` **`sls –D`**
[ ]	Brackets enclose optional portions of commands or optional arguments to commands.
Italic	Italics indicate either a variable or a term being defined. For a variable, you must replace the variable with a real name or value. For example: `server#` **`mount`** *mnt_pt*
|	The pipe symbol indicates that one of two or more optional arguments can be specified.
Volume	A named area on a cartridge for storing data. A cartridge has one or more volumes. Double-sided cartridges have two volumes, one on each side.

Other LSC Publications

In addition to this manual, the following LSC publications might be useful to you:

- *Migration Toolkit Guide*, publication SG-0002
- *SAM-Remote Administrator's Guide*, SG-0003
- *SAM-FS Man Page Reference Manual*, publication SR-0005
- *LSC File System Administrator's Guide*, publication SG-0006
- *QFS, SAM-FS, and SAM-QFS Installation and Configuration Guide*, SG-0007
- *SAM-FS and SAM-QFS Storage and Archive Management Guide*, SG-0008
- *Peripherals and Third-Party Software Supported*, LSC URL `http://www.lsci.com/lsci/services/supported-peripherals.shtml`

To order additional manuals, please send us a request using one of the methods described in the "Reader Comments" subsection.

Other File System Publications

In addition to publications from LSC, the following publications on UNIX file systems might interest you:

- Filesystem Hierarchy Standard (FHS) web pages at the following URL:
  `http://www.pathname.com/fhs/2.0/fhs-toc.html`
- Sun Microsystems online documentation web pages at the following URL:
  `http://docs.sun.com`

Licensing

Licenses for LSC products can be obtained from LSC. In some cases, the capabilities that these additional licenses can provide are described in this document because these products can be used in conjunction with this product. For information on obtaining licenses for LSC software, contact your sales representative, your Authorized Service Provider (ASP), or LSC.

The following LSC products are licensed separately:

- Migration Toolkit
- QFS standalone
- SAM-FS
- SAM-QFS
- SAM-Remote client
- SAM-Remote server
- SAM-Segment
- SAN-QFS

This document and the programs described in it are furnished under license from LSC and cannot be used, copied, or disclosed without prior approval from LSC in accordance with such license.

Reader Comments

If you have comments about the technical accuracy, content, or organization of this document, please let us know. We value your comments and will respond to them promptly. You can contact us by one of the following methods:

- Send email to `info@lsci.com`
- Send a facsimile (FAX) with your comments to "Technical Publications" in Eagan, Minnesota. Our fax number is: +1-651-554-1540
- Send written comments to the following address:

  LSC, Inc.
  Publications Department
  1270 Eagan Industrial Road
  Suite 160
  Eagan, MN 55121-1231
  USA

Overview
Chapter 1

The LSC file systems are dynamic and configurable file systems that present a standard UNIX file system interface to users. LSC licenses several different products. The individual components of the LSC product line can be combined in several different ways. The following list shows how various LSC file systems can be used or combined with LSC's storage and archive management software:

LSC Product	Components
QFS	QFS standalone file system
SAM-QFS	QFS file system plus the storage and archive management utility, SAM
SAM-FS	LSC standard file system plus the storage and archive management utility, SAM

While technologically similar, there are differences between each file system. This chapter presents an overview of the features common to all LSC file systems, highlights the features that differentiate the file systems, and explains the commands available with each file system. Specifically, this chapter is divided into the following subsections:

- Features common to all file systems
- File system differences
- Commands

Features Common to all File Systems

The LSC file systems do not require changes to user programs, nor are changes required to the UNIX kernel. All LSC file systems share the following features:

- `vnode` interface. LSC file systems are implemented using the standard Solaris virtual file system (`vfs/vnode`) interface. The kernel intercepts all requests for files, including those that reside in LSC file systems. If the file is identified as an LSC file, the request is passed to the LSC file system. The LSC file system handles all requests for LSC files. All LSC file systems are identified as type `samfs` in the `/etc/vfstab` file and on the `mount`(1M) command.

  By using the `vfs/vnode` interface, LSC file systems work with the standard Solaris kernel and require no modifications within the kernel for file management support. Thus, the file system is protected from operating system changes and does not

normally require extensive regression testing when the operating system is updated or improved.

- Enhanced volume management. LSC file systems support both striped and round-robined disk access. The master configuration file (`mcf`) specifies the volume management features and lets the file system know the topology of the devices it controls. This is in contrast to most UNIX file systems that can address only one device or one portion of a device. LSC file systems do not require any additional volume management applications.

  The LSC integrated volume management features use the Solaris physical device drivers to pass I/O requests to and from the underlying devices. The LSC volume manager groups physical devices into family sets upon which each file system is created.

- Support for paged and direct I/O. LSC file systems support two different types of I/O: paged (also called *cached* or *buffered* I/O) and direct.

  When paged I/O is used, user data is cached in virtual memory pages and then written to disk by te Solaris Virtual Memory Manager (`vm`). LSC uses standard Solaris interfaces to manage paged I/O. This is the default type of I/O.

  When direct I/O is used, user data is written directly to disk cache. Direct I/O can be specified by using the Solaris `directio`(3C) function call. Large block, sequential, aligned I/O can realize substantial performance improvements by using direct I/O.

- Preallocation of file space. For fast sequential reads and writes, contiguous disk space can be preallocated by using the `setfa`(1) command.

- Interoperability with Application Programming Interface (API) routines. For example, these routines allow your program to preallocate contiguous disk space, access a specific striped group, and perform other operations.

- Unlimited capacity. You are virtually unlimited with regard to file size, the number of files that can reside in a file system, and the number of file systems you can specify.

  LSC file systems support files of up to 2^{64}-1 bytes in length. Such very large files can be striped across many disks or RAID devices, even within a single file system. This is true because LSC file systems are fully 64-bit compatible.

  The number of file systems you can configure is virtually unlimited. Each file system supports up to 252 disk partitions for virtually unlimited storage capacity.

  There is no predefined limit for the number of files on a SAM-FS file system. Because the inode space (which holds information about the files) is dynamically allocated, the maximum number of files is limited only by the amount of disk storage comprising the file system. The inodes are cataloged in the `.inodes` file under the mount point. The `.inodes` file requires 512 bytes of storage per file.

  For a QFS or SAM-QFS file system, the inodes are located on the metadata device(s) and are separated from the file data devices. The number of files in these file systems

is limited by the size of the metadata (`mm`) devices, and you can increase the size of the file system by adding more metadata devices.

- Support for special devices. Support for special devices is available for command interfaces that require special devices.

- Fast file system recovery. One of the key functions of a file system is to recover quickly after an unscheduled outage. Standard UNIX file systems require a lengthy file system check (`fsck`(1M)) to repair inconsistencies after a system failure. Running the `fsck`(1M) command on a terabyte-sized file system can take hours.

  LSC file systems often do not require file system checks after an unscheduled outage. In addition, they recover from system failures without using journaling. They accomplish this dynamically by using identification records, serial writes, and error checking for all critical I/O operations. After a system failure, an LSC file system can be remounted immediately, even for a terabyte-sized file system.

File System Differences

The LSC file systems share the common features described in the previous subsection. This subsection describes the areas in which they differ. One area of difference is speed. The QFS and SAM-QFS file systems provide the ability to attain raw, device-rated disk speeds with the administrative convenience of a file system.

The following list notes other ways in which the file systems differ:

- Disk allocation unit (DAU). The DAU is the basic unit of online storage. LSC's adjustable DAU is useful for tuning the file system with the physical disk storage device and for eliminating the system overhead caused by read-modify-write operations.

  The SAM-FS file system uses several sizes. The small DAU is 4 kilobytes. The large DAU is configurable into 16-, 32-, or 64-kilobyte units. The available DAU size pairs are 4/16 (default), 4/32, and 4/64.

  The QFS and SAM-QFS file systems support a fully configurable DAU, sized from 16 kilobytes (default) through 65528 kilobytes. The DAU you specify must be multiple of 8 kilobytes. There is no small DAU in a QFS or SAM-QFS file system.

- Metadata storage. File systems use metadata to reference disk blocks on a disk device. Typically, metadata resides on the same device as the file data. This is true for the SAM-FS file system. The SAM-FS file system maintains file metadata information in separate files. This enables the number of files, and the file system as a whole, to be enlarged dynamically. The SAM-FS file system can span multiple partitions by using disk striping, thereby improving access to large files.

  The QFS and SAM-QFS file systems separate the file system metadata from the file data by storing them on separate devices. The QFS and SAM-QFS file systems allow you to define one or more separate metadata devices in order to

reduce device head movement and rotational latency, improve RAID cache utilization, or mirror metadata without mirroring file data.

- Support for multiple striped groups within a file system. In order to support multiple RAID devices in a single file system, striped groups can be defined in QFS and SAM-QFS file systems. Disk block allocation can be optimized for a striped group, reducing the overhead for updating the on-disk allocation map. Users can assign a file to a striped group either through an API routine or by using the `setfa`(1) command.
- SAM interoperability. The SAM-QFS and SAM-FS file systems combine LSC file system features with LSC's storage and archive management utility, SAM. Users can read and write files directly from magnetic disk, or they can access archive copies of files as though they were all on primary disk storage.

  When possible, LSC products use the standard Solaris disk and tape device drivers. For devices not directly supported under Solaris, such as certain automated library and optical disk devices, LSC provides special device drivers in the SAM-FS and SAM-QFS software packages.

Commands

Specialized LSC file system commands are included in the QFS, SAM-FS, and SAM-QFS environments. These commands operate in conjunction with the standard UNIX file system commands. Some commands are specific to only one or two of these environments. All the commands are documented in UNIX `man`(1) pages.

The following subsections show the commands supported within each environment.

User Commands

By default, LSC file system operations are transparent to the end user. Depending on your site practices, however, it may be desirable to make some commands available to users at your site. The following commands can help users fine tune certain operations:

Command	Description	Used By
`archive`(1)	Sets archive attributes on files.	SAM-FS, SAM-QFS
`exarchive`(1)	Manipulates (exchanges) archive copies.	SAM-FS, SAM-QFS
`release`(1)	Releases disk space and sets release attributes on files.	SAM-FS, SAM-QFS
`request`(1)	Creates a removable media file.	SAM-FS, SAM-QFS

Command	Description	Used By
`sdu`(1)	Summarizes disk usage. The `sdu`(1) command is based on the GNU version of the `du`(1) command and has been extended for use with LSC file systems.	QFS, SAM-FS, SAM-QFS
`segment`(1)	Sets segment file attributes. Segmented file features are enabled through the SAM-Segment license key.	SAM-FS, SAM-QFS
`setfa`(1)	Sets file attributes.	QFS, SAM-FS, SAM-QFS
`sfind`(1)	Searches for files in a directory hierarchy. The `sfind`(1) command is based on the GNU version of the `find`(1) command and contains options for displaying file system options.	QFS, SAM-FS, SAM-QFS
`sls`(1)	Lists contents of directories. The `sls`(1) command is based on the GNU version of the `ls`(1) command and contains options for displaying file system attributes and information.	QFS, SAM-FS, SAM-QFS
`ssum`(1)	Sets the checksum attributes on files.	SAM-FS, SAM-QFS
`stage`(1)	Copies offline files to disk and sets stage attributes on files.	SAM-FS, SAM-QFS

General System Administrator Commands

The file system provides system management and maintenance commands for administering the system. The following table summarizes the general system administrator commands:

Command	Description	Used By
`samcmd`(1M)	Executes a single operator utility command.	SAM-FS, SAM-QFS
`samd`(1M)	Starts or stops daemons.	SAM-FS, SAM-QFS
`samfsinfo`(1M)	Displays information about a file system.	QFS, SAM-FS, SAM-QFS
`samu`(1M)	Invokes the curses-based operator interface. Displays the status of devices and allows the operator to control automated libraries.	SAM-FS, SAM-QFS

File System Commands

The file system commands are used to maintain file system operations. These commands are as follows:

Command	Description	Used By
`mount`(1M)	Mounts a file system. The man page name for this command is `mount_samfs`(1M).	QFS, SAM-FS, SAM-QFS
`qfsdump`(1M) `qfsrestore`(1M)	Creates/restores a dump file of the control structures associated with a QFS file system.	QFS
`samfsck`(1M)	Checks and repairs a file system.	QFS, SAM-FS, SAM-QFS
`samfsdump`(1M) `samfsrestore`(1M)	Creates/restores a dump file of the control structures associated with a SAM-FS or SAM-QFS file system.	SAM-FS, SAM-QFS
`samgrowfs`(1M)	Expands a file system by adding disk partitions.	QFS, SAM-FS, SAM-QFS
`sammkfs`(1M)	Makes a new file system from disk partitions.	QFS, SAM-FS, SAM-QFS
`samncheck`(1M)	Returns a full path name given the mount point and inode number.	QFS, SAM-FS, SAM-QFS

Additional LSC Commands

LSC also provides the following additional types of commands for use in the SAM-FS and SAM-QFS environments:

- Automated library commands
- Archiver, stager, releaser, and recycler commands
- Specialized maintenance commands
- Operational utility commands

The preceding commands are described on individual man pages and in the *SAM-FS and SAM-QFS Storage and Archive Management Guide*, publication SG-0008.

File System Design
Chapter 2

Good file system design allows you to quickly access information as well as recover information when necessary. There are a number of factors to consider when configuring an LSC file system.

This chapter presents the following topics:

- Design basics
- Inode files and user settings
- Shared reader/shared writer (QFS file systems only)
- Specifying disk allocation units (DAUs) and stripe widths
- File allocation methods

Design Basics

LSC file systems are multithreaded, advanced storage management systems. To take maximum advantage of these capabilities, create multiple file systems whenever possible.

Like UFS, LSC file systems use a linear search method when performing directory lookups. They search from the beginning of the directory to the end. As the number of files in a directory increases, the search time through the directory increases. Users who have directories with thousands of files can experience excessive search times. These search times are also evident when you restore a file system. To increase performance and speed up file system dumps and restores, you should keep the number of files in a directory under 4000.

Inode Files and File Characteristics

The types of files to be stored in a file system can affect how the file system is designed. An inode is a 512-byte block of information that describes the characteristics of a file or directory. This information is allocated dynamically within the file system.

The inodes are stored in the `.inodes` file located under the file system mount point. The SAM-FS `.inodes` file resides on the same physical devices as the file data and is interleaved with the file data. In contrast, a QFS or SAM-QFS `.inodes` file resides on metadata devices and is separate from the file data devices.

Like a standard Solaris inode, an LSC file system inode contains the file's POSIX standard inode times: file access, file modification, and inode changed times. The LSC file systems add a creation time, attribute change time, and a residence time. To summarize, the following times are recorded in the inode:

Time	Incident
`access`	Time the file was last accessed. POSIX standard.
`modification`	Time the file was last modified. POSIX standard.
`changed`	Time the inode information was last changed. POSIX standard.
`attributes`	Time the attributes specific to LSC file systems were last changed. LSC extension.
`creation`	Time the file was created. LSC extension.
`residence`	Time the file changed from offline to online or vice versa. LSC extension.

The attributes specific to LSC file systems include both user settings and general file states. The following two subsections describe these characteristics.

User Settings

A file's attributes are stored in its inode. These inode attributes can be displayed by using the `sls`(1) command with its `-D` option. For more information on `sls`(1) options, see the `sls`(1) man page.

A user can set attributes with the following user commands:

- `archive`(1).
- `ssum`(1).
- `release`(1).
- `segment`(1). Requires a SAM-Segment license.
- `setfa`(1).
- `stage`(1).

Users can also set attributes from within an application by using the following API routines:

- `sam_archive`(3).
- `sam_release`(3).
- `sam_segment`(3). Requires a SAM-Segment license.
- `sam_setfa`(3).
- `sam_ssum`(3)
- `sam_stage`(3)

The following attributes are listed in the inode:

Attribute	Definition	Used By
`archive –n`	The file is marked never archive. This attribute can be set by the superuser with the `archive`(1) command.	SAM-FS, SAM-QFS
`directio`	The file is marked for direct I/O. For more information, see the `setfa`(1) command's `–D` option in your Solaris documentation.	QFS, SAM-FS, SAM-QFS
`release –a`	This file is marked for release as soon as one archive copy is made. This attribute can be set from within the `archiver.cmd` file or by using the `release`(1) command.	SAM-FS, SAM-QFS
`release –n`	This file is marked never release. This attribute can be set from within the `archiver.cmd` file or by using the `release`(1) command. To set this from the `release`(1) command, you must be superuser.	SAM-FS, SAM-QFS
`release –p`	The file is marked for partial release. The `partial=`*n*`k` option specifies that the first *n* kilobytes of disk space be retained in disk cache (on the disk) for the file. This attribute can be set from within the `archiver.cmd` file or by using the `release`(1) command.	SAM-FS, SAM-QFS
`segment` *x*`m` `stage_ahead` *y*	The file is marked for segmentation. Requires a SAM-Segment license. The *x*`m` notation indicates that the segment is *x* megabytes in size. The `stage_ahead` *y* attribute indicates the number of attributes (*y*) to be staged ahead. These attributes can be set by using the `segment`(1) command.	SAM-FS, SAM-QFS
`stage –a`	The file is marked for associative staging. This attribute can be set from within the `archiver.cmd` file or by using the `stage`(1) command.	SAM-FS, SAM-QFS

Attribute	Definition	Used By
`stage –n`	The file is marked never stage. This signifies direct access to removable media cartridges. This attribute can be set from within the `archiver.cmd` file or by using the `stage`(1) command. To set this from the `stage`(1) command, you must be superuser.	SAM-FS, SAM-QFS

All the preceding attributes can be set on directories, too. Files that are written to a directory after an attribute is set inherit all the directory attributes at the time of the write. Files written before an attribute is applied to the parent directory do not inherit that directory attribute.

Users can gather information on file attributes by using the LSC `sls`(1) command, which is described later in this chapter under the heading “Displaying File Information”.

File States

The QFS, SAM-FS, and SAM-QFS file systems set various states for a file. These states are listed in the inode and are as follows:

Attribute	Definition	Used By
`archdone`	Indicates that the file’s archive requirements have been met. There is no more work archiver can do on the file. Note that `archdone` does not necessarily indicate that the file has been archived. This attribute is set by the archiver and cannot be set by a user.	SAM-FS, SAM-QFS
`damaged`	The file is damaged. This attribute is set by the stager or by the `samfsrestore`(1M) command. You can use the `undamage`(1M) command to reset this attribute to undamaged.	SAM-FS, SAM-QFS
`offline`	The file data has been released. This attribute is set by the releaser. This attribute can also be set by using the `release`(1) command.	SAM-FS, SAM-QFS

Users can gather information on file states by using the LSC `sls`(1) command, which is described in the following subsection, “Displaying File Information”.

Displaying File Information

The LSC `sls`(1) command extends the standard UNIX `ls`(1) command and provides more information about a file. The following example shows the detailed output for the `sls`(1) command. It displays the inode information for the file `hgc2`:

```
hgc2:
  mode: -rw-r--r--  links:   1  owner: root      group: other
  length:     14971  inode:       30
  archdone;
  segments 3, offline 0, archdone 3, damaged 0;
  copy 1: ---- Jun 13 17:14     2239a.48   lt MFJ192
  copy 2: ---- Jun 13 17:15      9e37.48   lt AA0006
  access:      Jun 13 17:08  modification: Jun 13 17:08
  changed:     Jun 13 17:08  attributes:   Jun 13 17:10
  creation:    Jun 13 17:08  residence:    Jun 13 17:08
```

The first line, which begins with `mode:`, indicates the file's mode or permissions, the number of hard links to the file, the owner of the file, and the group to which the owner belongs.

The second line, which begins with `length:`, indicates the file's length in bytes and the inode number.

The third line, which begins with `archdone;`, displays file attributes specific to the QFS, SAM-FS, or SAM-QFS file system. For more information on this line, see the `sls`(1) man page.

The fourth line does not appear unless the file is a segment index. This requires a SAM-Segment license. The general format for this line is as follows:

`segments` *n*`, offline` *o*`, archdone` *a*`, damaged` *d*`;`

This line indicates that there are *n* data segments. There are *o* data segments offline. There are *a* data segments that have met their archiving requirements. The number of damaged data segments is *d*.

The fifth line, which begins with `copy 1:`, is the first archive copy line. This line appears in `sls`(1) output in SAM-FS and SAM-QFS environments. One archive copy line is displayed for each active or expired archive copy. The fields in the archive copy lines are as follows:

- The first field indicates the archive copy number.

- The second field contains 4 indicators, each of which is either a dash (`-`) or a letter. Reading them from left to right, the indicators convey the following information:

Dash (`-`) Position	Meaning
1	Indicates either a stale or active entry. An `S` indicates that the archive copy is stale. That is, the file was modified and this archive copy is a previous version of the file. A `U` indicates that the copy has been unarchived. *Unarchiving* is the process by which archive entries for files or directories are deleted A dash (`-`) indicates that the archive copy is active and valid.
2	Indicates whether the archive copy is to be rearchived. An `r` indicates that the archive copy is scheduled to be rearchived by the archiver. A dash (`-`) indicates that the archive copy is not to be rearchived by the archiver.
3	Unused.
4	Indicates whether the copy is damaged or undamaged. A `D` indicates that the archive copy is damaged. The archive copy is not a candidate for staging. A dash (`-`) indicates that the archive copy is not damaged. It is a candidate for staging.

- The third field shows the date and time the archive copy was written to the archive media.
- The fourth field contains two hexadecimal numbers separated by a decimal point (`.`). The first hexadecimal number (`2239a`) indicates the position of the beginning of the archive file on the cartridge. The second hexadecimal number (`48`) is the file byte offset (divided by 512) of this copy in the archive file.
- The fifth and sixth fields in the archive copy line indicate the media type and the Volume Serial Name (VSN) where the archive copy resides.

The sixth line, which begins with `copy 2:` lists characteristics of the second archive copy. This line appears in `sls`(1) output in SAM-FS and SAM-QFS environments.

The `checksum` line is displayed only if the file has a checksum-related attribute set (that is, generate, use, or valid). This line appears in `sls`(1) output in SAM-FS and SAM-QFS environments. The format of the checksum line is as follows:

```
checksum:  gen  use  val  algo:  1
```

The preceding line is displayed if all three checksum attributes are set for a file. If the `generate` attribute is not set, `no_gen` appears in place of `gen`. Similarly, if the `use` attribute is not set, `no_use` appears. `val` is displayed when the file has been archived and a checksum has been computed. If the file has not been archived and/or no checksum has been

computed, `not_val` appears. The keyword `algo` precedes the numeric algorithm indicator that specifies the algorithm that is used to generate the checksum value.

For additional information on `sls`(1) output, see the `sls`(1) man page.

Shared Reader/Shared Writer (QFS File System Only)

A *shared reader/shared writer* refers to the concept of having a file system that can be mounted and accessed by multiple servers. Multiple hosts can read from the file system, but only one host can write to the file system.

A shared reader is specified with the `-o shared_reader` option on the `mount`(1M) command. There is no limit to the number of file systems that can be mounted as shared readers. A shared reader file system is automatically mounted as read only.

The one-writer host is specified with the `-o shared_writer` option on the `mount`(1M) command. Only one file system can be mounted as a shared writer. If `-o shared_writer` is specified, directories are written through to disk at each change and files are written through to disk at close.

The QFS file system ensures that all servers that access the same file system can always access the current environment. When the writer closes a file, the QFS file system writes all information for that file to disk immediately. The reader should only access a file after the file is closed by the writer. These and other steps ensure that no server in a multiple reader/single writer QFS environment ever risks getting into an out-of-sync condition with the file system.

For more information on shared reader and shared writer file systems, see the `mount_samfs`(1M) man page.

Specifying Disk Allocation Units (DAUs) and Stripe Widths

Disk space is allocated in blocks. These are also called Disk Allocation Units (DAUs), which are the basic unit of online disk storage. While sectors and cylinders describe the physical disk geometry, the DAU describes the file system geometry. The appropriate DAU setting and stripe can improve performance and improve magnetic disk usage.

The DAU setting is the minimum amount of contiguous space that is used when a file is written. For example, assume that your DAU is set to 16 kilobytes and you have disabled striping by setting `stripe=0`. You are using round robin allocation (because of the `stripe=0` setting), and you have 2 files, as follows:

- The first file is a 15-kilobyte file. It occupies 1 DAU. The file data occupies 15 kilobytes of the DAU, and the other 1 kilobyte is not used.
- The second file is a 20-kilobyte file. It occupies 2 DAUs. The file data occupies all 16 kilobytes of the first DAU, and 4 kilobytes of the second DAU. The second DAU also contains 12 kilobytes that are not used.

The DAU setting is specified by the `-a` *allocation_unit* option on the `sammkfs`(1M) command.

If striped allocation is used, the *stripe width* determines the maximum number of DAUs written in one I/O event. This setting can be specified by the `-o stripe=`*n* option on the `mount`(1M) command, which is run after you have run the `sammkfs`(1M) command.

The following subsections describe how DAU settings and stripe widths can be configured.

DAU Settings and File System Geometry

Each file system can have its own DAU setting. Thus, several mounted file systems can be active on a server, each with a different DAU setting. Because the DAU setting is determined when the file system is created using the `sammkfs`(1M) command, this setting cannot be changed dynamically.

All LSC file systems allow you to specify your DAU settings. This adjustable DAU is useful for tuning the file system with the physical disk storage device. This eliminates the system overhead caused by the read-modify-write operation. Applications that manipulate very large files can benefit substantially from this feature. For an example that shows how to control the read-modify-write operation, see the `mount_samfs`(1M) man page's description of the `-o writebehind=`*n* option and the EXAMPLES section.

The possible DAU settings differ depending on your file system. The following subsections describe the DAU settings for each file system. These subsections also introduce the concept of the Master Configuration File (`mcf` file). You create this ASCII file at system installation time. It defines the devices and file systems used in your QFS, SAM-FS, or SAM-QFS environment. The `mcf` file is introduced in the following subsections, but it is more heavily discussed in chapter 3, "Volume Management".

SAM-FS DAU Settings

The SAM-FS file system, which is type `ms` in your `mcf` file, uses both a small DAU and a large DAU, as follows:

- The small DAU is always 4 kilobytes.
- By default, the large DAU is 16 kilobytes. The large DAU's size can be overridden when the file system is made by using the `-a` *allocation_unit* option to the `sammkfs`(1M) command. The large DAU specification can be 16, 32, or 64 kilobytes.

The available DAU size pairs are 4/16 (default), 4/32, and 4/64. Depending on the type of file data stored on the disks, selecting the larger DAU pairs can improve the file system performance significantly. For information on testing system performance, see the subsection on increasing file transfer performance in chapter 6, "Performance Topics".

When a file is created, the SAM-FS file system first writes the file to small DAUs; specifically, up to 8 4-kilobytes small DAUs are used initially. If more DAUs are needed, the SAM-FS file system writes the remainder of the file to one or more large DAUs. The result is better I/O performance for large files while minimizing the disk fragmentation that can result from having many small files.

QFS and SAM-QFS DAU Settings

The QFS and SAM-QFS file systems, which are type `ma` in your `mcf` file, use a single DAU size, but you can specify different DAU sizes by adjusting the default setting in 8-kilobyte block units. The DAU setting must be specified using the `-a` *allocation_unit* option to the `sammkfs`(1M) command. The following command specifies a DAU of 128 kilobytes:

```
server# sammkfs -a 128 samqfs1
```

For file data, the DAU setting can be anywhere from 16 kilobytes (the default) to 65528 kilobytes. For metadata, the DAU setting is 16 kilobytes.

Stripe Widths

Stripe width defaults differ between QFS, SAM-FS and SAM-QFS file systems. The stripe width is specified by the `-o stripe=`*n* option on the `mount`(1M) command. For all LSC file systems, however, if the stripe width is set to `0`, round-robin allocation is used.

The following subsections explain the differences that affect stripe widths on the various LSC file systems.

SAM-FS Stripe Widths

On SAM-FS file systems, the stripe width is set at mount time. If you do not specify a stripe width, the following defaults are used:

DAU	Default Stripe Width	Amount of Data Written to 1 Disk
16 kilobytes (default)	8	128 kilobytes
32 kilobytes	4	128 kilobytes
64 kilobytes	2	128 kilobytes

For example, if `sammkfs`(1M) is run with default settings, the default large DAU is `16` kilobytes. If the default stripe width is then used when the `mount`(1M) command is run, the stripe width set at mount time is `8`.

Note that if you multiply the number in the first column of the preceding list by the number in the second column of the preceding list, the resulting number is 128 kilobytes. LSC file systems operate more efficiently if the amount of data being written to disk is at least 128 kilobytes. The efficiency comes from lowered overhead and other factors.

QFS and SAM-QFS Stripe Widths – Not Using Striped Groups

On QFS and SAM-QFS file systems, the stripe width that is set at mount time depends on whether or not striped groups are configured. A *striped group* is a collection of devices that are striped as a group. For more information on striped groups, see "File Allocation Methods". This subsection describes stripe widths for QFS and SAM-QFS without stripe groups.

If striped groups are not configured, the DAU and stripe width relationships are similar to those for SAM-FS file systems. The differences being that a DAU of 128 kilobytes is possible and that the DAU is configurable in 8-kilobytes blocks.

By default, if no stripe width is specified, the amount of data written to disk is always at or near 128 kilobytes. Like other file systems, QFS and SAM-QFS are more efficient if write operations write at least one whole stripe per I/O request. If you do not specify a stripe width, the following defaults are used:

DAU	Default Stripe Width	Amount of Data Written to 1 Disk
16 kilobytes (default)	8	128 kilobytes
24 kilobytes	5	120 kilobytes
32 kilobytes	4	128 kilobytes
40 kilobytes	3	120 kilobytes
48 kilobytes	2	96 kilobytes
56 kilobytes	2	112 kilobytes
64 kilobytes	2	128 kilobytes
72 kilobytes	1	72 kilobytes
128 kilobytes	1	128 kilobytes
> 128 kilobytes	1	DAU size

QFS and SAM-QFS Stripe Widths – Using Striped Groups

If striped groups are configured for your QFS or SAM-QFS file system, the minimum amount of space allocated is the DAU multiplied by the number of devices in the striped group. The DAU can be very large when using striped groups. LSC recommends that the DAU be at least 128 kilobytes to make QFS and SAM-QFS more efficient.

When striped groups are used, data is written to several disk devices at once. This allocation treats a group of disks as if they were one device.

If striped groups are configured, data is written to each disk in disk cache using *hard striping*. Hard striping refers to writing to the same specific place on each disk when data is written.

QFS and SAM-QFS Data Alignment

Data alignment refers to matching the allocation unit of the RAID controller with the allocation unit of the file system. A mismatched alignment causes a read-modify-write operation.

The optimal QFS file system alignment formula is as follows:

allocation_unit = RAID stripe width X number of data disks

For example, if a RAID-5 unit has a total of 8 disks, with 1 of the 8 being the parity disk, the number of data disks is 7. If the RAID stripe width is 64 kilobytes, then the optimal allocation unit is 64 X 7 = 448 kilobytes.

Data files are striped or round-robined through each striped group (g*XXX*) or data disk (`mr`) defined within the same file system. By default, if you have multiple metadata devices, metadata is allocated using striped or round-robin allocation depending on the data disk setting. You can override this setting. For more information on striping the `.inodes` file, see chapter 5, "Advanced Topics".

The rest of this chapter provides more information for you to consider when setting DAUs and determining stripe widths.

File Allocation Methods

All LSC file systems allow you to specify both round-robined and striped allocation methods. These methods are as follows:

- Striped allocations spread the files across all devices configured in the file system. Striping is used when performance for one file requires the additive performance of all the devices. Striped disk access allows multiple I/O streams to simultaneously write a file across multiple disks, interlacing the file on logical disks. The size of the I/O transmission is determined by the DAU and the stripe width. By default, the file systems are striped.

  The QFS and SAM-QFS file systems enhance the concept of striped allocation by supporting *striped groups*. A striped group is a collection of devices within a QFS or SAM-QFS file system. Striped groups must be defined in the `mcf` file as g*XXX* devices. Striped groups allow one file to be written to and read from two or more devices. You can specify up to 128 striped groups within a file system.

- Round-robined allocation writes files to individual file system devices. Round-robined allocation is useful for multiple data streams because aggregate performance can exceed striping performance in this type of environment. Round-robined disk allocation allows a single file to be written to a logical disk. The next file is written to the next logical disk. I/O size is determined by the size of the file being written. Round-robined allocation can be explicitly specified in the `/etc/vfstab` file by entering `stripe=0`.

The following subsections describe round-robined allocation, striped allocation, and striped groups in more detail. At the end of this chapter there is an example that shows how to match your allocation method with the types of files in your file system.

Round-robined Allocation

The round-robined allocation method writes one data file at a time to each successive device in the family set. When the number of files written equals the number of devices defined in the family set, the file system starts over again with the first device.

If a file exceeds the size of the physical device, the first portion of the file is written to the first device, and the remainder of the file is written to the next device with available storage.

The following figures depict LSC file systems using round-robined allocation. In these figures, file 1 is written to disk 1, file 2 is written to disk 2, file 3 is written to disk 3, and so on. When file 6 is created, it is written to disk 1, starting the round-robined allocation scheme over again.

Figure 2-1 depicts a SAM-FS file system using round-robined allocation on five devices.

Figure 2-2 depicts a QFS or SAM-QFS file system using round-robined allocation on five devices.

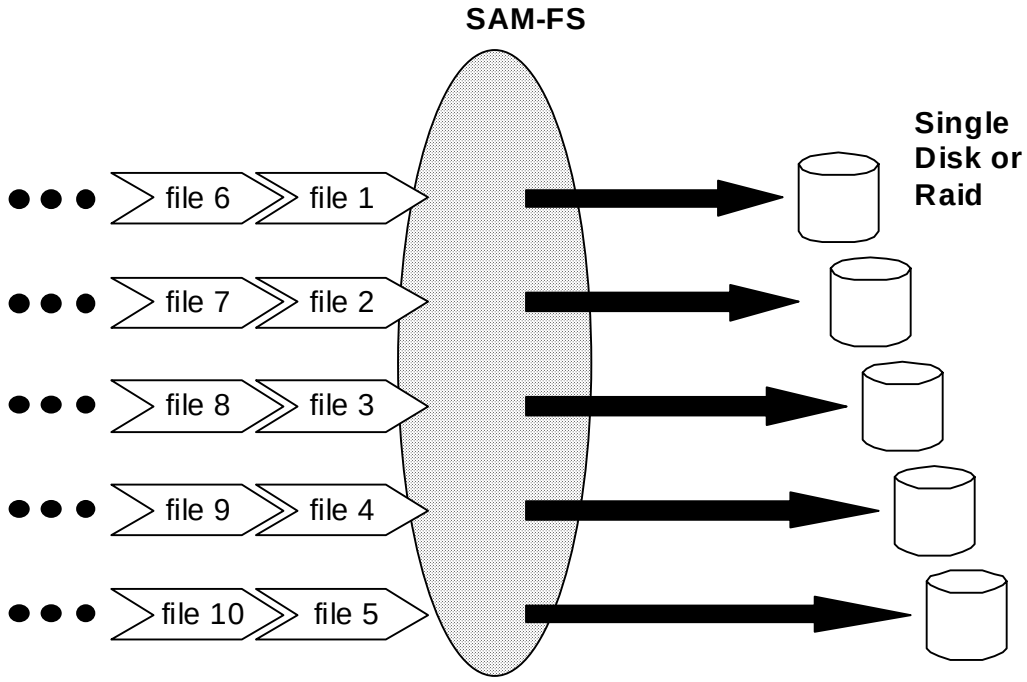

Figure 2-1. A round-robined SAM-FS file system using five devices

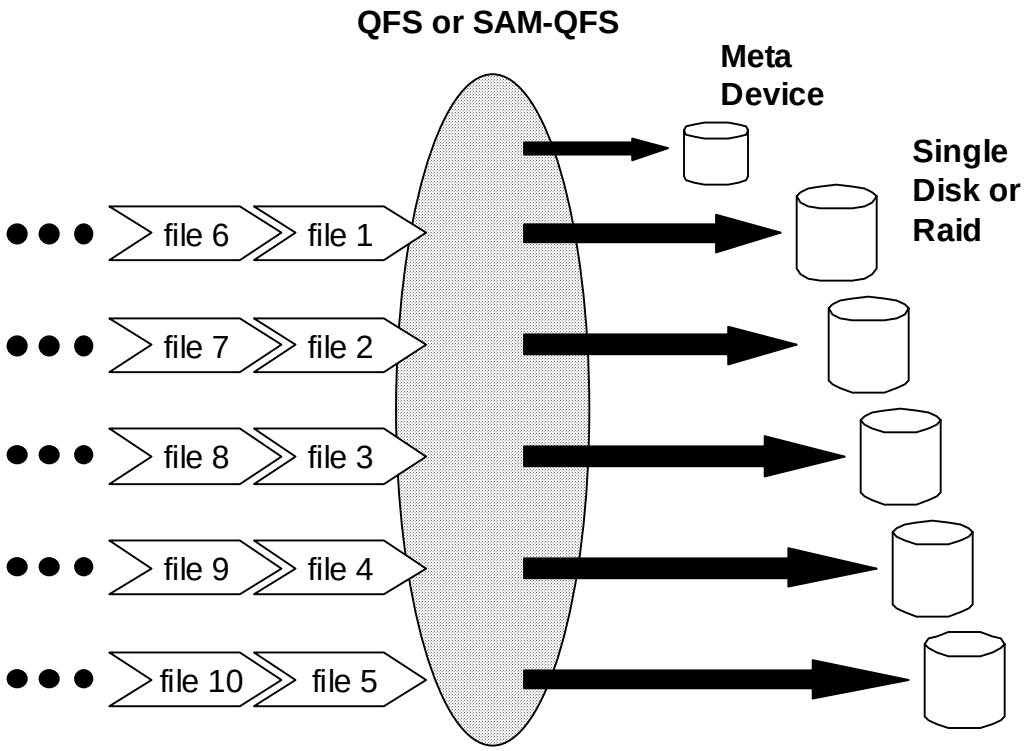

Figure 2-2. A round-robined SAM-QFS or QFS file system with five devices

Striped Allocation

By default, LSC file systems use a striped allocation method to spread file data over all the devices in the family set. *Striping* is a method of writing files in an interlaced fashion across multiple devices concurrently. A file system that is using striped devices addresses blocks in an interlaced fashion rather than sequentially. Striping generally increases performance because disk reads and writes are spread concurrently across disk heads.

In a striped file system, file 1 is written to disk 1, disk 2, disk 3, disk 4, and disk 5. File 2 is written to disks 1 through 5 as well. The DAU multiplied by the stripe width determines the amount of data written to each disk in a block.

When a SAM-FS file system starts to write a file, it first assumes that the file will fit into a small DAU, which is 4 kilobytes. If the file does not fit into the first 8 small DAUs (32 kilobytes) allocated, the file system writes the remainder of the file into one or more larger DAUs.

When a QFS or SAM-QFS file system starts to write a file, it writes first to one DAU, then another, and so on. The QFS and SAM-QFS file systems have only one DAU size.

The following figures depict LSC file systems using striped allocation. In these figures, DAU X *stripe_width* bytes of the file are written to disk 1, DAU X *stripe_width* bytes of the file are written to disk 2, DAU X *stripe_width* bytes of the file are written to disk 3, and so on. The order of the stripe is first-in-first-out for the files. Striping spreads the I/O load over all the disks.

Figure 2-3 depicts a SAM-FS file system using five striped devices. Figure 2-4 depicts a QFS or SAM-QFS file system using five striped devices.

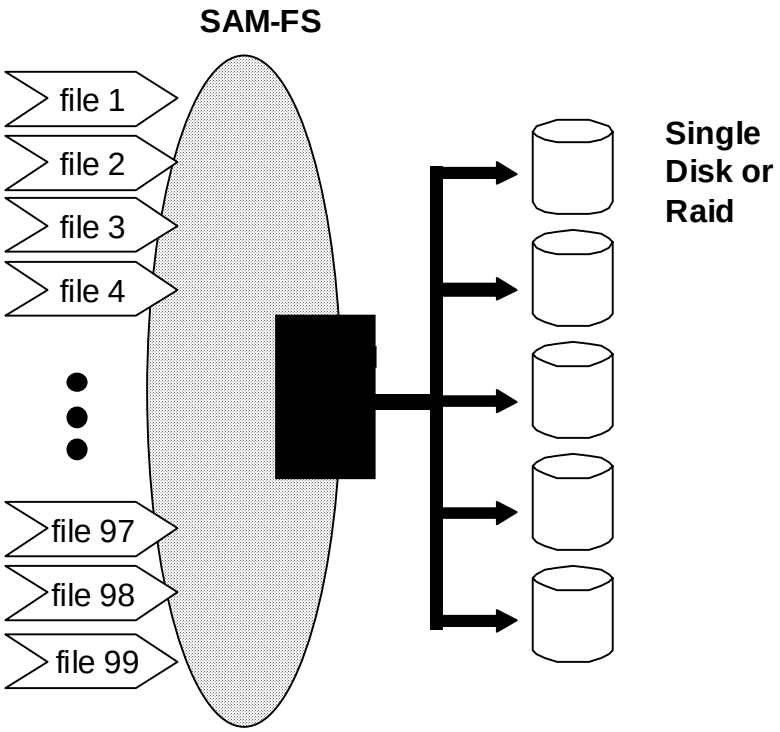

Figure 2-3. A SAM-FS file system using five striped devices

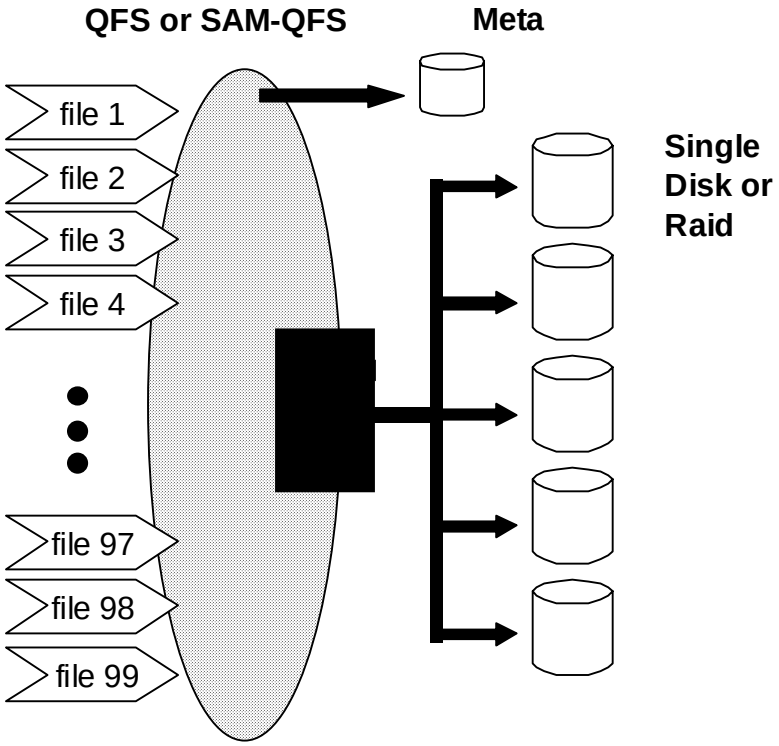

Figure 2-4. A QFS or SAM-QFS file system with five striped devices

Striped Groups (QFS and SAM-QFS File Systems Only)

A *striped group* is a special QFS and SAM-QFS allocation method that is designed for sites that have extremely large I/O requirements and terabytes of disk cache. A striped group allows you to designate an equipment type that contains multiple physical disks. Multiple striped group equipment types can make up a single QFS or SAM-QFS file system. Striped groups save bit map space and system update time for very large RAID configurations. Figure 2-5 shows striped groups.

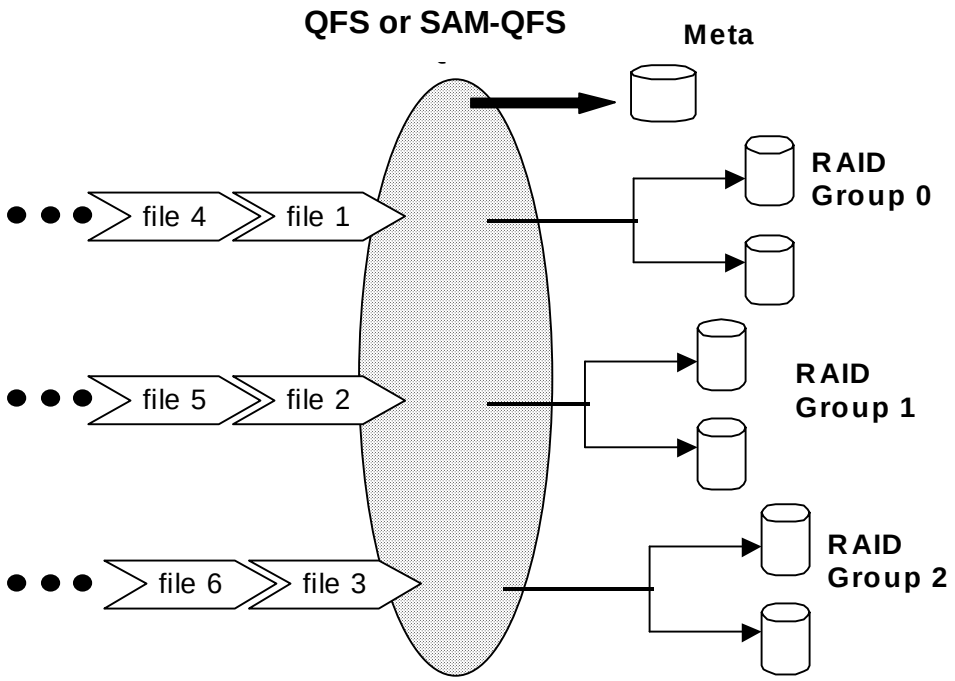

Figure 2-5. A QFS or SAM-QFS file system using striped groups and a round-robined allocation

In figure 2-5, files written to the `qfs1` file system are round-robined between groups `g0`, `g1`, and `g2`. Three striped groups are defined (`g0`, `g1`, and `g2`). Each group consists of two physical RAID devices. The mount point option in `/etc/vfstab` is set to `stripe=0`. These striped groups are declared as follows in the `mcf` file:

```
qfs1                10  ma  qfs1
/dev/dsk/c0t1d0s6   11  mm  qfs1  -   /dev/rdsk/c0t1d0s6
/dev/dsk/c1t1d0s2   12  g0  qfs1  -   /dev/rdsk/c1t1d0s2
/dev/dsk/c2t1d0s2   13  g0  qfs1  -   /dev/rdsk/c2t1d0s2
/dev/dsk/c3t1d0s2   14  g1  qfs1  -   /dev/rdsk/c3t1d0s2
/dev/dsk/c4t1d0s2   15  g1  qfs1  -   /dev/rdsk/c4t1d0s2
/dev/dsk/c5t1d0s2   16  g2  qfs1  -   /dev/rdsk/c5t1d0s2
/dev/dsk/c6t1d0s2   17  g2  qfs1  -   /dev/rdsk/c6t1d0s2
```

Figure 2-6 shows an example of striped groups in which the data is striped across groups.

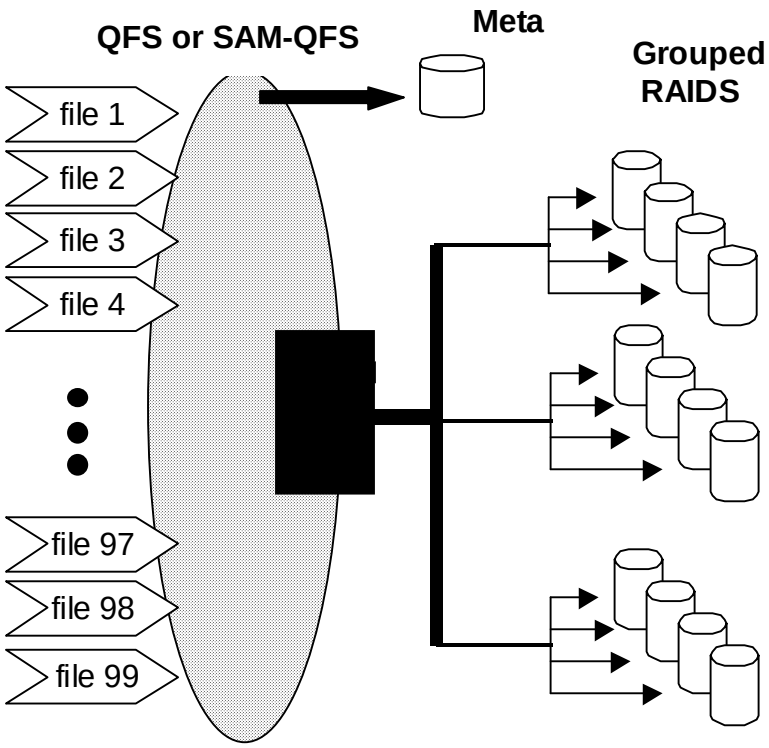

Figure 2-6. A QFS or SAM-QFS file system using striped groups (striped allocation)

In figure 2-6, files written to the `qfs1` file system are striped throughout groups `g0`, `g1`, and `g2`. Each group is comprised of four physical RAID devices. The mount point option in `/etc/vfstab` is set to `stripe=1` or greater.

Mismatched Striped Groups (QFS and SAM-QFS File Systems Only)

It is possible to build a file system with mismatched striped groups. *Mismatched striped groups* are those that do not contain the same number of devices in each group. LSC file systems support mismatched striped groups, but they do not support striping on mismatched groups. File systems with mismatched striped groups are round robin file systems.

The following example presents a situation and shows how a file system can be set up to store different types of files.

Assumptions

Assume that you have a QFS license, and you need to create a file system at your site that contains video and audio data.

Storing the Video and Audio Files

Video files are quite large and require greater performance than audio files. You want to store them in a file system with a large striped group because striped groups maximize performance for very large files.

Audio files are smaller and require less performance than video files. You want to store them in a small striped group. One file system can support both video and audio files.

The file system you want is depicted in figure 2-7.

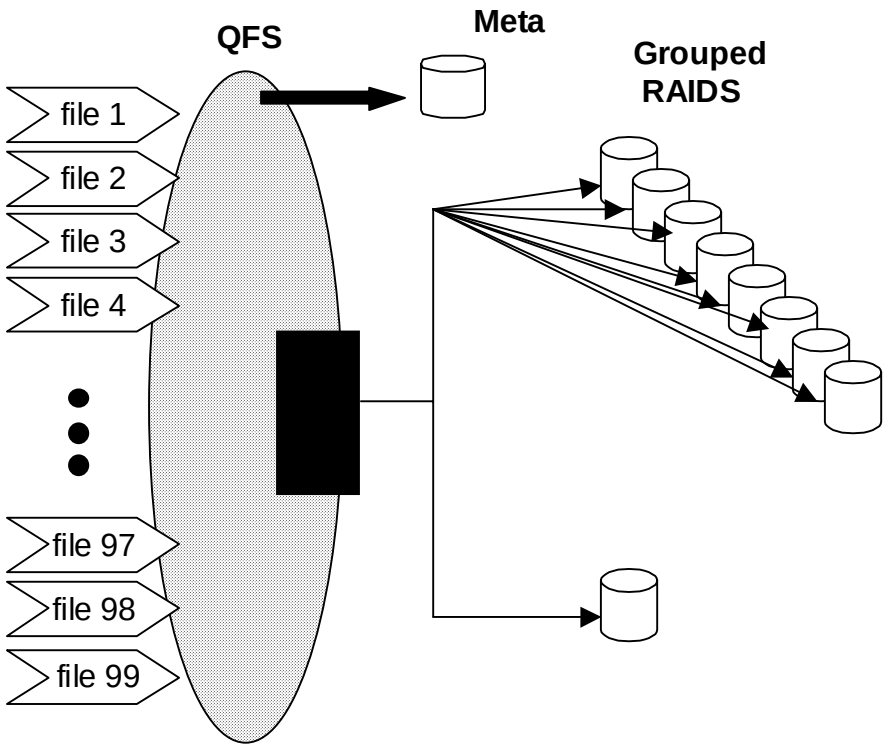

Figure 2-7. A QFS file system using mismatched striped groups (striped allocation)

The characteristics of this file system would be as follows:

Characteristic	Notes
File system name	`avfs`
Number of stripe groups	2. The video file group is `g0`. The audio file group is `g1`.
Stripe width	0
DAU	128 kilobytes
Number of disks for `g0`	8
Minimum block size for `g0`	8 disks X 128-kilobyte DAU = 1024 kilobytes (This is the amount of data written in one block write. Each disk receives 128 kilobytes of data, so the total amount written to all disks at one time is 1024 kilobytes.)
Number of disks for `g1`	1
Minimum block size for `g1`	1 disk X 128-kilobyte DAU = 128 kilobytes

The `/etc/vfstab` file needs to have the following line added to it in order for the environment to recognize the `avfs` file system:

```
avfs    -    /avfs    samfs    -    no    stripe=0
```

Note that in the `/etc/vfstab` file, `stripe=0` is used to specify a round-robin file system. This is used because a value greater than 0 (`stripe ≥ 0`) is not supported for mismatched striped groups.

The `mcf` file for this file system is as follows:

```
Equipment           Eq   Eq    Fam      Dev    Additional
Identifier          Ord  Type  Set      State  Parameters
#
avfs                100  ma    avfs
/dev/dsk/c00t1d0s6  101  mm    avfs  -  /dev/rdsk/c00t1d0s6
#
/dev/dsk/c01t1d0s6  102  g0    avfs  -  /dev/rdsk/c01t1d0s6
/dev/dsk/c02t2d0s6  103  g0    avfs  -  /dev/rdsk/c02t2d0s6
/dev/dsk/c03t3d0s6  104  g0    avfs  -  /dev/rdsk/c03t3d0s6
/dev/dsk/c04t4d0s6  105  g0    avfs  -  /dev/rdsk/c04t4d0s6
/dev/dsk/c05t5d0s6  106  g0    avfs  -  /dev/rdsk/c05t5d0s6
/dev/dsk/c06t6d0s6  107  g0    avfs  -  /dev/rdsk/c06t6d0s6
/dev/dsk/c07t1d0s6  108  g0    avfs  -  /dev/rdsk/c07t1d0s6
/dev/dsk/c08t1d0s6  109  g0    avfs  -  /dev/rdsk/c08t1d0s6
#
/dev/dsk/c09t1d0s6  110  g1    avfs  -  /dev/rdsk/c09t1d0s6
```

After the `mcf` file for this file system is ready, you can enter the following `sammkfs`(1M) and `mount`(1M) commands to create and mount the `avfs` file system:

```
server# sammkfs –a 128 avfs
```

After the file system is mounted, you can create 2 directories for the two types of files by issuing the following commands:

```
server# mkdir video
server# mkdir audio
```

The previous two commands create two directories in the file system.

```
server# setfa –g0 video
server# setfa –g1 audio
```

The first of the preceding two commands assigns the large striped group to video. The second of the preceding two commands assigns the small striped group to audio. Files created in these directories are allocated on their respective striped groups because attributes are inherited.

For more information on the `sammkfs`(1M) command, see the `sammkfs`(1M) man page. Fore more information on the `mount`(1M) commands, see the `mount_samfs`(1M) man page. For more information on the `setfa`(1) command, see the `setfa`(1) man page.

Volume Management
Chapter 3

The volume manager groups physical devices into family sets upon which the file system is created. The master configuration file (`mcf`) controls the volume manager's features.

The configuration process is part of the installation process. The steps in the configuration process are as follows:

1. Create the `/etc/opt/LSCsamfs/mcf` file.
2. Edit the `/etc/vfstab` file.
3. Use the `sammkfs`(1M) command to construct the new file system.
4. Use the `mount`(1M) command to mount the file system.

The installation and configuration process is described completely in the *QFS, SAM-FS, and SAM-QFS Software Installation and Configuration Guide*, publication SG-0007. This chapter provides more information on configuring the file systems used in the QFS, SAM-FS, and SAM-QFS environments.

This chapter describes the following topics:

- Creating the `/etc/opt/LSCsamfs/mcf` file
- `mcf` file examples
- Interactions between file settings, options, and directives
- Configuration examples

Creating the `mcf` File

The first step toward configuring an LSC file system is to create a master configuration file in `/etc/opt/LSCsamfs/mcf`. The `mcf` file contains the information that the LSC file system needs in order to identify and organize RAID and disk devices into file systems. It also contains entries for each automated library or device included in a file system. A sample `mcf` file is located in `/opt/LSCsamfs/examples/mcf`.

An `mcf` file is an ASCII file that consists of lines of specification code divided into six *columns*, or *fields*. The following format shows the six fields that comprise each line in an `mcf` file:

```
Equipment   Equipment   Equipment   Family  Device   Additional
Identifier  Ordinal     Type        Set     State    Parameters
```

The following rules pertain to how data can be entered in the `mcf` file:

- Enter either space or tab characters between the fields in the file.
- You can include comment lines in an `mcf` file. Comment lines start with a pound sign (#).
- As the following table indicates, some fields are optional. If you do not enter meaningful data in a field, enter a dash character (`-`) to indicate that this is an omitted field.

The following table describes the fields.

Field	Description
`Equipment Identifier`	Required. This field is either the name of a file system, a `/dev/dsk` entry, a `/dev/samst` entry, or a `/dev/rmt` entry. If this field contains the name of a file system, the subsequent lines in the `mcf` file all define the disks or devices included in the file system. More than one file system can be declared in an `mcf` file. Typically, the first data line in an `mcf` file declares the first file system, and subsequent lines specify the devices included in the file system. The other file systems declared in the `mcf` file can be preceded by a blank comment line for readability. File system names must start with an alphabetic character and can contain only contain alphabetic characters, numeric characters, or underscore (_) characters. If this field is a `/dev/dsk` entry, it identifies a disk partition or slice. If this field is a `/dev/samst` entry, it identifies an automated library or optical drive. If you are configuring a network-attached automated library, see the *QFS, SAM-FS, and SAM-QFS Installation and Configuration Guide*, publication SG-0007, and the *SAM-FS and SAM-QFS Storage and Archive Management Guide*, publication SG-0008, for more information. If this field is a `/dev/rmt` entry, it identifies a tape drive.
`Equipment Ordinal`	Required. This field assigns a numeric identifier to the piece of equipment declared in the preceding `Equipment Identifier` field. There must be an equipment ordinal for each element in the file system. Enter a unique integer from `1` to `65535`.

Field	Description
`Equipment Type`	Required. Enter a 2-, 3-, or 4-character code for the device type. The SAM-FS file system uses the following equipment types: `ms` Defines a SAM-FS file system. `md` Defines a striped or round-robined device for storing file data and metadata information. The QFS and SAM-QFS file systems use the following equipment types: `ma` Defines a QFS or SAM-QFS file system. `mm` Defines a metadata device for storing inode and other non-data information. `mr` Defines a round-robined or striped data device. `g`*XXX* Striped group data device. Striped groups start with the letter `g` followed by a number. The number must be an integer such that $0 \leq XXX \leq 127$. For example, `g12`. All members in a striped group must be the same type and size. Different striped groups within one file system are not required to have the same number of members. `mr` and `g`*XXX* devices cannot be mixed in one file system. Besides the file system equipment types, other codes are used to identify automated libraries and other devices. For more information on specific equipment types, see the `mcf`(4) man page.

Field	Description
`Family Set`	Required for file system devices. Optional for other devices. If this is an optional field, enter a dash (`-`) character to indicate that this field is omitted. Naming conventions for family set names are identical to those for file system names. The names must start with an alphabetic character and can contain only contain alphabetic characters, numeric characters, or underscore (_) characters. For a file system, this field is required because a family set associates all devices with the same family set name together as a file system. The family set name is physically recorded on all the devices in the file system when the `sammkfs`(1M) command is issued. It is possible to change this name by using the `–F` and `–R` options together on the `samfsck`(1M) command. For more information on the `sammkfs`(1M) command, see the `sammkfs`(1M) man page. For more information on the `samfsck`(1M) command, see the `samfsck`(1M) man page. In SAM-FS and SAM-QFS environments, this field can be either a family set name or a dash (`-`). If the device is associated with a family set (that is, a file system or an automated library), enter the family set name for this device. If the device is a manually loaded drive, enter a dash to indicate that this field is omitted.
`Device State`	Optional. Enter a state for the device when the file system is initialized. Valid device states are `on` and `off`. If `on` or `off` are not entered, enter a dash (`-`) character to indicate that this field is omitted.
`Additional Parameters`	Required for file system components. Optional for other devices. If this is an optional field, enter a dash (`-`) character to indicate that this field is omitted. If this line identifies a disk slice (has a `/dev/dsk` equipment identifier), this field must contain a corresponding `/dev/rdsk` identifier. If this line identifies an automated library, this field must be the path name to the library catalog file. The default library catalog file is `/var/opt/LSCsamfs/catalog/`*family_set_name*.

For more information on writing the `mcf` file, see the `mcf`(4) man page.

mcf File Examples

Each file system configuration is unique. System requirements and actual hardware differ from site to site. The following subsections show sample mcf files for QFS, SAM-FS, and SAM-QFS environments.

SAM-FS Volume Management Examples

For the SAM-FS file system, family sets are defined in the /etc/opt/LSCsamfs/mcf file in the Equipment Type field using the following equipment types:

- ms for the SAM-FS file system type.
- md for the devices. Data is striped or round robined across these devices. The stripe width is set with the -o stripe=*n* option on the mount(1M) command. The default stripe width is set based on the DAU size. For more information on stripe widths and DAU sizes, see chapter 2, "File System Design".

Metadata (including inodes, directories, allocation maps, and so on) and file data on SAM-FS file systems are located on the same disk. Data files are striped or round-robined through each disk partition defined within the same file system.

The following example shows an mcf file for a SAM-FS file system:

```
# SAM-FS file system configuration example
#
# Equipment       Eq    Eq   Fam.  Dev.    Additional
# Identifier      Ord  Type  Set   State   Parameters
#-----------      ---   --  ------ ------  ------------------
samfs1             10  ms  samfs1
/dev/dsk/c1t1d0s6  11  md  samfs1  -       /dev/rdsk/c1t1d0s6
/dev/dsk/c2t1d0s6  12  md  samfs1  -       /dev/rdsk/c2t1d0s6
/dev/dsk/c3t1d0s6  13  md  samfs1  -       /dev/rdsk/c3t1d0s6
/dev/dsk/c4t1d0s6  14  md  samfs1  -       /dev/rdsk/c4t1d0s6
/dev/dsk/c5t1d0s6  15  md  samfs1  -       /dev/rdsk/c5t1d0s6
```

QFS and SAM-QFS Volume Management Examples

For the QFS and SAM-QFS file systems, family sets are defined in the /etc/opt/LSCsamfs/mcf file in the Equipment Type field using the following device types:

- ma for the QFS or SAM-QFS file system type.
- mm for metadata. File data is not written to this device. You can specify multiple metadata devices.
- mr for devices upon which file data is to be striped or round-robined. The stripe width is defined as a mount option. The default stripe width is set based on the DAU size. For more information on stripe widths and DAU sizes, see chapter 2, "File System Design".

- g*XXX*. For devices upon which file data is to be striped as a group. A striped group is a logical group of devices that are striped as a unit. Data is striped across the members of each group.

  Groups are specified with `g0` through `g127` equipment type numbers with the stripe width on each device being the DAU. All devices in a striped group must be the same size. Different striped groups within one file system are not required to have the same number of members. `mr` and g*XXX* devices cannot be mixed in a file system.

  Data can be striped (if all groups contain the same number of devices) or round-robined between groups. The default is round robin.

Metadata (including inodes, directories, allocation maps, and so on) on QFS and SAM-QFS file systems is located on the metadata device(s) and is separated from the file data devices. By default, metadata is allocated using round-robin allocation if you have multiple metadata devices.

Data files are striped or round-robined through each data disk partition (`mr` or g*XXX*) defined within the same file system.

Example 1. The following example shows an `mcf` file with two QFS or SAM-QFS file systems:

```
# SAM-QFS file system configuration example
#
# Equipment       Eq    Eq   Fam.  Dev.    Additional
# Identifier      Ord  Type  Set   State   Parameters
#-----------      ---   --  ------ ------  ------------------
qfs1               10  ma   qfs1   -
/dev/dsk/c2t1d0s7  11  mm   qfs1   -   /dev/rdsk/c2t1d0s7
/dev/dsk/c3t0d0s6  12  mr   qfs1   -   /dev/rdsk/c3t0d0s6
/dev/dsk/c3t0d1s6  13  mr   qfs1   -   /dev/rdsk/c3t0d1s6
```

Example 2. The following example shows an `mcf` file for a QFS or SAM-QFS file system with two striped groups:

```
# QFS file system configuration
#
# Equipment       Eq   Eq    Fam.  Dev.    Additional
# Identifier      Ord  Type  Set   State   Parameters
#-----------      ---   --  ------ ------  ------------------
qfs1               10   ma   qfs1    -
/dev/dsk/c2t1d0s7  11   mm   qfs1    -     /dev/rdsk/c2t1d0s6
/dev/dsk/c3t0d0s6  12   g0   qfs1    -     /dev/rdsk/c3t0d0s6
/dev/dsk/c3t0d1s6  13   g0   qfs1    -     /dev/rdsk/c3t0d1s6
/dev/dsk/c4t0d0s6  14   g1   qfs1    -     /dev/rdsk/c4t0d0s6
/dev/dsk/c4t0d1s6  15   g1   qfs1    -     /dev/rdsk/c4t0d1s6
```

Example 3. The following example shows an `mcf` file with three QFS or SAM-QFS file systems:

```
# SAM-QFS file system configuration example
#

# Equipment       Eq    Eq   Fam.  Dev.    Additional
# Identifier      Ord  Type  Set   State   Parameters
#-----------      ---   --  ------ ------  ------------------
qfs1               10  ma   qfs1   -
/dev/dsk/c1t13d0s6 11  mm   qfs1   -
/dev/rdsk/c1t13d0s6
/dev/dsk/c1t12d0s6 12  mr   qfs1   -
/dev/rdsk/c1t12d0s6
#
qfs2               20  ma   qfs2   -
/dev/dsk/c1t5d0s6  21  mm   qfs2   -       /dev/rdsk/c1t5d0s6
/dev/dsk/c5t1d0s6  22  mr   qfs2   -       /dev/rdsk/c5t1d0s6
#
qfs3               30  ma   qfs3   -
/dev/dsk/c7t1d0s3  31  mm   qfs3   -       /dev/rdsk/c7t1d0s3
/dev/dsk/c6t1d0s6  32  mr   qfs3   -       /dev/rdsk/c6t1d0s6
/dev/dsk/c6t1d0s3  33  mr   qfs3   -       /dev/rdsk/c6t1d0s3
/dev/dsk/c5t1d0s3  34  mr   qfs3   -       /dev/rdsk/c5t1d0s3
```

For more examples showing file system configuration in the `mcf` file, see the *QFS, SAM-FS, and SAM-QFS Software Installation and Configuration Guide*, publication SG-0007.

Interactions Between File Settings, Options, and Directives

The `mcf` file defines each LSC file system, but file system behavior depends on interactions between default systems settings, settings in the `/etc/vfstab` file, settings in the `samfs.cmd` file, and options on the `mount`(1M) command line.

You can specify certain system settings, for example the stripe width, in more than one place. When this happens, settings in one place can override the settings in another. Various files and command lines interact as follows:

Highest priority (overrides all others)	`mount`(1M) options
|/	`/etc/vfstab` settings
|	`samfs.cmd` file settings
Lowest Priority	System defaults

The following subsections describe these system components in more detail, explain when to use these files and commands, and show the order in which they take precedence.

System Defaults

The default system settings are the configurable settings already defined for your Solaris system. These system settings can be overridden by settings in the `samfs.cmd` file, the `/etc/vfstab` file, and on the `mount`(1M) command.

The `samfs.cmd` File

The `samfs.cmd` file is an LSC command file that allows you to specify mount parameters for all your LSC file systems. This file can be useful when you have multiple file systems configured and you want to specify differing mount parameters for any of them.

The list of possible mount parameters is very comprehensive. The possible mount parameters you can specify pertain to I/O settings, readahead, writebehind, the stripe width, various storage and archive management (SAM) settings, and Qwrite. The mount parameters that can be specified in the `samfs.cmd` file are identical to those that you can specify in the `/etc/vfstab` file or as arguments to the `-o` option on the `mount`(1M) command. For more information on possible mount parameters, see the `samfs.cmd`(4) man page.

Using this file allows you to define all your mount parameters in one place in an easily readable format. Directives specified toward the beginning of this file apply to all LSC file systems. The second part of this file allows you to indicate the specific parameters that you want to apply to each file system. The ability to specify the common parameters once, and only in one place, differentiates this file from the `/etc/vfstab` file, in which you must specify all mount parameters for each file system in the seventh field.

In the `samfs.cmd` file, the directives are written one per line. The file can contain comments, which must begin with a pound character (#). Characters that appear to the right of the pound character are treated as comments.

Directives that appear before any `fs =` line apply globally to all file systems. A line that starts with `fs =` must precede directives that are specific to a particular file system. Directives specific to a particular file system override global directives.

NOTE

Make sure that there is a space on each side of every equal sign (" = ") in the `samfs.cmd` file. Failure to include the spaces causes a syntax error, prevents the file from running, and creates an error message in the log file for SAM-FS and SAM-QFS file systems.

The following example `samfs.cmd` file sets the low and high water marks for disk cache utilization and specifies individualized parameters for two file systems:

```
inodes = 32768
low = 50
high = 75
   fs = samfs1
   high = 65
   writebehind = 512
   readahead = 1024
   fs = samfs5
   partial = 64
```

The directives in the `samfs.cmd` file serve as defaults and override any default system settings, but arguments to the `mount`(1M) command override any directives in this file. Entries in the `/etc/vfstab` file also override directives specified in the `samfs.cmd` file.

For information on which directives can be entered in the `samfs.cmd` file, see the `samfs.cmd`(4) man page. For information on the `mount`(1M) command, see the `mount_samfs`(1M) man page.

The `/etc/vfstab` File

The `/etc/vfstab` Solaris system file must contain a line for each LSC file system that is defined in the `mcf` file. For each file system, you must fill in seven fields that contain the following information:

Field Number	Content
1	The file system name.
2	The file system to `fsck`(1M).
3	The mount point.
4	The file system type (always `samfs`, even for QFS and SAM-QFS file systems).
5	The `fsck`(1M) pass.
6	Mount at boot options.
7	Mount parameters.

The fields in the `/etc/vfstab` file must be separated by either space or tab characters. The mount parameters in the seventh field, however, must each be separated by a comma character (`,`) without any intervening spaces.

The mount parameters field can contain any of the mount parameters listed as arguments to the `-o` option on the `mount_samfs`(1M) man page. These parameters are identical to those that you can specify as directive lines in the `samfs.cmd` file or as arguments to the `-o` option on the `mount`(1M) command. As with the `samfs.cmd` file, you can specify various I/O settings, readahead, writebehind, the stripe width, various storage and archive management (SAM) settings, and qwrite.

For more information on possible mount parameters, see the `mount_samfs`(1M) man page. For more information on modifying the `/etc/vfstab` file, see the *QFS, SAM-FS, and SAM-QFS Software Installation and Configuration Guide*, publication SG-0007, or see the `vfstab`(4) man page.

The `mount`(1M) Command

The Solaris `mount`(1M) command mounts the file system and allows you to specify settings that override the settings specified in the `/etc/vfstab` file and in the `/etc/opt/LSCsamfs/samfs.cmd` file. For example, you can specify the stripe width, readahead, writebehind, high and low water marks for disk cache utilization, and so on. For more information, see the `mount_samfs`(1M) man page.

One way to use the mount(1M) command in conjunction with the samfs.cmd file is to use the samfs.cmd file as your main location for mount options and to use options on the mount(1M) command when experimenting with or tuning your system. The mount(1M) command options override both the /etc/vfstab entries and the directives in the samfs.cmd file.

Example. The following command mounts file system qfs1 at /work with setuid execution disallowed. The qfs1 file system name is the equipment identifier. This also appears in the mcf file's Equipment Identifier field for this file system.

```
server# mount –o nosetuid qfs1 /work
```

For more information on the mount(1M) command, see the mount_samfs(1M) man page.

The sammkfs(1M) Command

The sammkfs(1M) command constructs new file systems. It allows you to specify the DAU setting. For SAM-FS and SAM-QFS, this command is also used when restoring files or file systems. Another command, samfsinfo(1M) can be used to gather the configuration information for an existing file system.

The sammkfs(1M) command must be issued prior to issuing the mount(1M) command when installing and configuring and LSC file system. For more information, see the sammkfs(1M) man page.

Configuration Examples

The rest of this chapter presents sample QFS configurations and shows various steps and considerations in setting up the mcf file on a server. The following procedures are described:

- How to configure a metadata disk and round-robined data disks
- How to configure a metadata disk and striped disks
- How to configure striped groups

Note that all sample QFS configurations could have automated libraries and other removable media devices defined as well, essentially extending the size of the disk cache. Removable media device configurations are not shown. For information on configuring removable media devices see the *QFS, SAM-FS, and SAM-QFS Installation and Configuration Guide*, publication SG-0007, and the *SAM-FS and SAM-QFS Storage and Archive Management Guide*, publication SG-0008.

The sample configurations assume that the LSC file system is loaded on the system and all file systems are unmounted.

QFS Round-robined Disk

This sample configuration illustrates a QFS file system that separates the metadata on to a low-latency disk. Round-robin allocation is used on four partitions. The file system is created using the `sammkfs`(1M) command. Each disk is on a separate LUN.

The following assumptions are used:

- The metadata device is a single partition (`s6`) used on controller 5, LUN 0.
- The data devices consist of four disks attached to four controllers.

Step 1: Write the mcf File

The following is a sample `mcf` file for a round-robin disk configuration:

```
# QFS disk cache configuration – Round-robin mcf sample
#
# Equipment       Eq   Eq    Fam.  Dev     Additional
# Identifier      Ord  Type  Set   State   Parameters
#-----------      ---   --  ------ ------  ------------------
qfs1                1   ma   qfs1    -
/dev/dsk/c5t0d0s6  11   mm   qfs1    on    /dev/rdsk/c5t0d0s6
/dev/dsk/c1t1d0s0  12   mr   qfs1    on    /dev/rdsk/c1t1d0s0
/dev/dsk/c2t1d0s0  13   mr   qfs1    on    /dev/rdsk/c2t1d0s0
/dev/dsk/c3t1d0s0  14   mr   qfs1    on    /dev/rdsk/c3t1d0s0
/dev/dsk/c4t1d0s0  15   mr   qfs1    on    /dev/rdsk/c4t1d0s0
```

Step 2: Modify the `/etc/vfstab` File

The `/etc/vfstab` is edited. Because the QFS file system with `mr` data devices uses striped allocation as a default, you must set `stripe=0` for round-robin allocation.

To explicitly set round-robin on the file system, set the `stripe=0` as follows:

```
qfs1    -    /qfs    samfs    -    yes    stripe=0
```

Step 3: Run the `sammkfs`(1M) Command

Initialize the QFS file system by using the `sammkfs`(1M). The default DAU is 16 kilobytes, but the following example sets the DAU size to 64 kilobytes:

```
server# sammkfs -a 64 qfs1
```

SAM-FS Round-robined Disk

This sample configuration illustrates a SAM-FS file system. Striped allocation is used by default on four partitions. You need to set `stripe=0` to specify round robin allocation. The file system is created using the `sammkfs`(1M) command. The data devices consist of four disks attached to four controllers. Each disk is on a separate LUN.

Step 1: Write the `mcf` *File*

The following is a sample `mcf` file for a round-robin disk configuration:

```
# SAM-FS disk cache configuration – Round-robin mcf sample
#
# Equipment       Eq   Eq    Fam.   Dev     Additional
# Identifier      Ord  Type  Set    State   Parameters
#-----------      ---   --  ------  ------  -----------------
samfs1              1   ms   samfs1
/dev/dsk/c1t1d0s0  11   md   samfs1  on    /dev/rdsk/c1t1d0s0
/dev/dsk/c2t1d0s5  12   md   samfs1  on    /dev/rdsk/c2t1d0s0
/dev/dsk/c3t1d0s0  13   md   samfs1  on    /dev/rdsk/c3t1d0s0
/dev/dsk/c4t1d0s1  14   md   samfs1  on    /dev/rdsk/c4t1d0s0
```

Step 2: Modify the `/etc/vfstab` *File*

The `/etc/vfstab` is edited. Because the SAM-FS file system uses striped allocation by default, you must set the `stripe=0` to obtain round-robined allocation.

To explicitly set round-robin on the file system, set the `stripe=0` as follows:

```
samfs1    -    /samfs1    samfs    -    yes   stripe=0
```

Step 3: Run the `sammkfs` *(1M) Command*

Initialize the SAM-FS file system by using the `sammkfs`(1M). The default DAU is 16 kilobytes, but the following example sets the DAU size to 64 kilobytes:

```
server# sammkfs -a 64 samfs1
```

QFS Striped Disk Configuration

This sample configuration illustrates a QFS file system that again separates the metadata onto a low-latency disk. By default, file data is striped to four partitions. The file system is created using the `sammkfs`(1M) command, and the DAU size is specified.

The following assumptions are used:

- The metadata device is a single partition (`s6`) used on controller 0, LUN 1.
- The data devices consist of four disks attached to four controllers. Each disk is on a separate LUN.

Step 1. Write the `mcf` *File*

Write the `mcf` file using the disk configuration assumptions. The following is a sample `mcf` file for a striped disk configuration:

```
# QFS disk cache configuration – Striped Disk mcf sample
#
# Equipment       Eq   Eq    Fam.  Dev.    Additional
# Identifier      Ord  Type  Set   State   Parameters
#-----------      ---   --  ------ ------  ------------------
qfs1               10   ma  qfs1
/dev/dsk/c0t1d0s6  11   mm  qfs1     on    /dev/rdsk/c0t1d0s6
/dev/dsk/c1t1d0s0  12   mr  qfs1     on    /dev/rdsk/c1t1d0s0
/dev/dsk/c2t1d0s0  13   mr  qfs1     on    /dev/rdsk/c2t1d0s0
/dev/dsk/c3t1d0s0  14   mr  qfs1     on    /dev/rdsk/c3t1d0s0
/dev/dsk/c4t1d0s0  15   mr  qfs1     on    /dev/rdsk/c4t1d0s0
```

Step 2: Modify the `/etc/vfstab` *File*

Set the stripe width using the `stripe=` option. The following example sets the stripe width equal to one DAU, which is the default:

```
qfs1    -    /qfs    samfs    -    yes   stripe=1
```

This setting stripes file data across all four of the `mr` data drives with a stripe width of one DAU. Note the DAU is the allocation unit you set when you initialize the file system (see "Step 3", following).

Step 3: Run the `sammkfs` *(1M) Command*

Initialize the QFS file system using the `sammkfs`(1M) command. The following example sets the DAU size to 128 kilobytes:

```
server# sammkfs -a 128 qfs1
```

With this striped disk configuration, any file written to this file system is striped across all of the devices in increments of 128 kilobytes.

Metadata is written to device 11 only.

SAM-FS Striped Disk Configuration

This sample configuration illustrates a SAM-FS file system that again separates the metadata onto a low-latency disk. File data is striped to four disk drives. The file system is created using the `sammkfs`(1M) command, and the DAU size is specified. The data devices consist of four disks attached to four controllers. Each disk is on a separate LUN.

Step 1. Write the `mcf` *File*

Write the `mcf` file using the disk configuration assumptions. The following is a sample `mcf` file for a striped disk configuration:

```
# SAM-FS disk cache configuration – Striped Disk mcf sample
#
# Equipment       Eq   Eq    Fam.  Dev.    Additional
# Identifier      Ord  Type  Set   State   Parameters
#-----------      ---   --  ------ ------  ------------------
samfs1             10   ms  samfs1
/dev/dsk/c1t1d0s0  11   md  samfs1   on    /dev/rdsk/c1t1d0s0
/dev/dsk/c2t1d0s0  12   md  samfs1   on    /dev/rdsk/c2t1d0s0
/dev/dsk/c3t1d0s0  13   md  samfs1   on    /dev/rdsk/c3t1d0s0
/dev/dsk/c4t1d0s0  14   md  samfs1   on    /dev/rdsk/c4t1d0s0
```

Step 2: Modify the `/etc/vfstab` *File*

It is not necessary to modify the `/etc/vfstab` file because this example uses defaults.

Step 3: Run the `sammkfs` *(1M) Command*

Initialize the SAM-FS file system using the `sammkfs`(1M) command. The following example sets the DAU size to 128 kilobytes:

```
server# sammkfs -a 128 samfs1
```

With this striped disk configuration, any file written to this file system is striped across all of the devices in increments of 128 kilobytes.

QFS Striped Groups Configuration

Striped groups allow you to group RAID devices together for very large files. Normally, a DAU is represented by one bit in the bit maps. If the striped group has *n* devices, *n* multiplied by the DAU is the minimum allocation. Only one bit in the bit maps is used to represent *n* X DAU. This method of writing huge DAUs across RAID devices saves bit map space and system update time. Striped groups are useful for writing very large files to a group of RAID devices.

NOTE

The minimum disk space allocated in a striped group is as follows:

minimum_disk_space_allocated = DAU X *number_of_disks_in_the_group*

Writing a single byte of data fills the entire *minimum_disk_space_allocated* of a striped group. The use of striped groups is for very specific applications. Make sure that you understand the effects of using striped groups with your file system.

Files less than the aggregate stripe width times the number of devices (in this example, files less than 128 kilobytes X 4 disks = 512 kilobytes in length) still use 512 kilobytes of disk space. Files larger than 512 kilobytes have space allocated for them as needed in total space increments of 512 kilobytes.

The devices within a group must be the same size. It is not possible to add devices to increase the size of a striped group. You can use the `samgrowfs`(1M) command to add additional striped groups, however. For more information on this command, see the `samgrowfs`(1M) man page.

This sample configuration illustrates a QFS file system that separates the metadata onto a low-latency disk. Two striped groups are set up on four drives.

The following assumptions are used:

- The metadata device is a single partition (`s6`) used on controller 0, LUN 1.
- The data devices consist of four disks (two groups of two identical disks) attached to four controllers. Each disk is on a separate LUN. The entire disk is used for data storage assuming that partition 6 is the entire disk.

Step 1. Write the `mcf` *File*

Write the `mcf` file using the disk configuration assumptions. The following is a sample `mcf` file for a striped groups configuration:

```
# QFS disk cache configuration – Striped Groups mcf sample
#
# Equipment       Eq   Eq    Fam.  Dev.    Additional
# Identifier      Ord  Type  Set   State   Parameters
#-----------      ---   --  ------ ------  ------------------
qfs1               10   ma  qfs1
/dev/dsk/c0t1d0s6  11   mm  qfs1     on    /dev/rdsk/c0t1d0s6
/dev/dsk/c1t1d0s6  12   g0  qfs1     on    /dev/rdsk/c1t1d0s6
/dev/dsk/c2t1d0s6  13   g0  qfs1     on    /dev/rdsk/c2t1d0s6
/dev/dsk/c3t1d0s6  14   g1  qfs1     on    /dev/rdsk/c3t1d0s6
/dev/dsk/c4t1d0s6  15   g1  qfs1     on    /dev/rdsk/c4t1d0s6
```

Step 2: Modify the `/etc/vfstab` *File*

Set the stripe width using the `stripe=` option. This example uses the default stripe setting of `stripe=0`, which essentially specifies a round-robined allocation from striped group `g0` to striped group `g1`:

```
qfs1    -    /qfs    samfs    -    yes   stripe=0
```

Step 3: Run the `sammkfs` *(1M) Command*

Initialize the QFS file system by using the following `sammkfs`(1M) command:

```
server# sammkfs –a 128 qfs1
```

In this example, there are two striped groups, `g0` and `g1`. With `stripe=0` in `/etc/vfstab`, devices 12 and 13 are striped; devices 14 and 15 are striped; and files are round robined around the 2 striped groups. You are really treating a striped group as a bound entity; that is, the configuration of a striped group, after it is created, cannot be changed. You cannot change these groups without issuing another `sammkfs`(1M) command.

File System Operations
Chapter 4

This chapter presents topics related to file system operations. It presents the following topics:

- How to make a file system
- How to mount a file system
- How to unmount a file system
- How to check file system integrity
- How to repair a file system
- How to dump and restore QFS file systems
- How to prepare for a hardware upgrade
- How to add disk cache to a file system
- How to replace disks in a file system
- How to upgrade a server
- How to upgrade your Solaris operating system (SAM-FS and SAM-QFS environments)
- How to upgrade your Solaris operating system (QFS environment)

Certain other types of operations and upgrades also need to be performed within a QFS, SAM-FS, or SAM-QFS environment. The following LSC publications describe these other types of upgrades:

- The *QFS, SAM-FS, and SAM-QFS Installation and Configuration Guide*, publication SG-0007, describes upgrading QFS, SAM-FS, and SAM-QFS software.
- The *SAM-FS and SAM-QFS Storage and Archive Management Guide*, publication SG-0008, describes the following types of operations and upgrades in chapter 10, "How to Upgrade Your Environment":
  - How to add slots in an automated library
  - How to upgrade or replace an automated library
  - How to upgrade DLT tape drives

How to Make a File System

A QFS, SAM-FS, or SAM-QFS file system can be created by using the `sammkfs`(1M) command. The following example shows this command in its simplest form, with the file system name as its only argument:

```
server# sammkfs samqfs1
```

For more information on the `sammkfs`(1M) command, see chapter 3, "Volume Management", or see the `sammkfs`(1M) man page.

How to Mount a File System

Mount parameters are used to manipulate file system characteristics. These parameters can be specified in the following ways:

- As options to the `mount`(1M) command.
- Through the operator utility, `samu`(1M). This is temporary setting. (SAM-FS and SAM-QFS only)
- As directives in the `Additional parameters` field in the `/etc/opt/LSCsamfs/samfs.cmd` command file. Note that options to the `mount`(1M) and parameters in the `/etc/vfstab` file override directives in the `samfs.cmd` file.

Example. Assume that the `sam` file system is defined in the `/etc/vfstab` file. The following `mount`(1M) command mounts a SAM-FS file system at `/sam`:

```
server# mount /sam
```

For more information, see the following man pages: `mount_samfs`(1M) and `samfs.cmd`(4).

How to Unmount a File System

Enter the following command to unmount a file system:

```
server# umount /mountpoint
```

For example, if your file system is mounted at `/sam`, you would enter the following command:

```
server# umount /sam
```

If you experience difficulties unmounting because the file system is busy, perform the following procedure:

1. Obtain information on busy processes under a specific *mountpoint* by entering the following command:

```
server# fuser -c /mountpoint
```

This command does not return information on processes on other systems that are accessing the server through NFS.

2. Notify users that all processes are about to be terminated, and then kill all processes by entering the following command:

```
server# fuser -ck /mountpoint
```

3. Unshare and unmount the file system. The following commands force NFS to relinquish the file system:

```
server# unshare /mountpoint
server# umount /mountpoint
```

If all attempts to unmount the file system still fail, edit the `/etc/vfstab` file and change the sixth field (the mount at boot field) for all file systems from `yes` or `delay` to `no`. Then reboot your system.

For more information, see the `fuser`(1M), `unshare`(1M), and `umount`(2) man pages.

How to Check File System Integrity

LSC file systems write validation records in all records critical to file system operations: directories, indirect blocks, and inodes. If corruption is detected while searching a directory, an `EDOM` error is returned, and the directory is not processed. If an indirect block is not valid, an `ENOCSI` error is returned, and the file is not processed. The following list summarizes these error indicators:

Error	Solaris meaning	SAM-FS meaning
`EDOM`	Argument is out of domain	Directory corrupted
`ENOCSI`	No CSI structure is available	Indirect block corrupted

In addition, inodes are validated and cross checked with directories.

You should monitor the `/var/adm/sam-log` file for the preceding errors. You should watch the `/var/adm/messages` file for device errors. If a discrepancy is noted, the file system should be unmounted and checked using the `samfsck`(1M) command. You can send output from `samfsck`(1M) to both your screen and to a file by using it in conjunction with the `tee`(1) command, as follows:

```
server# samfsck -V family_set_name | tee /pathname_to_output_file
```

Nonfatal errors returned by `samfsck`(1M) are preceded by `NOTICE`. Nonfatal errors are lost blocks and orphans. The file system is still consistent if `NOTICE` errors are returned. These nonfatal errors can be repaired during a convenient, scheduled maintenance outage.

Fatal errors are preceded by `ALERT`. These errors include duplicate blocks, invalid directories, and invalid indirect blocks. The file system is not consistent if these errors occur. It is recommended that you notify LSC if the `ALERT` errors cannot be explained by a hardware malfunction.

For more information on the samfsck(1M) and tee(1) commands, see the samfsck(1M) and tee(1) man pages.

How to Repair a File System

If the samfsck(1M) command detects file system corruption by returning ALERT messages, the reason for the corruption should be determined. If hardware is faulty, it should be repaired prior to repairing the file system. Then the file system should be repaired by specifying the -F option to the samfsck(1M) command, as follows:

```
server# samfsck -F -V family_set_name
```

The samfsck(1M) command should be run when a file system is not mounted.

How to Dump and Restore QFS File Systems

This subsection describes the QFS control structure information and shows how to dump and restore QFS file systems.

NOTE

Disaster recovery procedures for SAM-FS and SAM-QFS are identical. These procedures are described in the *SAM-FS and SAM-QFS Storage and Archive Management Guide*, publication SG-0008.

File systems are made up of directories, files, and links. QFS maintains the file system by keeping track of all files in the .inodes file which is stored on a separate metadata device. All file data is stored on the data devices.

It is important to periodically perform control structure and data dumps to protect your data from a disaster. The dumps should be done at least once a day, but the frequency depends on your site's requirements. By dumping file system data on a regular basis, old files and file systems can be restored or moved from one file system to another, or even from one server to another.

NOTE

Certain restrictions apply to the capabilities of the qfsdump(1M) and qfsrestore(1M) commands. The qfsdump(1M) command supports only full dumps of specified files and directories; incremental dump is not available. The qfsdump(1M) command dumps all data of a sparse file, and the qfsrestore(1M) command restores all data. This can lead to files occupying more space on dump files and on restored file systems than anticipated. Support for sparse files is not available.

Making Backups for QFS File Systems

The QFS file system uses the `qfsdump`(1M) and `qfsrestore`(1M) commands for backing up file systems. These commands create and restore control structure and data dumps of a current directory. The `qsdump`(1M) command saves the relative path information for each file contained in a complete file system or in a portion of a file system. The `qfsdump`(1M) command must be issued on a mounted file system. You can also use these utilities for restoring a single directory or file.

LSC recommends performing full `qfsdump`(1M) dumps daily. You need to determine whether this is right for your site. The following are general guidelines for performing dumps:

- QFS dumps are performed with the file system mounted. Inconsistencies may arise as new files are being created on disk. Dumping file systems during a quiet period (a time when files are not be created or modified) is a good idea and may eliminate these inconsistencies.
- Perform dumps on a regular basis. You can run the `qfsdump`(1M) command as a `cron`(1) job by creating an entry in the `crontab` file.
- Dumps are a preventative measure against total hardware failure on the devices supporting the QFS file system. You should use the `qfsdump`(1M) command to create dumps to guard your system from such a failure.
- Ensure that you dump control structures and data for all QFS file systems. Look in `/etc/vfstab` for all file systems of type `samfs`.
- The `qfsdump`(1M) command does not support volume overflow on raw devices, but a removable media file can be used if the dump is larger than the raw device. For more information on removable media files, see the `request`(1M) command.

How to Dump QFS File Systems

The `qfsdump`(1M) command dumps file systems, and the `qfsrestore`(1M) command restores file systems generated by the `qfsdump`(1M) command. For a complete description of the syntax and options available with these commands, see the `qfsdump`(1M) and `qfsrestore`(1M) man pages.

To manually create a QFS dump for a file system, or a directory within a file system, enter the following commands:

1. Login as `root`.
2. Use the `cd`(1) command to go to the directory that contains the mount point for the file system.

   ```
   server# cd /qfs1
   ```

3. Create a dump file by entering the `qfsdump`(1M) command. For example:

   ```
   server# qfsdump -f dump_file
   ```

To automate your dump procedures, use the `cron`(1) command to create an entry in `root`'s `cron` table.

Example. The following example `cron` table entry performs a dump and manages the files within a dump directory:

```
10 0 * * * (cd /samfs1; /opt/LSCsamfs/sbin/qfsdump –B 512 –f /dev/rmt/0cb)
```

At 10 minutes after midnight, you use the `cd`(1) command to change to the mount point, and you execute the `/opt/LSCsamfs/sbin/qfsdump` command to write the data to the `/dev/rmt/0cb` tape device.

If you have multiple QFS file systems, make similar entries for each. Be sure that you save each dump in a separate file.

How to Restore QFS File Systems

This example assumes that you have a QFS dump file as described in the previous subsection.

1. Use the `cd`(1) command to move into a QFS file system. For example:

   ```
   server# cd /qfs1
   ```

2. Restore the directory using an existing dump file. For example:

   ```
   server# qfsrestore -f dump_file
   ```

How to Restore Single Files and Directories

You can also use the `qfsdump`(1M) command to restore a single file or directory, relative to the current directory. Enter the following:

1. List the name of the file or directory that you want restored.

   ```
   server# qfsrestore -ft dump_file
   ```

2. Restore the file relative to the current directory. *file_name* must exactly match the name of the file or directory as it was listed in the step above.

   ```
   server# qfsrestore  -f  dump_file  file_name
   ```

How to Prepare for a Hardware Upgrade

Whether upgrading a server, adding a new tape drive, adding an automated library, or installing a different drive into an existing automated library, it is best to perform advance planning. This subsection is intended to prepare you for hardware upgrades to devices within your environment.

The following actions are recommended prior to the upgrade:

- Determine if the hardware addition or change requires a new license from LSC. Examples of changes that do not require a license upgrade include adding memory and increasing disk cache. Examples of changes that require a license upgrade include adding more slots in an automated library and changing the model of your server.
- Read the hardware manufacturer's installation instructions carefully. Also read the documentation on adding hardware in your Solaris system administrator documentation.
- The system should be quiet with no users logged in.

- Check the equipment ordinals between your old and new master configuration files. For information on the `mcf` file, see the `mcf`(4) man page.
- In a QFS environment, use the `qfsdump`(1M) command to dump all data. For more information on this process, see the `qfsdump`(1M) man page or see the back-up procedure described later in this chapter under the heading "How to UpgradeYour Solaris Operating System (QFS Environment)".
- In SAM-FS and SAM-QFS environments, ensure that all of the files that need to be archived have an archive copy. Enter the following for each LSC SAM-FS or SAM-QFS file system to see which files do not have an archive copy (in the following example, `/sam` is the mount point):

  ```
  server# sfind /sam \!-archived \!-empty -type f -print
  ```

- In SAM-FS and SAM-QFS environments, insure that the archiver is in a wait mode. The archiver must be in a wait mode, and not running, during an upgrade. You can idle the archiver by inserting a `wait` directive into the `/etc/opt/LSCsamfs/archiver.cmd` file. For more information on the `wait` directive and the `archiver.cmd` file, see the `archiver.cmd`(4) man page.
- In SAM-FS and SAM-QFS environments, the control structure information for all file systems should be backed up prior to upgrading. For information on dumping control structures, see the *SAM-FS and SAM-QFS Storage and Archive Management Guide*, publication SG-0008.

How To Add Disk Cache to a File System

At some point, you may want to add disk partitions or disk drives in order to increase the disk cache for a file system. This is accomplished by updating the `mcf` file and using the `samgrowfs`(1M) command as described in this subsection. There is no need to remake or restore the file system.

In SAM-FS and SAM-QFS environments, note that when adding disks or partitions, the equipment ordinal of the historian may be updated. The equipment ordinal of the historian is automatically generated by the system unless you specifically call it out. For more information, see the `historian`(7) man page.

To add disk partitions or drives to an existing file system, follow these steps:

1. Unmount all the file systems you want to grow.
2. In a SAM-FS or SAM-QFS environment, enter the following commands:

   ```
   server# samcmd idle eq          # see NOTE
   server# samd stop
   ```

NOTE

The drives in your environment must be idled prior to issuing the **samd stop** command, so enter a **samcmd idle** *eq* command for each *eq* configured in your mcf file. Alternatively, you can also idle the drives by using the samu(1M) operator utility or by using either the robottool(1M) or libmgr(1M) Graphical User Interface (GUI) tools. For more information on the samcmd(1M) command, see the samcmd(1M) man page.

3. Edit the /etc/opt/LSCsamfs/mcf file. Save the changes, and quit the editor.

NOTE

Add one more partition after the existing partitions for the file system to which you want to add disk cache.

To increase the size of a QFS file system, at least one new meta equipment must be added. Zero or more data equipment can be added.

Do not change the equipment identifier name in the /etc/opt/LSCsamfs/mcf file. If the name in the mcf file does not match the name in the superblock, the file systems can no longer be mounted. Instead, the following message is logged in /var/adm/messages:

```
WARNING SAM-FS superblock equipment identifier <id>s on eq
<eq> does not match <id> in mcf
```

If you want to change the name of the file system in conjunction with increasing its size, you must do so in separate operations. Use the samfsck(1M) command with its -R option to rename the file system, and then use the samgrowfs(1M) command to increase the size of the renamed file system. For more information on these commands, see the samfsck(1M) and samgrowfs(1M) man pages.

4. Run the samgrowfs(1M) command on the file system (named samfs1 in this example) that is affected:

```
server# samgrowfs samfs1
```

5. Mount the file systems.

How To Replace Disks in a File system

At some point, you may want to perform the following tasks:

- Change disks or partitions
- Add disks or partitions
- Remove disks or partitions

To accomplish these tasks, you need to back up and recreate the file system. Specifically, you need to recreate the file system by following these steps:

1. Create a control structure dump of each file system to be upgraded using a unique dump file name for each file system. If any file warnings are issued, these files need to be backed up before unmounting the file systems. This process differs a little depending on your file system. If you are using the QFS file system, use `qfsdump`(1M) to create the dump. If you are using SAM-FS or SAM-QFS, use `samfsdump`(1M) to create the dump.

   Example 1. SAM-FS and SAM-QFS example. In this example, `dump_file` is the newly created dump structure and `samfs.mt.pt` is a SAM-FS mount point:

   ```
   server# cd /samfs.mt.pt
   server# samfsdump -T -f /dumpster/dump_file
   ```

   Example 2. QFS example. In this example, `dump_file` is the newly created dump structure and `qfs.mt.pt` is the QFS mount point:

   ```
   server# cd /qfs.mt.pt
   server# qfsdump -T -f /dumpster/dump_file
   ```

   For information on creating a SAM-FS or SAM-QFS control structure dump, see the *SAM-FS and SAM-QFS Storage and Archive Management Guide*, publication SG-0008. For information on creating a QFS control structure dump, see "How to Dump and Restore QFS File Systems" earlier in this chapter.

2. For a SAM-FS or SAM-QFS file system, enter the following commands:

   ```
   server# samcmd idle eq          # see NOTE
   server# samd stop
   ```

NOTE

The drives in your SAM-FS environment must be idled prior to issuing the **`samd stop`** command, so enter a **`samcmd idle`** *eq* command for each *eq* configured in your `mcf` file. Alternatively, you can also idle the drives by using the `samu`(1M) operator utility or by using either the `robottool`(1M) or `libmgr`(1M) Graphical User Interface (GUI) tools. For more information on the `samcmd`(1M) command, see the `samcmd`(1M) man page.

3. Unmount all file systems.

4. Edit the `/etc/opt/LSCsamfs/mcf` file to include the new disk or partitions. Save the changes and quit the editor.

   Do not change the equipment identifier name in the `/etc/opt/LSCsamfs/mcf` file. If the name in the `mcf` file does not match the name in the superblock, the file systems can no longer be mounted. Instead, the following message is logged in `/var/adm/messages`:

   ```
   WARNING SAM-FS superblock equipment identifier <id>s on
   eq <eq> does not match <id> in mcf
   ```

If you want to change the name of the file system in conjunction with increasing its size, you must do so in separate operations. Use the `samfsck`(1M) command with its `-R` option to rename the file system, and then use the `samgrowfs`(1M) command to increase the size of the renamed file system. For more information on these commands, see the `samfsck`(1M) and `samgrowfs`(1M) man pages.

5. Make a new file system. Run the `sammkfs`(1M) command on the file system being created (named `samfs1` in this example):

```
server# sammkfs samfs1
```

6. Mount the file systems.
7. Use the `cd`(1) command to change to the mount point of the file system.
8. Restore each file system directory using the control structure dump file created in Step 1. If you are using the QFS file system, use the `qfsrestore`(1M) command. If you are using the SAM-FS or SAM-QFS file system, use the `samfsrestore`(1M) command.

   Example:

```
server# samfsrestore -T -f /dumpster/dump_file
```

How To Upgrade a Server

When it comes time to upgrade the server being used for the file system, you should take the following into account:

- It is wise to move to the new server while the existing server is still in operation. This allows time to install, configure, and test the new hardware platform with your applications.
- Moving to a new server is equivalent to installing the LSC software for the first time.

  In SAM-FS and SAM-QFS environments, you need to reinstall the software and update the configuration files (specifically the `mcf` file, the `/kernel/drv/st.conf` file, and the `/etc/opt/LSCsamfs/inquiry.conf` file). In addition, you need to copy your existing `archiver.cmd` and `defaults.conf` files to the new system, configure system logging, and so on.

  You can use the installation instructions in the *QFS, SAM-FS, and SAM-QFS Software Installation and Configuration Guide*, publication SG-0007, when re-installing the software.
- The license key needs to be updated. License keys are tied to the CPU host ID. Replacing the system requires a new license.
- Before powering down the old server, perform a control structure dump for each file system. For a QFS file system, use the `qfsdump`(1M) command; in SAM-FS and SAM-QFS environments, use the `samfsdump`(1M) command.

  This dump file is used to recreate the file system on the new server. For more information on completing a control structure dump, see the `qfsdump`(1M) or `samfsdump`(1M) man pages or see the *SAM-FS and SAM-QFS Storage and Archive Management Guide*, publication SG-0008.

How to Upgrade Your Solaris Operating System (SAM-FS and SAM-QFS Environments)

Many of the steps involved in upgrading your Solaris level are identical to the steps involved in upgrading your SAM-FS or SAM-QFS environment. Some of the steps in this procedure reference procedures in the *QFS, SAM-FS, and SAM-QFS Installation and Configuration Guide*, publication SG-0007.

Perform the following steps if you wish to upgrade a Solaris operating system upon which the SAM-FS or SAM-QFS system is installed:

Step 1: Back Up Each File System

If you do not have current backup files for each of your file systems, create them now using `samfsdump`(1M) and by copying the inodes file using `dd`(1M), as follows:

1. Make sure that any site-customized system files or configuration files (such as `mcf`, `archiver.cmd`, `defaults.conf`, `samfs.cmd`, `inquiry.conf`, `ReservedVSNs`, and so on.) have been copied to a backup location.

   In particular, make sure that you have backup copies of files in the `/etc/opt/LSCsamfs` directory, files in the `/var/opt/LSCsamfs` directory, library catalogs, the historian, and any parameter files for network-attached automated libraries.

   If you do not know the names and locations of your catalog files, examine the `mcf` file with `vi`(1) or another viewing command and find the first `rb` entry in the `mcf` file. That entry contains the name of the library catalog file. If this is not present, the default location is `/var/opt/LSCsamfs/`*catalog*.

2. Back up each SAM-FS file system using `samfsdump`(1M). The location to which you dump each system should be outside the file system.

   The following example assumes that you have a file system named `samfs1` that you want to back up to `samfs1.bak`, which exists outside of the SAM-FS file system:

   ```
   server# cd /samfs1
   server# samfsdump –f /csd_dump_dir/samfs1.bak
   ```

   The `samfsdump` command dumps file names and inode information, not data. For more information on this, see the `samfsdump`(1M) man page.

3. Write this information to a file for safe keeping, as follows:

```
server# /bin/dd if=/sam1/.inodes of=/inode_dump_dir/samfs1.inodes bs=512k
```

You need back up files for each file system, so repeat the preceding steps for each file system in your LSC environment.

Step 2: Stop the SAM Daemons

To stop the file system, enter the following command:

```
server# samcmd idle eq              # see NOTE
server# samd stop
```

NOTE

The drives in your SAM-FS or SAM-QFS environment must be idled prior to issuing the **samd stop** command, so enter a **samcmd idle** *eq* command for each *eq* configured in your mcf file. Alternatively, you can also idle the drives by using the samu(1M) operator utility or by using either the robottool(1M) or libmgr(1M) Graphical User Interface (GUI) tools. For more information on the samcmd(1M) command, see the samcmd(1M) man page.

The samd(1M) command is installed in /opt/LSCsamfs/sbin.

Step 3: Unmount the File Systems

Unmount the file systems using the procedure described earlier in this chapter in the subsection called, “How to Unmount a File System”.

Step 4: Remove Existing LSC Software

Use the pkginfo(1) command, as follows, to determine which software packages are installed on your system:

```
server# pkginfo | grep LSC
```

Use the pkgrm(1M) command to remove the existing software. You must remove all existing LSC packages before installing the new packages or the new operating system level. If you are using any optional LSC packages make sure that you remove these prior to the main LSCsamfs package.

The following example commands remove all of the LSC packages:

```
server# pkgrm LSCibm
server# pkgrm LSCstk
server# pkgrm LSCdst
server# pkgrm LSCsony
server# pkgrm LSCgui
server# pkgrm LSCjre
server# pkgrm LSCdoc
server# pkgrm LSCsamfs          (must be last package removed)
```

Step 5: Upgrade Solaris

Install the new Solaris revision using the Solaris upgrade procedures for the operating system level you are installing.

Step 6: Add the Packages

The LSC software package uses the Solaris packaging utilities for adding and deleting software. As such, you must be logged in as superuser (root) to make changes to software packages. The `pkgadd`(1M) command prompts you to confirm various actions necessary to upgrade the LSC packages.

On the CD-ROM, the LSC packages and all optional products reside in the `/cdrom/cdrom0` directory.

To satisfy product dependencies, you must upgrade the `sampkg` first. Run the `pkgadd`(1M) command to upgrade all packages, answering `yes` to each question:

`server#` **`pkgadd -d sampkg`** (must be first)

If your environment includes certain network-attached automated libraries, such as certain models from StorageTek, Ampex, or IBM, you may need to install one or more vendor-specific media changer daemons. These daemons are supplied by LSC. For information on whether or not you need to install a vendor daemon package, see the *SAM-FS and SAM-QFS Storage and Archive Management Guide*, publication SG-0008.

To install one or more of these daemon packages, install them using `pkgadd`(1M), as follows:

`server#` **`pkgadd -d samstk`** (optional StorageTek package)

`server#` **`pkgadd -d samdst`** (optional Ampex package)

`server#` **`pkgadd -d samibm`** (optional IBM package)

`server#` **`pkgadd -d samsony`** (optional Sony package)

An optional set of GUI tools is included with the SAM-FS and SAM-QFS software packages. The GUI tools require the presence of a Java runtime environment. Add these packages as follows:

`server#` **`pkgadd -d samjre`** (optional)

`server#` **`pkgadd -d samgui`** (optional)

LSC documentation is available in PDF format. Add this package as follows:

`server#` **`pkgadd -d samdoc`** (optional)

During the installation, the system detects the presence of conflicting files and prompts you to indicate whether or not you want to continue with the installation. You can go to another window and copy the files you wish to save to an alternate location.

Step 7: Mount the File System(s) (Optional)

You must perform this step if you have not modified the `/etc/vfstab` file to have `yes` or `delay`.

Use the `mount`(1M) command to mount the file systems and continue operation with the upgraded SAM-FS or SAM-QFS software.

In the following example, `samfs1` is the file system name to be mounted:

```
server# mount samfs1
```

How to Upgrade Your Solaris Operating System (QFS Environment)

Many of the steps involved in upgrading your Solaris level are identical to the steps involved in upgrading your SAM-QFS environment. Some of the steps in this procedure reference procedures in the *QFS, SAM-FS, and SAM-QFS Installation and Configuration Guide*, publication SG-0008.

Perform the following steps if you wish to upgrade a Solaris operating system upon which the QFS system is installed:

Step 1: Back Up Each File System

If you do not have current backup files for each of your file systems, create them now using `qfsdump`(1M):

1. Make sure that any site-customized system files or configuration files (such as `mcf`, `archiver.cmd`, `defaults.conf`, `samfs.cmd`, `inquiry.conf`, `ReservedVSNs`, and so on.) have been copied to a backup location. In particular, make sure that you have backup copies of files in the `/etc/opt/LSCsamfs` directory, files in the `/var/opt/LSCsamfs` directory, library catalogs, the historian, and any parameter files for network-attached automated libraries.

   If you do not know the name and locations of your catalog files, examine the `mcf` file with `vi`(1) or another viewing command and find the first `rb` entry in the `mcf` file. That entry contains the name of the library catalog file. If this is not present, the default location is `/var/opt/LSCsamfs/`*catalog*.

2. Back up each file system using `qfsdump`(1M). The location to which you dump each system should be outside the QFS file system.

   The following example assumes that you have a file system named `qfs1` that you want to back up to `qfs1.bak`, which exists outside of the QFS file system:

   ```
   server# cd /qfs1
   server# qfsdump –f /csd_dump_dir/qfs1.bak
   ```

   The `qfsdump` command dumps file names and inode information, not data. For more information on this, see the `qfsdump`(1M) man page.

You need back up files for each file system, so repeat the preceding steps for each file system in your QFS file system.

Step 2: Unmount the File Systems

Unmount the file systems using the procedure described earlier in this chapter in the subsection called, “How to Unmount a File System”.

Step 3: Remove Existing LSC Software

Use the `pkginfo`(1) command, as follows, to determine which LSC software packages are installed on your system:

```
server# pkginfo | grep LSC
```

Use the `pkgrm`(1M) command to remove the existing LSC software. You must remove all existing LSC packages before installing the new packages or the new operating system level. If you are using any optional LSC packages make sure that you remove these prior to the main `LSCsamfs` package.

The following example commands remove all of the LSC packages:

```
server# pkgrm LSCdoc        (if installed)
server# pkgrm LSCqfs        (must be last package removed)
```

Step 4: Upgrade Solaris

Install the new Solaris revision using the Solaris upgrade procedures for the operating system level you are installing.

Step 5: Add the Packages

The QFS software package uses the Solaris packaging utilities for adding and deleting software. As such, you must be logged in as superuser (root) to make changes to software packages. The `pkgadd`(1M) command prompts you to confirm various actions necessary to upgrade the LSC packages.

On the CD-ROM, the QFS packages and all optional products reside in the `/cdrom/cdrom0` directory.

To satisfy product dependencies, you must upgrade the `sampkg` first. Run the `pkgadd`(1M) command to upgrade all packages, answering `yes` to each question:

```
server# pkgadd -d samqfs     (must be first)
```

LSC documentation is available in PDF format. Add this package as follows:

```
server# pkgadd -d samdoc     (optional)
```

During the installation, the system detects the presence of conflicting files and prompts you to indicate whether or not you want to continue with the installation. You can go to another window and copy the files you wish to save to an alternate location.

Step 6: Mount the File System(s) (Optional)

You must perform this step if you have not modified the `/etc/vfstab` file to have `yes` or `delay`.

Use the `mount`(1M) command to mount the file systems and continue operation with the upgraded QFS software.

In the following example, `qfs1` is the name of the file system to be mounted:

```
server# mount qfs1
```

Advanced Topics
Chapter 5

This chapter discusses advanced topics that are beyond the scope of basic system administration and usage.

The following topics are presented.

- Striping the `.inodes` file
- Using the `setfa`(1) command to set file attributes
- Accommodating large files
- Setting up the SAN-QFS file system (QFS file system)

Striping the `.inodes` File (QFS and SAM-QFS File Systems)

QFS and SAM-QFS `.inodes` files are allocated in 16-kilobyte blocks as needed. An inode uses 512 bytes. In a QFS or SAM-QFS file system, the metadata devices (device type `mm`) are striped at the 16-kilobyte DAU level. This means that the first 32 inodes would be created on the first metadata device, then the next 32 inodes would be created on the next metadata device.

The stripe specification is taken from the `-o stripe=`*n* option on the `mount`(1M) command. Thus, if `-o stripe=0`, the file system writes to one meta device until it is full, then it switches to the next one.

If you would like to stripe the metadata, but round-robin the file data, this can be accomplished using the `setfa`(1) command on the `.inodes` file, as follows:

```
server# setfa -s 1 /sam/.inodes
```

Using the `setfa`(1) Command to Set File Attributes

LSC file systems allow users to set performance attributes for files and directories. These performance features can be enabled by applications on a per-file basis. The following subsections describe how the application programmer can use these features to select file attributes for files and directories, to preallocate file space, specifying the allocation method for the file, and specifying the disk stripe width.

Selecting File Attributes for Files and Directories

File attributes are set using the `setfa`(1) command. The `setfa`(1) command sets attributes on a new or existing file. The file is created if it does not already exist.

Attributes can be set on a directory as well as a file. When using `setfa`(1) with a directory, files and directories created within that directory inherit the attributes set in the original directory. To reset attributes on a file or directory to the default, use the `-d` (default) option. When the `-d` option is used, attributes are first reset to the default and then other attributes are processed.

Preallocating File Space

A user can preallocate space for a file. This space is associated with a file so that no other files in the file system can use this space. Preallocation ensures that both space is available for a given file, which avoids a file system full condition, and that this space is allocated sequentially as defined by the file system. Preallocation is assigned at the time of the request rather than when the data is actually written to disk.

Note that space can be wasted when preallocating files. If the file size is less than the allocation amount, the kernel allocates space to the file from the current file size up to the allocation amount. When the file is closed, space below the allocation amount is not freed.

A file is preallocated by using the `setfa`(1) command with the `-l` (lowercase letter `L`) option and specifying the file length in bytes (`b`), kilobytes (`k`), megabytes (`m`), or gigabytes (`g`).

For example, to preallocate a 1-gigabyte file named `/qfs/file_alloc`, enter the following:

```
server# setfa -l 1g /qfs/file_alloc
```

After space for a file has been preallocated, truncating a file to 0 length or removing the file returns all space allocated for a file. There is no way to return only part of a file's preallocated space to the file system. In addition, if a file is preallocated in this manner, there is no way to extend the file beyond its preallocated size in future operations.

Selecting A File Allocation Method and Stripe Width

By default, a file created uses the allocation method and stripe width specified at mount time (see the `mount_samfs`(1M) man page). However, a user may wish to use a different allocation scheme for a file or directory of files, and this can be accomplished by using the `setfa`(1) command with the `-s` (stripe) option.

The allocation method can be either round-robined or striped. The `-s` option determines the allocation method and the stripe width, as follows:

`-s` *stripe*	Allocation Method	Stripe Width	Explanation
0	Round-robin	Not applicable	The file is allocated on one device until that device has no space.
1-255	Striped	1 – 255 DAUs	The file stripes across all disk devices with this number of DAUs per disk.

The following example shows how a file can be created explicitly by specifying a round-robined allocation method. The command also preallocates 100 megabytes of space for a file called `/qfs/100MB.rrobin`:

```
server# setfa –s 0 –l 1m /qfs/100MB.rrobin
```

The next example shows how a file can be created explicitly by specifying a striped allocation method with a stripe width of 64 DAUs. Preallocation is not used.

```
server# setfa –s 64 /qfs/file.stripe
```

Selecting a Striped Group Device (QFS Only)

A user can specify that a file begin allocation on a particular striped group. If the file allocation method is round robin, the file is allocated on the designated stripe group.

For example, the following `setfa`(1) commands specify that `file1` and `file2` be independently spread across two different striped groups:

```
server# setfa -go –so file1
server# setfa -g1 –so file2
```

This capability is particularly important for turnkey applications that must achieve raw device speeds. For more information, see the `setfa`(1) man page.

Accommodating Large Files

When manipulating very large files, careful attention must be given to the size of disk cache available on the system. If you try to write a file that is larger than your disk cache, behavior differs depending on the type of LSC file system you are using, as follows:

- If you are using the QFS file system, the system returns an `ENOSPC` error.
- If you are using the SAM-FS or SAM-QFS file system, the program blocks, waiting for space that may never exist, because there is not enough disk space available to handle such requests.

If you are operating within a SAM-FS or SAM-QFS environment and if your application requires writing a file that is larger than the disk cache, you can segment the file using the `segment`(1) command. For more information on the `segment`(1) command, see the `segment`(1) man page or see the *SAM-FS and SAM-QFS Storage and Archive Management Guide*, publication SG-0008.

Note that segmented file features are compatible with the SAM-FS and SAM-QFS file systems and are enabled through the SAM-Segment license key.

WARNING

Even though the SAM-FS and SAM-QFS file systems do not use the `tar`(1) command to read or write data onto cartridges, data appears on the cartridges in the industry standard `tar`(1) format. This is done for compatibility reasons. In addition, this practice allows users to read cartridges even when the SAM-FS or SAM-QFS file system is not available.

For disaster recovery purposes, and for when SAM-FS or SAM-QFS is not available, LSC provides the `star`(1) command. This command can be used to restore data on any UNIX system. The `star`(1) command overcomes the `tar`(1) command's size limitations. The SAM-FS and SAM-QFS file systems do not use the standard UNIX `tar`(1) command for disaster recovery purposes for the following reasons:

1. `tar`(1) does not support a block size that is large enough for SAM-FS and SAM-QFS media.
2. `tar`(1) can only recover files that are up to 2 gigabytes in length.

Note, however, that `star`(1M) does not work with files that are 1 terabyte or larger in size.

Setting Up the SAN-QFS File System (QFS File System)

The QFS file system can be used in conjunction with fiber-attached devices in a Storage Area Network (SAN). When the SAN-QFS license key from LSC is enabled, the QFS file system enables high-speed access to data using software such as Tivoli SANergy File Sharing. LSC supports the SAN-QFS file system with the SANergy File Sharing 2.2 protocol.

The SAN-QFS file system allows multiple users to access the same data at full disk speeds. This product can be especially useful for database, data streaming, web page service, or any application that demands high performance, shared-disk access in a heterogeneous environment.

A SAN-QFS license key is needed in order to enable the features included in the SAN-QFS file system.

The following procedure describes how to enable the SAN-QFS file system.

Step 1: Verify Your Environment

The SAN-QFS software must be installed on a server upon which the QFS file system is already fully operational. In addition, you must have a SAN-QFS license key from LSC.

Step 2: Mount the File System and Enable NFS Access

Mount the QFS file system on your server and enable NFS access to client hosts by using the following command:

```
server# share qfs_file_system_name
```

In the preceding format, *qfs_file_system_name* is the name of your QFS file system. For example, `qfs1`.

You may wish to automatically enable this access at boot time by editing the file system table. For example, on Solaris systems, you can edit file `/etc/dfs/dfstab`.

For more information on the `share`(1M) command, see the `share`(1M) or `share_nfs`(1M) man pages.

Step 3: Add the File System to Each Client

Edit the file system table on each client machine and add the *qfs_file_system_name* from Step 2 to the table.

Example. On Solaris systems, edit the `/etc/vfstab` file and add a line similar to the following:

```
server:/qfs1  -  /qfs1  samfs  -  yes  stripe=1
```

For more information on editing the `/etc/vfstab` file, see the *QFS, SAM-FS, and SAM-QFS Installation and Configuration Guide*, publication SG-0007.

Step 4: Mount the QFS File System

Use the `mount`(1M) command to mount the QFS file system on each client. For example:

```
client# mount qfs1
```

You must enter one `mount`(1M) command per client. For more information on the `mount`(1M) command, see the `mount`(1M) or the `mount_samfs`(1M) man pages.

Step 5: Configure the SAN-QFS File System

Use the `config`(1M) command (stored in `/opt/SANergy/config`) to invoke the SANergy configuration tool. The SANergy configuration tool has a graphical user interface, and you should provide the information requested at each step in its process. For more information on this tool, see your Tivoli SANergy documentation.

Performance Topics
Chapter 6

This chapter discusses the following performance topics:

- Increasing file transfer performance
- I/O performance
- Qwrite
- Setting the flush-behind rate

Increasing File Transfer Performance

LSC file systems are tuned to work with a mix of file sizes. You can increase the performance of disk file transfers by enabling file system settings. This process is described in the following subsections.

Step 1: Set the Maximum Device Read/Write Directive

The `maxphys` parameter in the Solaris `/etc/system` file controls the maximum number of bytes a device drive reads or writes at any one time. Set `maxphys` as follows:

```
set maxphys = 0x800000
```

Step 2: Set the SCSI Disk Maximum Transfer Parameter

The `sd` driver enables large transfers for a specific file by looking for the `sd_max_xfer_size` definition in the `/kernel/drv/sd.conf` file. If it is not defined, it uses the value defined in the `sd` device driver definition `SD_MAX_XFER_SIZE`, which is 1024*1024 bytes.

To enable large transfers, set the `sd_max_xfer_size` equal to the `maxphys` setting from step 1.

Example. The following lines are from a sample `/kernel/drv/sd.conf` file:

```
name="sd" class="scsi"
     sd_max_xfer_size=0x800000
     target=2 lun=0;

name="sd" class="scsi"
     sd_max_xfer_size=0x800000
     target=3 lun=0;
```

Step 3: Set the Fiber Disk Maximum Transfer Parameter

The `ssd` driver enables large transfers for a specific file by looking for the `ssd_max_xfer_size` definition in the `/kernel/drv/sd.conf` file. If it is not defined, it uses the value defined in the `ssd` device driver definition `SSD_MAX_XFER_SIZE`, which is 1024*1024 bytes.

Add the following line at the end of the `/kernel/drv/ssd.conf` file:

```
set ssd_max_xfer_size = 0X800000;
```

Step 4: Reboot the System

To activate the system settings from the previous steps, you must reboot the system.

Step 5: Set the `writebehind` *Parameter*

The `writebehind` parameter specifies the number of bytes that are written behind by the file system when performing paged I/O on an LSC file system. This parameter is specified in units of kilobytes and is truncated to an 8-kilobyte multiple.

The `writebehind` size should be set to a multiple of the RAID 5 stripe size for both hardware and software RAID 5. The RAID 5 stripe size is the number of data disks multiplied by the stripe width.

Example. Assume that you configure a RAID 5 device with 3 data disks plus 1 parity disk (3+1) with a stripe width of 16k. The `writebehind` value should be 48k, 96k, or some other multiple, to avoid the overhead of the read-modify-write RAID 5 parity generation.

For QFS and SAM-QFS file systems, the DAU (`sammkfs -a` option) should also be a multiple of the RAID 5 stripe size. This assures that the blocks are contiguous.

You should test the system performance after resetting the `writebehind` size. The following example shows testing timings of disk writes:

```
server# timex dd if=/dev/zero of=/sam/myfile bs=256k count=2048
```

The `writebehind` parameter can be set from either a mount option, from within the `samfs.cmd` file, or from a command within the `samu`(1M) utility. For information on enabling this from a mount option, see the `-o writebehind=`*n* option on the `mount_samfs`(1M) man page. For information on enabling this from the `samfs.cmd` file, see the `samfs.cmd`(4) man page. For information on enabling this from within `samu`(1M), see the `samu`(1M) man page (SAM-FS and SAM-QFS file systems only).

The `maxcontig` setting is retained for backward compatibility and is in units of 16 kilobytes.

Step 6: Set the `readahead` *Parameter*

The `readahead` parameter specifies the number of bytes that are read ahead by the file system when performing paged I/O on an LSC file system. This parameter is specified in units of kilobytes and is truncated to an 8-kilobyte multiple.

Increasing the size of the `readahead` parameter increases the performance of large file transfers, but only to a point. You should test the performance of the system after resetting the `readahead` size until you see no more improvement in transfer rates. The following is an example method of testing timings on disk reads:

```
server# timex dd if=/sam/myfile of=/dev/null bs=256k
```

The `readahead` parameter should be set to a size that increases the I/O performance for paged I/O. Also note that a big `readahead` size can hurt performance. You should test various `readahead` sizes for your environment. It is important to consider the amount of memory and number of concurrent streams when you set the `readahead` value.

The `readahead` setting can be enabled from either a mount option, from within the `samfs.cmd` file, or from a command within the `samu`(1M) utility. For information on enabling this from a mount option, see the `-o readahead=`*n* option on the `mount_samfs`(1M) man page. For information on enabling this from the `samfs.cmd` file, see the `samfs.cmd`(4) man page. For information on enabling this from within `samu`(1M), see the `samu`(1M) man page (SAM-FS and SAM-QFS file systems only).

The `maxcontig` setting is retained for backward compatibility and is in units of 16 kilobytes.

Step 7: Set the Stripe Width

The `-o stripe=`*n* option on the `mount`(1M) command specifies the stripe width for the file system. The stripe width is based on the disk allocation unit (DAU) size. The *n* argument specifies that *n* * DAU bytes are written to one device before switching to the next device. The DAU size is set when the file system is initialized by the `sammkfs`(1M) command's `-a` option.

If `-o stripe=0` is set, files are allocated to file system devices using the round-robin allocation method. If `-o stripe` is set to an integer greater than `0`, files are allocated to file system devices using the stripe method. To determine the appropriate `-o stripe=`*n* setting, try varying the setting and taking performance readings.

The stripe width can also be set from the `/etc/vfstab` file or from the `samfs.cmd` file. Options on the `mount`(1M) command override settings in the `/etc/vfstab` file. Settings in the `/etc/vfstab` file override directives in the `samfs.cmd` file.

For more information on the `mount`(1M) command, see the `mount_samfs`(1M) man page. For more information on the `samfs.cmd` file, see the `samfs.cmd`(4) man page.

I/O Performance

LSC file systems support 64-bit files for both paged and direct I/O. The type of I/O can be selected by using the Solaris `directio`(3C) function call. You can also set the direct I/O attribute by using the `setfa`(1) command, as follows:

```
server# setfa -D
```

The following subsections describe explicit I/O, as specified for a file or directory, and the default I/O that LSC file systems provide.

Direct I/O

The `-D` option sets the direct I/O attribute for a file and/or directory and is used with the `setfa`(1) command and the `sam_setfa`(3) library routine. If applied to a directory, the direct I/O attribute is inherited by any files and directories created in that directory. After the `-D` option is set, the file uses direct I/O. Data is transferred directly between the user's buffer and the disk, which means that much less time is spent in the system.

Set this attribute only for large block aligned sequential I/O. The default I/O mode is buffered (uses the page cache). Any I/O attempted on these files is direct. For more information on these commands, see the `setfa`(1) and `sam_setfa`(3) man pages.

To enable direct I/O on a file system basis, see the `-o forcedirectio` option on the `mount_samfs`(1M) command.

Default I/O

Automatic, direct I/O switching can be enabled. By default, it is disabled. Automatic, direct I/O switching allows the system to perform a certain amount of consecutive I/O operations and then automatically switch from paged I/O (also called buffered or cached I/O) to direct I/O. This capability should reduce page cache usage on large I/O operations. To enable this, use the `dio_wr_consec` and `dio_rd_consec` parameters as directives in the `samfs.cmd` file or as options to the `mount`(1M) command.

For more information on these options, see the `mount_samfs`(1M) or `samfs.cmd`(4) man pages.

Qwrite (QFS and SAM-QFS file systems only)

By default, the QFS and SAM-QFS file systems disable simultaneous reads and writes to the same file. This is the mode defined by the UNIX `vnode` interface standard, which gives exclusive access to only one write while other writers/readers must wait. Qwrite enables simultaneous reads and writes to the same file from different threads.

The Qwrite feature can be used in database situations to enable multiple simultaneous transactions to the same file. Database applications typically manage large files and issue simultaneous reads and writes to the same file. Unfortunately, each system call to a file acquires and releases a read/write lock inside the kernel. These locks prevent the overlapping (or simultaneous) operations to the same file. Since these applications usually implement file locking mechanisms of their own, the kernel locking mechanism impedes performance by unnecessarily serializing I/O.

The `-o qwrite` option on the `mount`(1M) command bypasses the file system locking mechanisms and lets the application control data access. If `qwrite` is specified, the file system enables simultaneous reads and writes to the same file from different threads. This option improves I/O performance by queuing multiple requests at the drive level.

The following example of uses Qwrite on a database file system:

```
server# mount -F samfs -o qwrite /dev/dsk/x0t1d0s1 /db
```

For more information on this feature, see the `qwrite` directive on the `samfs.cmd`(4) man page or the `-o qwrite` option on the `mount_samfs`(1M) man page.

Setting the Flush-behind Rate

Two mount parameters control the flush-behind rate for pages written sequentially and stage pages. The `flush_behind` and `stage_flush_behind` mount parameters are read from the `samfs.cmd` file, the `/etc/vfstab` file, or from the `mount`(1M) command.

The `flush_behind` mount parameter sets the maximum flush-behind value. Modified pages that are being written sequentially are written to disk asynchronously to help the Solaris VM layer keep pages clean. This parameter has the following format:

```
flush_behind = n
```

To enable this feature, set *n* to be an integer, $16 \leq n \leq 8192$. By default, *n* is set to `0`, which disables this feature. The *n* argument is specified in kilobyte units.

The `stage_flush_behind` mount parameter sets the maximum stage flush-behind value. Stage pages that are being staged are written to disk asynchronously to help the Solaris VM layer keep pages clean. This parameter has the following format:

```
stage_flush_behind = n
```

To enable this feature, set *n* to be an integer such that, $16 \leq n \leq 8192$. By default, *n* is set to `0`, which disables this feature. The *n* argument is specified in kilobyte units.

For more information on these mount parameters, see the `mount_samfs`(1M) man page or the `samfs.cmd`(4) man page.

LSC Product Support
Appendix A

This appendix includes the following topics:

- LSC software support
- How to report a problem
- What LSC does when your Authorized Service Provider (ASP) reports a problem
- LSC support contacts

LSC Software Support

The following subsections relate to problems or questions regarding LSC software products, documentation, or support

ASPs

LSC sells and supports its products through ASPs. Your ASP is generally a specialist within your country or industry market that provides support for LSC products. The ASPs provide help desk assistance and have LSC-trained service representatives. The ASPs provide primary support on the product including taking the initial software support call and providing immediate problem and question resolution whenever possible.

LSC Support Center

When an ASP cannot provide a resolution, the ASP escalates the problem to the LSC support center located in Minnesota. These secondary support calls require a higher level of expertise for problems that cannot be resolved by the Level-1 ASPs. If you have an ASP, the escalation of a problem to the LSC support center must be done through the ASP. Do not report a problem directly to LSC. You must use your ASP.

How to Report a Problem

This subsection describes the steps LSC suggests to report your problem to your ASP. If your ASP has a different procedure, please follow the ASP's procedure instead.

What to Do Before You Call

The first step to take when you encounter a problem or a technical question is to review the product documentation. It is possible that a solution or answer is provided in the documentation. If the documentation does not provides a solution, take the following steps before you call your support ASP.

Step 1: Identify the Severity Level of the Problem

You must determine how critical the software problem is, based on the severity level descriptions in this subsection.

The severity levels range from A to D, with severity level A being the most critical. The severity levels only apply to processing software problem incidents. All problems should be reported by email if possible to avoid errors. We can help you in the most efficient way if you have emailed a description of the problem and have sent the output of the `info.sh` script prior by FTP prior to telephoning. In the definitions that follow, a *software problem* refers to both software and documentation problems.

The problem severity levels are:

- Severity Level A - The software product is non-operational, resulting in a critical system condition requiring immediate resolution. Support personnel may require continuous access to your resources until a workaround or resolution is provided. When reporting a Severity A incident, a telephone call after sending materials by email is preferred. Email without a call is acceptable, but it may delay an ASP or LSC response. LSC requires that any support calls at Severity Level A to LSC include availability of the customer system administrators to LSC.
- Severity Level B - The software product is operational, but it is severely restricted in functionality or presents a system degradation. When reporting a Severity B incident, a telephone call after sending materials by email is preferred, although email alone is acceptable.
- Severity Level C - The software product is operational, but functional limitations or restrictions that are not critical to the overall system operations are present. When reporting a Severity C incident, email is preferred.
- Severity Level D - Problems that have little, or no, impact on system operations are present. Severity D incidents should be reported only via email.

Step 2: Record Your Company Information

You will be asked to provide the following information:

- Your company name and customer number. For example, Company X, L0666.
- Your name and telephone number. Also provide the name and number for an alternate contact, if possible. For Severity A or B, a pager number is also requested.
- Your email address. For example, `jjohnson@amcorp.com`.
- Details of the software problem or technical question.

- Severity level of the software problem or a specific time and date by which you require a response to a technical question. For more information, see the previous step, "Step 1: Identify the Severity Level of the Problem". You will be given an incident number to track this support call.

Step 3: Record Your Site and Configuration Information

You should have the following information accessible:

- The name of the software product, including the release level. For example, SAM-FS 3.5.0. This can be obtained by entering the following command:

```
server# pkginfo –l LSCsamfs
```

- The names and releases of your system software, such as the operating systems or any other appropriate software. For example, Solaris 2.7.

Step 4: Detail the Problem

Use the LSC Problem Identification Checklist (Table A-1) to help you collect details about the software or documentation problem. You may make copies of the checklist. Collect this information for each problem or question.

Keep a record of your checklist responses in a convenient place for reference. Your ASP and the LSC software support center may request an answer to some or all of the questions.

Step 5: Gather Diagnostic Output

During problem resolution, your ASP and the LSC support center may request that you provide specific diagnostic material. The output of the `info.sh`(1M) command provides us with a current snapshot of your system configuration as it relates to your SAM-FS or SAM-QFS environment. Only limited log information is captured by this script, so make sure you run `info.sh` when the problem is occurring or as soon a possible after it occurs.

Submit a separate copy of the `info.sh`(1M) output for each problem you report. The output should be sent by FTP to your ASP. LSC may ask you to send information to the LSC site, but unless otherwise directed, you should always work through your ASP. The following guidelines apply when sending diagnostic information:

- Do not send SAM reports as attachments to email. If your ASP has directed you to send a SAM report to LSC, send it to the LSC FTP site, which is `ftp.lsci.com`.
- Do not send any other files or core dumps unless you have been specifically requested to do so by staff at the LSC support center.
- Do not compress the ASCII files unless you use the Solaris `compress`(1) utility. Please do not zip these files on a PC using Microsoft Windows software.

Occasionally a problem is reported many hours or even days after the problem first occurred. Because the SAM-FS and SAM-QFS environments generate a lot of log messages, a review of the log files during the problem period is important. The `info.sh`(1M) command only gathers the last 1000 log entries. If many log entries are generated after a problem occurs, a snapshot of the relevant time period may be needed for diagnosis.

For more information on the `info.sh`(1M) command, see the `info.sh`(1M) man page.

Step 6: Contact Support

When you have completed all steps described in this subsection, "What To Do Before You Call", it is time to contact your ASP. Before contacting your ASP, make sure that you have collected all of the information on the Problem Identification Checklist.

The Problem Identification Checklist

Table A-1 is the LSC problem identification checklist. It summarizes the information that you need to gather prior to calling your ASP to report a problem.

Table A-1. LSC Problem Identification Checklist

For software problems, provide the following information:

- Severity Level (see page A-2)
- Company name and customer number (see page A-2)
- Site and configuration information (see page A-3)
- What statement or command are you using?
- What are you expecting to happen, versus what is actually happening?
- What, if any, error messages are you receiving?
- When did you first notice the problem?
- Have you attempted this activity before? Was it successful?
- What has changed since the activity last operated correctly? For instance, was the software upgraded or have you changed the configuration?
- Is the problem reproducible? If so, under what conditions?
- Has the problem occurred before?
- If the problem does not occur consistently, describe the conditions under which the problem does and does not occur.
- What other information can you provide concerning this problem?

For documentation problems, provide the following information:

- Severity level
- Company name and customer number
- Site and configuration information
- The document name, document number, and date of the publication.
- The number of the page that contains the problem.
- A description of the documentation problem. Please be specific.
- Any other information that you can provide concerning this problem.

What LSC Does When Your ASP Reports a Problem

The LSC support center answers problems regarding our released software products. As problems are reported and analyzed, we work to clarify our documentation, fix our software, and implement design requests. The following process describes flow from reporting a problem to generating a fix.

Step 1: Log the Incident

When LSC receives a call or an email, an LSC support analyst assigns it an incident number (for example, `W711030018`). The incident number is to be used for any communications regarding this specific problem. Multiple problems usually receive multiple incident numbers. Be sure that when you are responding to LSC emails that you include the appropriate incident number in the email subject line.

Step 2: Assign a Support Analyst to the Incident

The incident is assigned to one of the support analysts in the LSC support center. This analyst works with your ASP to understand, analyze, and close out the problem. Not all incidents are necessarily answered immediately. If LSC cannot answer a problem right away, LSC tells your ASP why and keeps your ASP posted as analysis progresses.

Step 3: Analyze the Problem Incident

Further analysis of the incident may be required. If so, the support analyst works with the proper LSC staff members to try and resolve the problem.

Step 4: Request Additional Information

During problem resolution, the support analyst may request additional information from you through your ASP. The analyst will refer to the problem using the assigned incident number. You should remember to use this number in all communications.

Step 5: Close Out the Incident Report

When the problem is resolved, LSC closes the incident with a response to you through your ASP. An incident is closed when one of the following occurs:

- A question is answered
- A problem that you encountered is fixed in a release that may require you to upgrade your software
- LSC has analyzed the problem and determined that there is a problem in our software that needs to be fixed. In this case, LSC closes the incident but opens a software problem report as described in the next step.

Step 6: LSC Opens a Software Problem Report

If an incident occurred because of an error in our software, the LSC support analyst opens an internal software problem report. Software problem reports are tracked in an LSC internal database and are assigned to software developers for further analysis and fixing.

Step 7: Take Corrective Action

The software developer works to fix the problem. Although LSC works to close all software problem reports in a timely manner, no guarantees are made as to when these problems might be resolved.

Step 8: Integrate and Test

The software fix is integrated into an internal software release and tested before it is released. Sometimes LSC requests that a customer help in the testing process by running a fix at their site. This is done in situations in which a unique set of circumstances exists for the problem only at the customer site.

Step 9: Identify a Release in Which to Package the Software Fix

Finally, the software fix is integrated into one or more releases of the software product. LSC's software fixes are packaged in releases indicated by the last digit in the release number. For example, SAM-FS 3.5.0-9 is the ninth bugfix edition of 3.5.0.

LSC Support Contacts

If your support contact is directly with LSC, then contact the LSC support center. The following table shows the telephone numbers and email addresses to use for LSC service. Before contacting LSC, please read the "What To Do Before You Call" subsection, which appeared previously in this chapter.

The LSC contact information is as follows:

Communication method	Contact information
Telephone, domestic	1 (800) 650-2337
Telephone, international	+1 (612) 482-5683
Email	`support@lsci.com`
Fax	+1 (651) 554-1540
Regular mail	LSC, Inc. Attention: Software Support 1270 Eagan Industrial Road, Suite 160 Eagan, MN 55121-1231 USA

The LSC support center hours of operations are Monday through Friday, 09:00 - 18:00, central standard time. This does not include LSC holidays.

Glossary

addressable storage

The storage space encompassing online, nearline, and offline storage that is user referenced through an LSC file system.

archiver

The archive program that automatically controls the copying of files to removable cartridges.

archive storage

Copies of file data that have been created on removable cartridges for long-term offline storage.

audit (full)

The process of reading the VSNs from each cartridge in an automated library. For non-tape cartridges, the capacity and space information is determined and entered into the automated library's catalog.

automated library

A robotically controlled device designed to automatically load and unload removable media cartridges without operator intervention. An automated library contains one or more drives and a robot that moves cartridges to and from the storage slots and the drives.

backup storage

A snapshot of a collection of files for the express purpose of preventing inadvertent loss. A backup includes both the file's attributes and associated data.

block allocation map

A bit map representing each available block of storage on a disk and indicating whether the block is in use or free.

cartridge

The physical entity that contains media for recording data. A tape or optical disk. Sometimes referred to as *a piece of media*, *a volume*, or *the medium*.

catalog

A record of the VSNs in an automated library. There is one catalog for each automated library, and at a site, there is one historian for all automated libraries.

data device

For a file system, a device or group of devices upon which file data is stored.

data space

The portion of a collection of files that is the actual data information.

DAU (Disk Allocation Unit)

The basic unit of online storage.

The SAM-FS file system uses several sizes. The small DAU is 4 kilobytes (2^{17} or 4096 bytes). The large DAU is 16, 32, or 64 kilobytes. The available DAU size pairs are 4/16, 4/32, and 4/64.

The QFS and SAM-QFS file systems support a fully adjustable DAU, sized from 16 kilobytes through 65528 kilobytes. The DAU you specify must be multiple of 8 kilobytes.

device logging

A feature that provides device-specific error information used to analyze device problems.

device scanner

Software within the LSC file system that periodically monitors the presence of all manually mounted removable devices and detects the presence of mounted cartridges that may be requested by a user or other process.

devicetool

A SAM-FS and SAM-QFS administrative tool with a graphical user interface for viewing information about and managing individual devices.

direct I/O

An attribute used for large block-aligned sequential I/O. The `setfa`(1) command's `-D` option is the direct I/O option. It sets the direct I/O attribute for a file or directory. If applied to a directory, the direct I/O attribute is inherited.

disk allocation unit

See DAU.

disk cache family set

The definition for the devices that make up a family set. The name of the disk cache family set is found in the equipment identifier field of the Master Configuration File (`mcf` file). This is sometimes referred to as a *metadevice* in industry literature. Also see family set.

disk striping

The process of recording a file across several disks, thereby improving access performance and increasing overall storage capacity.

direct access

A file attribute (stage never) designating that a nearline file can be accessed directly from the archive cartridges and need not be staged for online access.

directory

A file data structure that points to other files and directories within the file system.

disk space thresholds

User-defined disk space thresholds that define the range of desirable disk cache utilization. The high threshold indicates the maximum level of disk cache utilization. The low threshold indicates the minimum level of disk cache utilization. The releaser controls disk cache utilization based on the pre-defined disk space thresholds.

drive

A mechanism for transferring data to and from a volume.

extent array

The array within a file's inode that defines where each data block assigned to the file is located on the disk.

family device set

See family set.

family set

A storage device that is represented by a group of independent physical devices, such as a collection of disks or the drives mounted within an automated library.

Also see disk cache family set.

FDDI

Fiber Distributed Data Interface. FDDI is a 100 megabytes-per-second fiber optic LAN.

file system-specific directives

Directives that follow global directives and begin with `fs =`. File system-specific directives apply until the next `fs =` directive line or until the end of file is encountered. If multiple directives affect a file system, the file system-specific directives override the global directives.

file system

A hierarchical collection of files and directories.

FTP

File Transfer Protocol. An internet protocol for transferring files between two hosts over a TCP/IP network.

global commands

Commands that apply to all file systems and appear before the first "`fs =` " line.

indirect block

A disk block that contains a list of storage blocks. The LSC file systems have up to three levels of indirect blocks. A first-level indirect block contains a list of blocks used for data storage. A second-level indirect block contains a list of first-level indirect blocks.

inode

Index Node. A data structure used by the file system to describe a file. An inode describes all the attributes associated with a file other than the name. The attributes include ownership, access, permission, size, and the file location on the disk system.

inode file

A special file (`.inodes`) on the file system that contains the inode structures for all files resident in the file system. All LSC inodes are 512 bytes long. The inode file is a metadata file, which is separated from file data in the QFS and SAM-QFS file systems.

kernel

The central controlling program that provides basic system facilities. The UNIX kernel creates and manages processes, provides functions to access the file system, provides general security, and supplies communication facilities.

LAN

Local Area Network.

library catalog

See catalog.

LUN

Logical Unit Number.

mcf

Master Configuration File. The file that is read at initialization time that defines the device topology within a QFS, SAM-FS, and SAM-QFS environment.

media

Tape or optical disk cartridges.

media recycling

The process of recycling or reusing archive cartridges with low use (that is, archive cartridges with few active files).

metadata

Data about data. The index information needed to locate the exact data position of a file on a disk. Metadata contains information pertaining to the directory, symbolic link, removable media, segmented file index, and `.inode`s.

metadata device

A separate device (for example a solid-state disk or mirrored device) upon which QFS and SAM-QFS file system metadata is stored. Separating file data from metadata can increase performance. In the `mcf` file, a metadata device is declared as an `mm` device within an `ma` file system.

mirror writing

The process of maintaining two copies of a file on disjoint sets of disks to prevent loss from a single disk failure. It is often referred to as shadowing.

mount point

The path to a directory where a file system is mounted.

name space

The portion of a collection of files that identifies the file, its attributes, and its storage locations.

nearline storage

Removable storage that requires robotic mounting before it can be accessed. Nearline storage is usually less expensive than online storage, but it incurs a somewhat longer access time.

network-attached automated library

A network-attached automated library, such as those from StorageTek, ADIC/Grau, IBM, or Sony, is controlled using a software package supplied by the vendor. The SAM-FS and SAM-QFS file systems interface with the vendor software using an LSC media changer daemon specifically designed for the automated library.

NFS

Network File System. A standard protocol that allows a UNIX file system to be remotely mounted via a network.

offline storage

Storage that requires operator intervention for loading.

offsite storage

Storage that is remote from the server and is used for disaster recovery.

online storage

Storage that is immediately available (for example, disk cache storage).

partition

A portion of a device.

preallocation

The process of reserving a contiguous amount of space on the disk cache for writing a file. This ensures that the space is contiguous. Preallocation can only be performed on zero-sized files. That is, the `setfa -l` command can only be specified for a file that is size zero. For more information, see the `setfa`(1) man page.

prioritizing preview requests

A method of assigning priority to archive and stage requests that cannot be immediately satisfied.

RAID

Redundant Array of Inexpensive/Independent Disks. A disk technology that uses several inexpensive disks to reliably store files. It may protect against data loss from a single disk failure, may provide a fault-tolerant disk environment, and may provide higher throughput than individual disks.

recycler

A SAM-FS and SAM-QFS component that reclaims space on cartridges that is occupied by unused archive copies.

release priority

A method of calculating the release priority of a file within a file system by multiplying various weights by the corresponding file properties and then summing the results.

releaser

A SAM-FS and SAM-QFS component that identifies archived files and releases their disk cache copies, thus making more disk cache space available. The releaser automatically regulates the amount of online disk storage to high and low thresholds.

remote procedure calls

See RPC.

removable media file

A special type of user file that can be accessed directly from where it resides on a removable media cartridge, such as magnetic tape or optical disk cartridge.

robot

The portion of an automated library that moves cartridges between storage slots and drives.

robottool

A SAM-FS and SAM-QFS administrative tool with a graphical user interface (GUI) for viewing and managing automated libraries.

round robin

A data access method in which entire files are written to logical disks in a sequential fashion. When a single file is written to disk, the entire file is written to the first logical disk. The second file is written to the next logical disk, and so on. The size of each file determines the size of the I/O.

By default, LSC file systems implement striped data access unless striped groups are present. Files are round robined if round robin access is specified. If the file system contains mismatched striped groups, striping is not supported and round robin is forced.

Also see glossary entries for striping.

RPC

Remote Procedure Calls. The underlying data exchange mechanism used by NFS to implement custom network data servers.

SAM-FS

The LSC Storage and Archive Manager File System. The SAM-FS software controls the access to all files stored and all devices configured in the Master Configuration File (`mcf`).

SAM-QFS

The SAM-QFS software combines the LSC Storage and Archive Manager with the QFS file system. SAM-QFS offers a high speed, standard UNIX file system interface to users and administrators in conjunction with the storage and archive management utilities. It uses many of the commands available in the SAM-FS command set as well as standard UNIX file system commands.

samfsdump

A program that creates a control structure dump and copies all the control structure information for a given group of files. It is analogous to the UNIX `tar`(1) utility, but it does not copy data.

samfsrestore

A program that restores a control structure dump.

samtool

A SAM-FS and SAM-QFS administrative tool with a GUI for invoking `robottool`, `devicetool`, and `previewtool`.

SCSI

Small Computer System Interface. An electrical communication specification commonly used for peripheral devices such as disk and tape drives and automated libraries.

SCSI-attached Library

An automated library connected directly to a server using the SCSI interface. These libraries are controlled directly by the SAM-FS or SAM-QFS software by using the SCSI standard for automated libraries.

shared writer/shared reader

The QFS shared reader/shared writer capability allows you to specify a file system that can be shared by multiple servers. Multiple hosts can read the file system while only one host can write to the file system. Shared readers are specified with the `-o shared_reader` option on the `mount`(1M) command. The one-writer host is specified with the `-o shared_writer` option on the `mount`(1M) command. For more information on the `mount`(1M) command, see the `mount_samfs`(1M) man page.

small computer system interface

See SCSI.

staging

The process of copying a nearline or offline file from archive storage back to online storage.

storage family set

A set of disks that are collectively represented by a single disk family device.

storage slots

Locations inside an automated library in which cartridges are stored when not being used in a drive. The contents of the storage slots are kept in the automated library's catalog.

stripe size

The number of disk allocation units (DAUs) to allocate before moving to the next device of a stripe. If `stripe=0`, the file system uses round-robin access, not striped access.

striped group

A collection of devices within a QFS or SAM-QFS file system and defined in the `mcf` file as two or more `g`*XXX* devices. Striped groups are treated as one logical device and are always striped with a size equal to the disk allocation unit (DAU). You can specify up to 128 striped groups within a file system.

striping

A data access method in which files are simultaneously written to logical disks in an interlaced fashion.

All LSC file systems allow you to declare either striped or round robin access for each individual file system. The QFS and SAM-QFS file systems allow you to declare striped groups within each file system.

Also see the glossary entry for round robin.

super block

A data structure in the file system that defines the basic parameters of the file system. It is written to all partitions in the storage family set and identifies the partition's membership in the set.

tar

Tape Archive. A standard file/data recording format used by the SAM-FS and SAM-QFS software for archive images.

TCP/IP

Transmission Control Protocol/Internet Protocol. The internet protocols responsible for host-to-host addressing and routing, packet delivery (IP), and reliable delivery of data between application points (TCP).

thresholds

A mechanism for defining the desirable available storage window for online storage. Thresholds set the storage goals for the releaser.

volume

A named area on a cartridge for sharing data. A cartridge has one or more volumes. Double-sided cartridges have two volumes, one on each side.

volume overflow

Allows the system to span a single file over multiple volumes. Volume overflow is useful for sites using very large files that exceed the capacity of their individual cartridges.

VSN

Volume Serial Name. A logical identifier for magnetic tape and optical disk that is written in the volume label.

WORM

Write Once Read Many. A storage classification for media that can be written only once but read many times.

Index