StorageTek®

QFS

Administrator’s Guide

November, 1999

Record of Revision

Version	*Description*
3.4	January 1999. Original printing to support QFS 3.4 release.

About This Guide

Introduction

This guide describes the QFS file system by Storage Technology Corporation (StorageTek). QFS is a high-speed, ASM compatible file system that is used in conjunction with large disks and RAIDs to provide maximum device-speed performance for large file applications.

Organization

This manual is organized as follows:

Chapter	Description
Chapter 1	QFS Overview
Chapter 2	Installing QFS
Chapter 3	Configuring QFS
Chapter 4	QFS Utilities and Operations
Appendix A	Printed QFS manpages

StorageTek License

Conventions

The following conventions are used throughout this document:

Typeface	Meaning	Example
`command`	The fixed-space courier font denotes literal items such as commands, files, routines, path names, and messages	`/etc/fs/samfs/mcf`
`Boldface Courier`	The boldface courier font denotes text you enter at the shell prompt	server# **`sls -D`**
`Italic Courier`	Italics indicate variables in a command line. Replace variables with a real name or value.	# **`mount`** *`mnt_pt`*

StorageTek Publications

If you have comments about the technical accuracy, content, or organization of this document, please tell us. We value your comments and will respond to them promptly. You can contact us in any of the following ways:

- Send us a facsimile with your comments to the attention of “Publications” in Louisville, CO. Our fax number is: 303-661-4904 or 303-661-4905.
- Send us your written comments to:
  StorageTek Solutions Business Group
  Publications Department
  One StorageTek Drive
  Louisville, CO 80028-0076
  USA

To order additional manuals, please send us a written request using one of the methods above.

Chapter 1. QFS Overview

Introduction

QFS from StorageTek is an ASM compatible file system that presents a high speed, standard UNIX file system interface to users and administrators. QFS provides the benefits of raw disk speed coupled with the administrative ease of an enhanced volume manager to provide disk striping or round-robin use of disk volumes. Application programming interfaces allow for pre-allocation of contiguous disk space or access to system-configured stripe groups. QFS can be used as a stand-alone file system or with the Application Storage Manager (ASM). QFS uses many of the commands available in the ASM command set as well commands available with the standard UNIX file system. QFS requires no changes to the UNIX kernel or users' programs.

QFS is designed for applications that use large files that can take advantage of using direct I/O. QFS disk configurations typically integrate multiple disks and RAID devices tuned for high speed, high capacity, and high availability. Example applications that may benefit from the speed provided by QFS include:

- Real time satellite telemetry and data capture
- Structural analysis modeling and visualization
- 3D seismic analysis and geologic reservoir modeling
- Large file caching for video and broadcast

QFS Features

QFS provides the following advantages:

- Very large file support. Files of up to 2^{64}-1 are supported by QFS. Very large files can be striped across many disks or RAID devices, even within a single QFS file system.
- Enhanced volume management. EVM supports both striped and round-robin disk access. Striped disk access allows multiple I/O streams to simultaneously write a file across multiple disks, interlacing the file on logical disks. I/O size is determined by the disk allocation unit. Round-robin disk access 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.
- Separation of file system metadata and file data. File systems use metadata to reference disk blocks on a disk device. Generally, this metadata resides on the same device as the file data. The QFS volume manager allows you to define a separate metadata device in order to reduce device head movement and rotational latency, improve RAID cache utilization, or mirror metadata without mirroring file data.
- Large block allocation method. Disk space is allocated by the system in disk allocation units (DAUs), the basic unit of on-line disk storage. While sectors and blocks describe the physical disk topology, the DAU describes the file system geometry. Using extent-based allocation, QFS supports a DAU of up to 512 1024-byte blocks. Using striped groups, an even larger DAU is possible, up to the size of the RAID devices themselves.
- Supports adjustable DAUs. QFS DAUs are adjustable from 1 to 512 1024-byte blocks. An adjustable DAU is useful for tuning the file system with the physical disk storage device, eliminating the system overhead caused by read-modify-write operations. An application that would benefit from this feature would use large block sequential I/O.
- UNIX direct I/O support. Direct I/O is selectable using the Solaris `directio(3)` function call. Additional performance gains are seen for very large data files because direct I/O does not use the Solaris Virtual Memory manager or buffer cache.
- Pre-allocation of file space. For fast sequential reads and writes, contiguous disk space can be pre-allocated using the `setfa(1)` command.
- Define multiple stripe groups within a file system. In order to support multiple RAID devices in a single file system, stripe groups can be defined. Disk block allocation can be optimized for a stripe group, reducing the overhead for updating the on-disk allocation map. An API is provided allowing users to assign a file to a striped group.

QFS is licensed separately from ASM. When ASM and QFS are used in conjunction with each other, QFS requires ASM 3.3 release or higher.

QFS Terminology

Device—A physical magnetic disk drive or disk partition.

Disk Allocation Unit—The basic unit of on-line disk storage. QFS has an adjustable DAU from 1 to 512 1024-byte blocks. QFS uses a default DAU of 16. ASM uses two DAU sizes: a small DAU (4) and a large DAU (16).

Family Set—A storage device that is represented by a group of independent physical devices, such as a collection of disks.

Disk Cache Family Set—Defines the devices that make up a family set. This name of the disk cache family set is found in the equipment identifier field of the mcf file. This is sometimes called a metadevice in the industry.

Round-robin—A data access method whereby files are wholly written to logical disks in a sequential fashion. As 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 written determines the size of the I/O. Note that standard ASM disk cache family sets (equipment type **ms**) uses round-robin devices (defined as equipment type **md**) only.

Striping—A data access method whereby files are simultaneously written across multiple logical disks in an interlaced fashion. Striped devices are a collection of devices (defined with the disk cache family set) that belong to an **ma** disk cache family set. Striping is available only with QFS.

Stripe size—The number of DAUs to allocate before moving to the next device of a stripe. The default stripe is 0; a stripe of 0 specifies round-robin devices, not striped devices.

Striped group—A collection of devices defined in the mcf file belonging to an **ma** disk cache family set. Striped groups allow the splitting of parts of a file on several different devices.

Metadata—The index information needed to locate the exact data position of a file on a disk. Metadata contains important information such as the file's location, size, etc.

Metadata device—A separate device (for instance, a solid-state disk or mirrored device) on which QFS metadata is stored. A separation of file data and metadata can be important to increase performance.

Data device—A device or group of devices on which file data is stored.

QFS Enhanced Volume Manager

The QFS Enhanced Volume Manager (EVM) uses the Solaris physical device drivers to pass I/O requests to and from the underlying devices. EVM is similar to the standard ASM volume manager in that physical devices are grouped together in a family set on which a QFS file system is mounted.

Family sets are defined using two device types: metadata devices and striped or round-robin devices. A third device type, the *striped group*, is available to configure many RAID devices into a device with a very large allocation size. All of these device types are defined using the equipment type field in the master configuration file, `/etc/fs/samfs/mcf`. A sample mcf file is shown below:

```
# QFS file system configuration example
#
# Equipment       Eq    Eq   Fam.  Dev.    Additional
# Identifier      Ord  Type  Set   State   Parameters
#-----------      ---   --   ------ ------  -----------------
-samfs1            10  ma  samfs1
/dev/dsk/c0t1d0s6  11  mm  samfs1  -   /dev/rdsk/c0t1d0s6
/dev/dsk/c1t1d0s2  12  mr  samfs1  -   /dev/rdsk/c1t1d0s2
/dev/dsk/c1t2d0s2  13  mr  samfs1  -   /dev/rdsk/c1t2d0s2
/dev/dsk/c1t3d0s2  14  mr  samfs1  -   /dev/rdsk/c1t3d0s2
/dev/dsk/c1t4d0s2  15  mr  samfs1  -   /dev/rdsk/c1t4d0s2
/dev/dsk/c1t5d0s2  16  mr  samfs1  -   /dev/rdsk/c1t5d0s2
```

Device	**Equipment Identifier**	**Description**
`Metadata Device`	mm	This device is used for metadata only. No file data is written to the device. You can use multiple metadata device specifications.
`Striped or Round-Robin Devices`	mr	Data is striped across these devices. The stripe width can be specified in `/etc/vfstab`. Note that default stripe width is 0, indicating that QFS will use allocate round-robin allocation for devices.
`Striped Groups`	**g***XX*	Allocates file data to a *striped group*. A striped group allows the ability to stripe a single file across a group of RAID devices, in either a striped or round-robin manner. Groups are named g0-g127. There is only 1 DAU per striped group. This saves bit map space.

The metadata device (**mm**) is used in conjunction with a data device to achieve device speeds on the data disk. Metadata, (inodes, directories, and such) on typical file systems are located along with the file data on the same disk. This causes the system to constantly seek in order to write short records. QFS separates the metadata and writes it to a separate disk device, thus eliminating this bottleneck. Accesses to these metadata devices are short; therefore you may want to use a low rotational latency disk or even a solid-state disk device, which has zero latency. Note that performance may suffer if you use a large I/O device, such as a RAID, as your metadata device. It is also possible to mirror the metadata devices.

File data resides on separate devices that can be striped or round-robined. Striped or round-robin device types are specified using the **mr** equipment identifier. Specifying multiple mr devices within a family set allows the file system to appear to access a single device. This is sometimes called a metadevice in other volume management systems.

Round-Robin Allocation

A file system using round-robin devices 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, we start over again with the first device. Figure 1-1 depicts a file system using five round-robin devices. File 1 is written to disk 1, File 2 is written to disk 2, and File 3 is written to disk 3, and so on. When File 6 is created, it is written to disk 1, starting the round-robin allocation scheme over again.

In the case of large files that exceed the size of the physical device, the first portion of the file is written to the first device, then the remainder of the file is written on the next device.

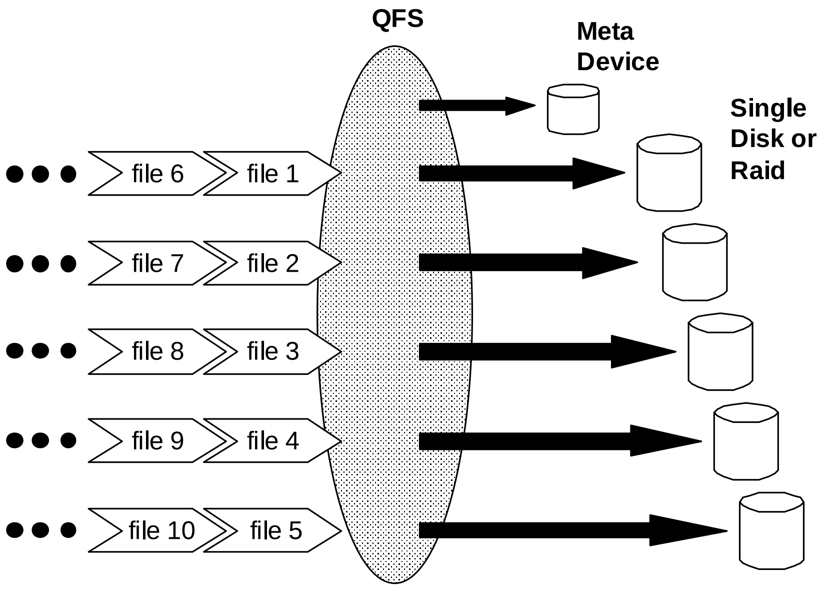

Figure 1-1 . A Round-Robin QFS File System with Five Devices

Note that round-robin devices are the default with QFS. If you do not specify a stripe width for a file or file system, the devices use round-robin allocation. Round-robin can be explicitly set as a mount point option in `/etc/vfstab using the mount option stripe=0.`

Striping

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 devices generally increases performance because disk reads/writes are spread concurrently across disk heads. Figure 1-2 shows an example of a striped file system.

In this striping example, 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 indicates the size of the writes to each disk. The DAU size is from 1 –512 1024-byte blocks. Figure 1-2 depicts a striped file system with a DAU size of 256.

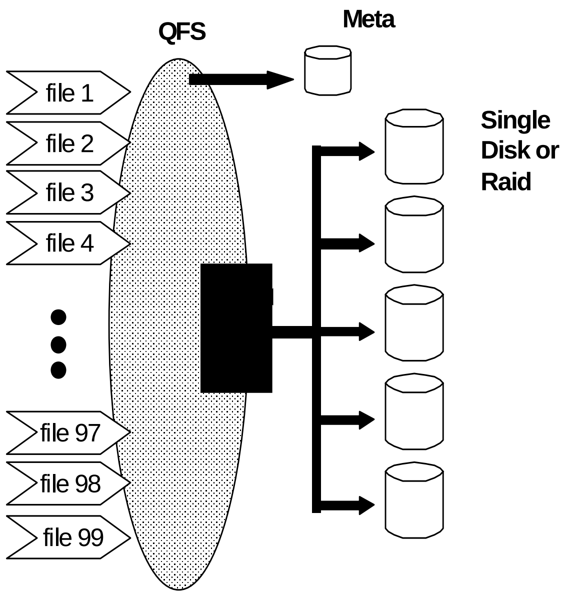

Figure 1-2   A Striped QFS File System with Five Devices

Striped Groups

A *striped group* is a special QFS configuration 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 file system. Striped groups save bit map space and system update time for very large RAID configurations.

Figure 1-3 shows an example using striped groups. In this example, files written to the samfs1 file system are round-robined between groups g0, g1, and g2. Note that you cannot change these groups without remaking the file system with `sammkfs(1M).` Three striped groups are defined (g0, g1, and g2); each group is comprised of two physical RAID devices. The mount point option in `/etc/vfstab is set to stripe=0`. A typical mcf file is shown.

```
samfs1             10  ma  samfs1
/dev/dsk/c0t1d0s6  11  mm  samfs1  -   /dev/rdsk/c0t1d0s6
/dev/dsk/c1t1d0s2  12  g0  samfs1  -   /dev/rdsk/c1t1d0s2
/dev/dsk/c1t2d0s2  13  g0  samfs1  -   /dev/rdsk/c1t2d0s2
/dev/dsk/c1t3d0s2  14  g1  samfs1  -   /dev/rdsk/c1t3d0s2
/dev/dsk/c1t4d0s2  15  g1  samfs1  -   /dev/rdsk/c1t4d0s2
/dev/dsk/c1t3d0s2  16  g2  samfs1  -   /dev/rdsk/c1t5d0s2
/dev/dsk/c1t4d0s2  17  g2  samfs1  -   /dev/rdsk/c1t6d0s2
```

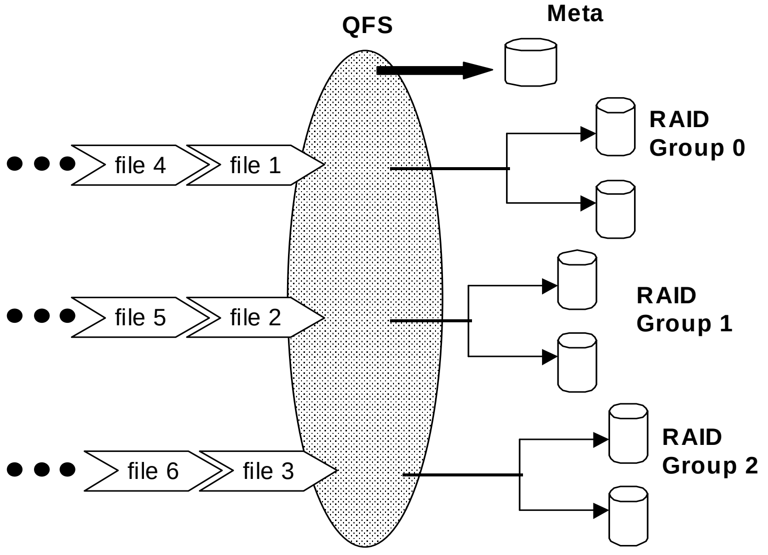

Figure 1-3 QFS Using Striped Groups – Round-Robin Method

Figure 1-4 shows an example of striped groups in which the data is striped across groups. In this example, files written to the qfs1 file system are sriped throughout groups g0, g1, and g2. Again, note that you cannot change these groups without remaking the file system with `sammkfs(1M).` Three striped groups are defined (g0, g1, and g2); each group is comprised of two physical RAID devices. The mount point option in `/etc/vfstab` is set to `stripe=1` or greater. A typical mcf file is shown.

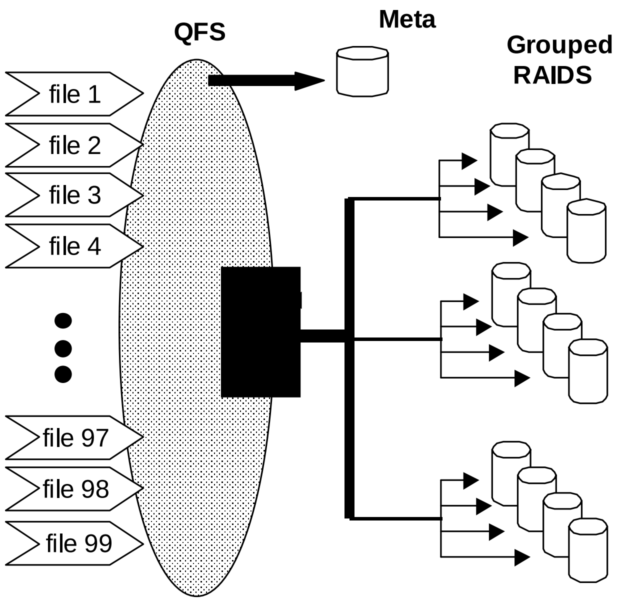

Figure 1-4   QFS Using Striped Groups – Stripe Method

QFS and ASM System Administration

The system administrator is responsible for all QFS administration. If you are using QFS in conjunction with ASM, it is important that the administrator be knowledgeable and have experience in the following:

- ASM installation and configuration
- ASM system administration
- ASM operations
- ASM troubleshooting and support

For complete information on these topics, see the “ASM System Administration Guide,” publication 311248302.

QFS Commands and Utilities

Many of the QFS-specific functions are accessed using options to the existing ASM commands. For a complete description of the QFS commands and options, see Appendix B, “Manpages”.

The following is a listing of commands and options available with QFS:

- `sammkfs(1M) – Constructs a new QFS file system. Use the –a` option to set the number of 1024-byte blocks to be allocated to a disk allocation unit (DAU).
- `mount_samfs(1M)` – Additional mount options for QFS:

  `stripe` – Set the stripe size for the file system to a number of DAUs.

  `noforcedirectio` – Use buffered (paged) I/O as the I/O mode.

  `forcedirectio` – Use direct I/O as the I/O mode.

  `qwrite` – Enables simultaneous reads and writes to the same file. This is useful for some applications such as databases
- `samfsinfo(1M)` – Displays information on a QFS or ASM file system.
- `setfa(1)` – Sets QFS attributes on a file or directory. This user command can be called from user applications for pre-allocating file space, specifying a stripe width for a file, or specifying the number of a striped group during pre-allocation. You can also set QFS attributes using the `sam_setfa(3)` call.
- `samgrowfs(1M)` – Expands a QFS file system by allowing the addition of logical disks.
- `samfsck(1M)` – Checks and optionally repairs a QFS file system.

- `qfsdump/qfsrestore(1M)` – Dumps and restores a QFS file system.
- `info.sh(1M)` – Creates a QFS diagnostic report
- `sam_trace(1M)` – Dump the QFS trace buffer
- `samncheck(1M)` – Generate pathnames vs. i-numbers for QFS file systems
- `setsyscall(1M)` – Change system call number in ASM library
- `sfind(1)` – Find files in a QFS file system
- `sls(1)` – List files in a QFS file system
- `sam_advise(3)` - Set attributes on a file
- `sam_initrpc(3x), sam_closerpc(3x)` - Perform RPC initialization and closing for the ASM RPC API library

Chapter 2. QFS Installation

Introduction

This section describes how to install QFS on a SPARC server running the Sun Solaris operating system.

If you will be using QFS in conjunction with ASM-QFS, you should refer to the “ASM System Administrator Guide,” publication 311248301.

The following topics are covered in this chapter:

- The QFS release package
- Directories and files
- Installation overview
- ASM system preparation
- QFS installation procedure
- QFS software upgrade procedure

QFS Release Package

QFS is available from authorized StorageTek resellers and directly from StorageTek. Software is generally available via anonymous ftp server, but is also available on CD-ROM for a nominal fee.

The QFS software package (LSCqfs) is released in Solaris `pkgadd(1M)` format. This package must reflect the Solaris version (2.5, 2.5.1, or 2.6) for the platform on which you will run ASM. See the instructions provided with the software for a complete description of the directories and files provided with the release package.

QFS releases are identified using alphanumeric characters and are ranked *M.F.B-P*, where the following characters represent the release level identification.

`M`	Major release identification
`F`	Feature addition identification
`B`	Bug fix upgrade identification
`P`	Patch level identification. 1 through 99 indicates a patch release.

The following examples illustrate the release identification convention. Note that the base release of a first feature release of a major release may not contain a patch level.

`QFS 3.4.0`	Base release of the first feature release of a major release
`QFS 3.4.0-5`	Patch level of a base release

Directories and Files

The following tables list the directories and files installed with QFS.

Directory	Description
`/etc/fs/samfs`	Configuration files and daemon binaries.
`/opt/LSCsamfs/bin`	User command binaries.
`/opt/LSCsamfs/examples`	Various example configuration files.
`/opt/LSCsamfs/include`	API include files.
`/opt/LSCsamfs/lib`	Relocatable libraries.
`/opt/LSCsamfs/man`	Man pages.
`/opt/LSCsamfs/sbin`	System administrator command binaries.

File	Description
`/etc/fs/samfs/LICENSE.3.3`	License file. For more information, see step 7 in the “Software Installation Procedure” section in this chapter.
`/etc/fs/samfs/mcf`	Master Configuration File. See `mcf(4).`
`/kernel/fs/samfs`	File system module.
`/kernel/sys/samsys`	System call module.

Additional information concerning QFS files can be obtained from the on-line man pages after they have been installed later in this procedure. A printed version of the manual pages can be found in “QFS Manual Pages”, Appendix C.

Modified System Files

The following table lists the system files that are modified during the installation of QFS.

File	Description
`/etc/name_to_major`	Major number information file.
`/etc/name_to_sysnum`	System call information file.

QFS System Verification

The QFS design supports large file and large I/O requests. As such, it is up to you to ensure that both the hardware and software on your system is capable of handling such requests. This subsection outlines the system requirements to be met before installing QFS.

Your system must meet the following requirements before proceeding with the ASM installation.

1. SPARC server
2. Direct-attached RAIDs or disks
3. Solaris 2.5, 2.5.1, or 2.6 operating system and patches
4. QFS 3.3.0 or greater release
5. QFS license key

QFS is released as a separate package from StorageTek and is available via anonymous ftp from your local authorized service provider or from StorageTek. If you willing be running QFS in conjunction with ASM, you will need a new license key that enable QFS on your server.

All of the steps in this section assume that you are a super-user logged in as root.

Requirement 1: SPARC Server

In order to handle the large I/O requests inherent with QFS, StorageTek recommends the following Sun Microsystems, Inc. SPARC-based processor systems:

- Sun Ultra 1 and above.
- Sun Enterprise 1000 and above.

Although not officially supported by StorageTek, QFS runs on SPARC clones, as well.

The server is to be installed and running prior to installing QFS.

Requirement 2: Verify RAID or Disk Array

The QFS file system is created on a RAID disk device or other disk storage subsystem. A minimum of one disk device or partition is required. Of course, multiple devices can increase the performance of QFS I/O. Examples of RAID devices used with QFS include:

- Photons
- Sonoma
- MAXSTRAT Gen5 family of storage servers
- StorageTek OPENstorage disk subsystems

The RAID device must be attached using a direct or fiber channel SCSI connection to the server. Individual disk partitions or the entire disk may be specified using the QFS enhanced volume manager. Of course, QFS supports RAIDs and disks controlled by third-party volume management software as well.

Use the `format(1M)` command to see the disks attached to your system. The following example shows three disks attached to a server, one internal disk connected via controller 0 on the first target (c0t1d0) and two external disks connect via controller 1 on targets 1 and 2 `(c1t1d0 and c1t2d0)`.

```
 server> format

0. c0t1d0 <SUN1.05 cyl 2036 alt 2 hd 14 sec 72>
/iommu@f,e0000000/sbus@f,e0001000/espdma@f,400000/esp@f,8
00000/sd@1,0

1. c1t1d0 <SEAGATE-ST424-0116 cyl 2604 alt 2 hd 19 sec 84>
/iommu@f,e0000000/sbus@f,e0001000/dma@2,81000/esp@2,80000
/sd@1,0

2. c1t2d0 <SEAGATE-ST424-0119 cyl 2604 alt 2 hd 19 sec 84>
/iommu@f,e0000000/sbus@f,e0001000/dma@2,81000/esp@2,80000
/sd@2,0
```

Requirement 3: Solaris Operating System and Patches

QFS relies on a properly configured Solaris 2.5, 2.5.1, or 2.6 operating system. Check to see that your server is running one of these levels of Solaris by entering the following:

```
server> uname -sr
SunOS 5.6
```

SunOS levels 5.*x* levels correspond to Solaris 2.*x* levels. The above system is running Solaris 2.6.

It is required to have the latest SunSoft recommended patches for the Solaris operating system. An up-to-date list of Solaris patches required by ASM is included with the ASM software in a file named, “README”. This list is accessible via the LSC World Wide Web site (URL http://www.lsci.com/) on the “Services” page.

The Solaris 2.6 patches needed at the time this manual was printed are:

- kernel jumbo patch (105181-09, or later). Note that this patch level has the Sun priority paging kernel. This feature has shown significant performance increases in memory utilization of high memory desktop and server systems. The following line must be included in the /etc/system configuration file in order to turn priority paging on:

  priority_paging=1

- libthread.so.1 patches (105568-08, or later).

To determine which patches are installed on your system, enter the following:

```
server# showrev -p | more
```

If the patches listed above are not installed, you will need to do so prior to installing ASM. Patches are provided to Sun maintenance contract customers via CD-ROM, anonymous ftp, and the Sun World Wide Web page (URL http://sunsolve.com/). LSC, Inc. does not redistribute Sun patches.

To install a patch, mount the CD-ROM or transfer the patch software on to your system. Follow the instructions outlined in the Patch Installation Instructions and Special Install Instructions of the README file.

Requirement 4: ASM 3.3 Release (optional)

If you will be running QFS with an ASM system, verify that the server to be be running QFS has the ASM 3.3 release or higher. ASM is installed in Solaris `pkgadd(1M)` format. The package name is `LSCsamfs.` Enter the following command on the server to check for the proper release version:

```
server# pkginfo -l LSCsamfs
```

```
ADD OUTPUT EXAMPLE HERE
```

If you are not running ASM 3.3 or higher, you must upgrade ASM. See the ASM System Administration Guide, publication 311248301 for upgrade procedures.

Requirement 5: QFS License

You must have a license key that supports QFS. The license is required for all QFS operations. Temporary licenses are available upon request.

If you do not have a valid QFS license key, contact your authorized service provider or LSC, Inc. You will be required to provide the following information:

- Company purchase order number
- Company name, address, phone, contact
- Server Host ID where ASM with QFS is being installed. To display the Host ID on the system use the `hostid(1)` command.
- Type of server on which to install QFS, for instance, a Sun Enterprise 10000 server.
- Optionally, if you will be running QFS with ASM, we need to know what type of media library you are using. For each media library, we need the following:
    1. The vendor name/model of the library and whether or not the library is direct or network-attached.
    2. The type of media used in this library.
    3. The number of slots for the media library/media type.
- Specify that you will be using QFS.

The license key allows the system to run indefinitely unless it is a temporary license. When a QFS temporary license expires, you will no longer be able to mount QFS file systems.

Once you have your license keys, place them, starting in column one, in `/etc/fs/samfs/LICENSE.3.3.` No other keywords, host ids, etc. may appear. The license becomes effective when the QFS file system is mounted for the first time.

QFS Installation

This section describes the step-by-step procedure for installing and configuring QFS for the first time. Upgrading QFS software on an existing server is described in the “QFS Software Upgrade Procedure” section later in this chapter. For most of the procedures in this section, you must be logged in as root.

Step 1: Add the SAM_FS Group (QFS 3.3 only)

A group named “SAM_FS” is *required* for QFS 3.3 releases. This requirement has been removed in the QFS 3.4 release.

The `SAM_FS` group name is in uppercase letters separated by an underscore character. The Group ID (GID) must be unique among all groups on the server. There should be no members in this group.

To create the SAM_FS group, edit the group file. The following example shows an `/etc/group entry for SAM_FS`.

```
SAM_FS:*:47:
```

Step 2: Copy QFS Installation File to Server

The QFS software is available via ftp server and CD-ROM.

Note: We reserve the right to update our installation procedures at any time. For this reason, the QFS software is accompanied by a README file that contains the most current installation instructions. These instructions provide up-to-date details on how to install QFS release files on the server.

Copy the QFS software on to your server in a permanent directory location. The software is located in a single file named “samqfs”. The following example creates a directory and copies samqfs in to this directory:

```
server> mkdir /usr/tmp/qfs

server> cp samqfs /usr/tmp/qfs
```

Step 3: Add the Packages

QFS 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. `pkgadd(1M)` prompts you to confirm various actions necessary to install the QFS package.

Install the QFS package using the `pkgadd command as follows:`

```
server# pkgadd -d /usr/tmp/qfs/samqfs
```

The following is a sample output from a QFS `pkgadd`. Responses to the interactive session are shown in bold.

# **pkgadd -d /tmp/samqfs**

```
The following packages are available:
  1  LSCqfs     Quick File System Solaris 2.6
                (SUNW,SPARCstation-10) 3.4.0

Select package(s) you wish to process (or 'all' to process
all packages). (default: all) [?,??,q]: all

Processing package instance <LSCqfs> from </tmp/samqfs>

Quick File System Solaris 2.6
(SUNW,SPARCstation-10) 3.4.0
LSC Incorporated


-----------------------------------------------------
```

```
DO YOU ACCEPT the terms of the LSC License Agreement
(YES,NO,VIEW) ?
```
YES
```
Previous samsys syscall # was: 180

The following line will be added to the /etc/name_to_sysnum
file:
          samsys 180

The original /etc/name_to_sysnum file will be saved as:
          /etc/name_to_sysnum.LSC
## Processing package information.
## Processing system information.
   7 package pathnames are already properly installed.
## Verifying disk space requirements.
## Checking for conflicts with packages already installed.
## Checking for setuid/setgid programs.

This package contains scripts which will be executed with
super-user permission during the process of installing this
package.

Do you want to continue with the installation of <LSCqfs>
[y,n,?]y

Installing Quick File System Solaris 2.6 as <LSCqfs>
## Installing part 1 of 1.
/etc/fs/samfs/mount
/etc/fs/samfs/nl_messages.cat
/etc/init.d/samfs_mt
/etc/rc3.d/K05samfs <symbolic link>
/etc/rc3.d/S05samfs <symbolic link>
/kernel/fs/samfs
/kernel/sys/samsys
/opt/LSCsamfs/bin/setfa
/opt/LSCsamfs/bin/sfind
/opt/LSCsamfs/bin/sls
/opt/LSCsamfs/client/include/samrpc.h <symbolic link>
/opt/LSCsamfs/client/lib/libsamrpc.a
/opt/LSCsamfs/client/lib/libsamrpc.so
/opt/LSCsamfs/client/src/client/Makefile
/opt/LSCsamfs/client/src/client/sam_attrtoa.c
/opt/LSCsamfs/client/src/client/samfs_clnt.c
/opt/LSCsamfs/client/src/client/samrpc.c
/opt/LSCsamfs/client/src/xdr/Makefile
/opt/LSCsamfs/client/src/xdr/sam_xdr.c
/opt/LSCsamfs/examples/mcf
/opt/LSCsamfs/include/samrpc.h
/opt/LSCsamfs/include/stat.h
/opt/LSCsamfs/include/version.h
/opt/LSCsamfs/lib/libsam.so
/opt/LSCsamfs/lib/libsamapi.so
/opt/LSCsamfs/lib/libsamfs.so
/opt/LSCsamfs/lib/libsamut.so
/opt/LSCsamfs/man/man1/setfa.1
/opt/LSCsamfs/man/man1/sfind.1
/opt/LSCsamfs/man/man1/sls.1
```

```
/opt/LSCsamfs/man/man1m/mount_samfs.1m
/opt/LSCsamfs/man/man1m/qfsdump.1m
/opt/LSCsamfs/man/man1m/qfsrestore.1m
/opt/LSCsamfs/man/man1m/sam_trace.1m
/opt/LSCsamfs/man/man1m/samcmd.1m
/opt/LSCsamfs/man/man1m/samfsck.1m
/opt/LSCsamfs/man/man1m/samgrowfs.1m
/opt/LSCsamfs/man/man1m/sammkfs.1m
/opt/LSCsamfs/man/man1m/samncheck.1m
/opt/LSCsamfs/man/man1m/setsyscall.1m
/opt/LSCsamfs/man/man3/sam_advise.3
/opt/LSCsamfs/man/man3/sam_setfa.3
/opt/LSCsamfs/man/man3x/sam_closerpc.3x
/opt/LSCsamfs/man/man3x/sam_initrpc.3x
/opt/LSCsamfs/man/man3x/sam_setfa.3x
/opt/LSCsamfs/man/man4/mcf.4
/opt/LSCsamfs/sbin/info.sh
/opt/LSCsamfs/sbin/qfsrestore
/opt/LSCsamfs/sbin/sam_trace
/opt/LSCsamfs/sbin/samfsck
/opt/LSCsamfs/sbin/sammkfs
/opt/LSCsamfs/sbin/samncheck
/opt/LSCsamfs/sbin/setsyscall
[ verifying class <none> ]
/opt/LSCsamfs/sbin/qfsdump <linked pathname>
/opt/LSCsamfs/sbin/samfsinfo <linked pathname>
/opt/LSCsamfs/sbin/samgrowfs <linked pathname>
## Executing postinstall script.
Samfs system call changed to 180 in
/opt/LSCsamfs/lib/libsam.so
Samfs system call 180 was already in /etc/name_to_sysnum

Installation of <LSCqfs> was successful.
```

Step 4: Set Up PATH and MANPATH

Set up PATH statements:

- For users who will be running the QFS user commands (for instance, `setfa(1)),` add `/opt/LSCsamfs/bin` to the user's PATH variable.
- For users who will be running the administrator commands, add `/opt/LSCsamfs/sbin` to the user's PATH variable.
- To gain access to the man pages, add `/opt/LSCsamfs/man` to the MANPATH variable.

Step 5: License QFS

QFS uses encrypted license keys. License keys are required to run QFS and any associated products from LSC, Inc. Keys are issued as described in requirement 5 in the “QFS System Preparation” section.

The license keys are encoded alphanumeric strings. You will receive one or more of these strings depending on the system configuration and the products being licensed. Place the keys, starting in column one, on the first line in `/etc/fs/samfs/LICENSE.3.3.` Each license key must be on a separate line ending with a newline and all keys must start in column one. No other keywords, host ids, comments, etc. may appear in the `LICENSE.3.3` file. The license becomes effective the next time QFS is mounted.

The license keys allow the system to run indefinitely unless you were issued a temporary license. When a temporary license expires, the system will no longer be able to mount QFS file systems.

Step 6: Configure ASM

Each QFS storage server configuration is unique. System requirements vary as well as the actual hardware used at each site. Therefore, you should understand that this section of the system administration guide presents sample configurations of a general nature. It is up to the system administrator at your site, using input from your local hardware engineer, to set up the specific configuration for your QFS server. Specific examples showing round-robin, striped, and striped group configurations are presented in Chapter 3, "QFS Configuration Examples".

To configure QFS file systems, create a master configuration file `/etc/fs/samfs/mcf.` The file contains information needed by QFS to identify and organize disk devices in to QFS file systems. A sample mcf file (shown following) is located in `/opt/LSCsamfs/examples/mcf.`

```
#
# QFS file system configuration
#
# Equipment       Eq   Eq    Fam.  Dev.    Additional
# Identifier      Ord  Type  Set   State   Parameters
#-----------      ---   --  ------ ------  ------------------
qfs1                1   ma   qfs1
/dev/dsk/c0t0d0s0  11   mm   qfs1    on    /dev/rdsk/c1t0d0s0
/dev/dsk/c1t1d0s2  12   mr   qfs1    on    /dev/rdsk/c1t1d0s2
/dev/dsk/c1t2d0s2  13   mr   qfs1    on    /dev/rdsk/c1t2d0s2
/dev/dsk/c1t3d0s2  14   mr   qfs1    on    /dev/rdsk/c1t3d0s2
```

Each line of the mcf file has the following format:

```
equipment     equipment    equipment    family   device    additional
identifier    ordinal      type         set      state     parameters
```

Delimit the field in each line with space or tabs. Comment lines start with # symbol. Use a dash (–) to indicate omitted fields. The following table describes the fields. Refer to the online manual page `mcf(4)` for more information.

Field	Description
equipment identifier	Required. This field is either the name of the file system or the `/dev/dsk` entry to a disk partition or slice.
equipment ordinal	Required. Enter a unique integer from 1 to 32757.
equipment type	Required. Enter a two- or three-character code for the device type. QFS use the following equipment types: ma Defines a QFS file system device mm Meta-data device. mr Round-robin or striped data device. g*X* Striped group data device. Striped groups start with the letter g followed by a number 1-xx, i.e., g12. See the `mcf(4) man page for specific equipment types.`
family set	Required. The family set associates all devices with the same family set name together as a QFS file system.
device state	Optional. Enter a state for the device. Valid device states are “on” and “off”.
additional parameters	Required. For each /dev/dsk/ equipment identifier there must be a corresponding /dev/rdsk identifier in this field.

The remainder of this subsection is an example showing the various steps and considerations involved in setting up the `mcf` file on a server. There is an example `mcf` file in `/opt/LSCsamfs/examples/mcf.`

Example QFS Hardware Configuration

The example server has a StorageTek Clarion RAID device with four StorageTek OPENstorage 9153 disk drives. Each drive has 34 GB storage.

The Solaris `format(1M)` command reports that the disks are partitioned as follows:

```
   # format
   Searching for disks...done
AVAILABLE DISK SELECTIONS:
0. c0t0d0 <SUN4.2G cyl 3880 alt 2 hd 16 sec 135>
          /sbus@1f,0/SUNW,fas@e,8800000/sd@0,0
1. c0t1d0 <SEAGATE-ST39140WC-1206 cyl 9004 alt 2 hd 8 sec 246>
          /sbus@1f,0/SUNW,fas@e,8800000/sd@1,0
2. c2t4d0 <STK-OPENstorage9153-0205 cyl 17338 alt 2 hd 64 sec 64>
          /pseudo/rdnexus@2/rdriver@4,0
3. c2t4d1 <STK-OPENstorage9153-0205 cyl 17338 alt 2 hd 64 sec 64>
          /pseudo/rdnexus@2/rdriver@4,1
4. c2t4d2 <STK-OPENstorage9153-0205 cyl 34977 alt 2 hd 64 sec 64>
          /pseudo/rdnexus@2/rdriver@4,2
5. c2t4d3 <STK-OPENstorage9153-0205 cyl 34977 alt 2 hd 64 sec 64>
          /pseudo/rdnexus@2/rdriver@4,3
6. c3t2d0 <SEAGATE-ST15230W-0168 cyl 3974 alt 2 hd 19 sec 111>
          /sbus@1f,0/QLGC,isp@2,10000/sd@2,0
```

One file system named `qfs1` will created on disks c2t4d0, c2t4d1, c2t4d2, and c2t4d3. Each disk is partitioned identically with slice 0 consuming the entire disk. The following is an example partition map for these disks.

Part	Tag	Flag	Cylinders	Size	Blocks
0	usr	wm	0-17377	33.86GB	`(17337/0/0)` 71012352
1	unassigned	wm	0	0	(0/0/0)
2	backup	wu	0-17377	33.86GB	`(17337/0/0)` 71012352
3	unassigned	wm	0	0	(0/0/0)
4	unassigned	wm	0	0	(0/0/0)
5	unassigned	wm	0	0	(0/0/0)
6	unassigned	wm	0	0	(0/0/0)
7	unassigned	wm	0	0	(0/0/0)

Figure 2-1 shows the file system entries in the `mcf` file. Line 1 defines the QFS file system. The name of this file system (`qfs1`) will be used later when writing the `/etc/vfstab` entry for the file system and when making the file system. Line 2 shows an mm device type entry for the meta-data device. Note that this entry is *not* part of the RAID device described above. A separate disk is used for caching inode information leaving the RAID for high-speed data accesses. Lines 3-5 then are the data devices using mr device type.

Note: Be sure that the `/dev/dsk` and `/dev/rdsk` names on each line reference the same c*nt*n*d*n*s*n partition.

```
# QFS file system configuration example
#
# Equipment       Eq    Eq   Fam.  Dev.    Additional
# Identifier      Ord  Type  Set   State   Parameters
#-----------      ---   --  ------ ------  ------------------
qfs1               10   ma  qfs1
/dev/dsk/c0t1d0s2  11   mm  qfs1     on    /dev/rdsk/c0t1d0s2
/dev/dsk/c2t4d0s0  12   mr  qfs1     on    /dev/rdsk/c2t4d0s0
/dev/dsk/c2t4d1s0  13   mr  qfs1     on    /dev/rdsk/c2t4d1s0
/dev/dsk/c2t4d2s0  14   mr  qfs1     on    /dev/rdsk/c2t4d2s0
/dev/dsk/c2t4d3s0  15   mr  qfs1     on    /dev/rdsk/c2t4d3s0
```

Figure 2-1 Example QFS mcf File

Caution: As with creating any type of file system, if you give the wrong partition names, you risk damaging user or system data. Be sure to specify disk partitions which are otherwise unused on your system. Do not use overlapping partitions.

Step 8: Create the Mount Point

This document assumes `/qfs` is the mount point of the `qfs1` file system, but you may pick a different name if you wish. If you do, substitute that name for `/qfs`.

Create the mount point:

```
server# mkdir /qfs
server# chmod 555 /qfs
```

Make the `/etc/vfstab` entry for each QFS file system. An example entry follows:

```
qfs1  -  /qfs  samfs  -  no  stripe=1
```

The first field (device to mount) specifies the name of the QFS file system to mount. This must be the same name as specified in the mcf file file system entry. The second file (device to fsck) contains a dash to show no options; do not use `fsck(1M)` on a QFS file system. The third field (mount point) is the default mount point. The fourth field (FS type) must be samfs. The fifth field (fsck pass) is unused and contains a dash to show no options.

The sixth field (mount at boot) specifies delay`,` a special flag which is interpreted by a script which is installed into the /etc/rc3.d directory by the pkgadd command. This script, which is run when the system enters init state 3, scans the /etc/vfstab file and mounts QFS file systems which are flagged delay. See the `mount_samfs (1M)` man page for the format of these entries. Specifying no indicates that you do not wish to automatically mount the file system. Note that you cannot use yes as an option.

Finally, the seventh field (mount options) is a list of comma-separated options (with no spaces) that are used in mounting the file system`.` See `mount_samfs(1M)` for a list of the available mount options. This example uses stripe=1, indicating a stripe width of one DAU.

Note: If you configured multiple mount points, repeat these steps for each mount point, using a different mount point `(/qfs)` and family set name `(qfs1)` each time.

Step 9: Make and Mount the File System

Using `sammkfs,` create a file system for each defined family set.

```
server# /opt/LSCsamfs/sbin/sammkfs qfs1
total data kilobytes       = 143265792
total data kilobytes free  = 143265760
total meta kilobytes       = 35506176
total meta kilobytes free  = 35504688
```

The file system(s) are now ready to be mounted:

```
server# mount /qfs
```

Note: The first time a QFS file system is mounted after a boot, QFS software will be loaded into the kernel and the QFS daemons will be started. This will take a few seconds.

If you wish to change the permissions, owner or group owner of the root directory of the file system, do so now:

```
server# chmod 755 /qfs
server# chown root /qfs
server# chgrp other /qfs
```

Step 10: Share the File System with Client Machines

The Solaris `share(1M)` command must be run to make the file system available for mounting by remote systems. Share commands are typically placed in the `/etc/dfs/dfstab` file and are executed automatically by Solaris when entering `init` state 3.

For example, on the server, enter the line:

```
server# share -F nfs -o rw=client1:client2 -d “QFS" /qfs
```

Note: Making the above entry in `/etc/dfs/dfstab` will cause Solaris to share the file system after the next system reboot. Should you wish to share the file system immediately, you must type the `share` command at a root shell prompt. If no file systems were shared when Solaris booted, the NFS server was not started. You must reboot after adding the first share entry to this file.

Step 11: Mount the File System on the Client Machines

On the client systems, arrange for the server's `/sam` file system to be mounted at a convenient mount point. Here, we will mount *server*:`/sam` on /`sam`. For example, on the client, enter the following line in `/etc/vfstab`:

```
   server:/qfs - /qfs    nfs - no hard,intr,timeo=1000
```

Then, on the command line, perform the `mount`:

```
   client# mount /qfs
```

This can also be done by using the automounter, if the site wishes. Follow your site procedures for adding *server*:`/qfs to your automounter maps.`

Note: It is strongly recommended that clients mount the file system with the `hard` option. At times, there may be a significant delay in ASM's response to client requests (for example, when a requested file resides on a piece of media which must be loaded into a DLT tape drive). If the `hard` option is not given, the client may return an error instead of retrying the operation until it completes.

If you must use the `soft` option, you need to set the value of `retrans` value to a large number, such as 120 (the default is 5). This sets the number of NFS retransmissions.

There are several NFS parameters that can impact the performance of a NFS mounted QFS file system. These NFS mount parameters are set using options in the `/etc/vfstab` file (see `mount_nfs(1M)`). Check the following parameters and set them accordingly:

`timeo =` *`n`* - This value set the NFS timeout to *`n`* tenths of a second. The default is 11 tenths of a second. LSC recommends setting this number to something much larger, such as 1000 or even 10000.

`rsize =` *`n`* - This value sets the read buffer size to *`n`* bytes. In NFS 2, the default value is 8192 bytes. In NFS 3, the default value is 32768. It should be fine in NFS 3. For NFS 2 users, set it to 32768.

`wsize =` *`n`* `- This value sets the write buffer size to` *`n`* bytes. In NFS 2, the default is 8192 bytes. In NFS 3, the default value is 32768. The value should be fine for NFS 3. For NFS 2 users, set it to 32768.

Software Upgrade Procedure

This section describes upgrading a server to newer QFS release. All steps in this section must be performed as superuser (root).

Step 1: Back Up Each QFS File System

If you do not have current backup files for your QFS file systems, create them now using `qfsdump .` See Chapter 4, "QFS Utilities and Operations" for more information.

Step 2: Copy QFS Installation File to Server

The QFS software is available via ftp server and CD-ROM.

Note: We reserve the right to update our installation procedures at any time. For this reason, the QFS software is accompanied by a README file that contains the most current installation instructions. These instructions provide up-to-date details on how to install QFS release files on the server.

Copy the QFS software on to your server in a permanent directory location. The software is located in a single file named "samqfs". The following example creates a directory and copies samqfs in to this directory.

```
server> mkdir /usr/tmp/qfs
server> cp samqfs /usr/tmp/qfs
```

Step 3: Unmount the File Systems

Using the Solaris `umount` command, unmount each QFS file system. If you encounter difficulty unmounting a file system, it may be because you or some other user is using files or has changed into directories into the file system. If you cannot identify and correct this situation, you may need to reboot with the mount at boot field in `/etc/vfstab` changed from `delay` to `no`. This inhibits the file systems from being mounted at reboot time. You can use the "`fuser` *`qfs1`*" command to show the process IDs of processes using the *qfs1* file system.

Step 4: Remove Existing QFS Software

Use `pkgrm` to remove the existing QFS software. You must remove all existing QFS packages before installing the new packages. The following example removes the current QFS package:

```
server# pkgrm LSCqfs
```

Step 5: Unload ASM Modules

In order to avoid a reboot, you need to unload the QFS modules. The following example checks to see if the modules are loaded, and in this case, unloads them:

```
   server# modinfo | grep sam
    91 f5ef4000 3904c 17 1  samfs (Storage and Archiving Mgmt FS)
    92 f5c3c800 3cc  180 1  samsys (SAMFS system)
   server# modunload -i 91
   server# modunload -i 92
```

If you cannot unload the modules, you will need to reboot the system.

Step 6: Add the Packages

Run `pkgadd(1M) to re-install the QFS package(s).  Run the pkgadd` command to install all packages answering yes to each of the questions:

```
   server# pkgadd -d samqfs
```

`pkgadd(1M)` will prompt you to confirm various actions necessary to install the package. You will also be asked to view the license agreement as part of the installation. Respond in the affirmative to each of these prompts.

Step 7: Mount the File System(s)

Mount the file systems and continue operation with the upgraded QFS software.

Chapter 3. Configuring QFS

Introduction

This chapter describes how to configure QFS.

Each QFS file server configuration is unique. System requirements vary as well as the actual hardware used at each site. This section of the QFS Administrator's Guide presents sample configurations of a general nature. It is up to the system administrator at your site, using input from your local hardware engineer, to set up the specific configuration for your QFS file server.

How to Configure QFS

This section describes how to:

- Create an `/etc/fs/samfs/mcf` file for your system disk configuration.
- Edit the `/etc/vfstab` file
- Construct a new QFS file system using `sammkfs(1M).`
- Mount the file system.

To configure QFS file systems, create a master configuration file `/etc/fs/samfs/mcf.` The file contains information needed by QFS to identify and organize RAID and disk devices in to QFS file systems. A sample mcf file (shown following) is located in `/opt/LSCsamfs/examples/mcf.`

```
#
# QFS file system configuration
#
# Equipment        Eq   Eq    Fam.  Dev.    Additional
# Identifier       Ord  Type  Set   State   Parameters
#-----------      ---   --  ------ ------  ------------------
qfs1                 1   ma   qfs1
/dev/dsk/c0t0d0s0  11   mm   qfs1    on    /dev/rdsk/c1t0d0s0
/dev/dsk/c1t1d0s2  12   mr   qfs1    on    /dev/rdsk/c1t1d0s2
/dev/dsk/c1t2d0s2  13   mr   qfs1    on    /dev/rdsk/c1t2d0s2
/dev/dsk/c1t3d0s2  14   mr   qfs1    on    /dev/rdsk/c1t3d0s2
```

Each line of the mcf file has the following format:

```
equipment   equipment   equipment   family  device   additional
identifier  ordinal     t type      set     state    parameters
```

Delimit the field in each line with space or tabs. Comment lines start with `#` symbol. Use a dash (–) to indicate omitted fields. The following table describes the fields. Refer to the online manual page `mcf(4)` for more information.

Field	Description
equipment identifier	Required. This field is either the name of the file system or the `/dev/dsk` entry to a disk partition or slice.
equipment ordinal	Required. Enter a unique integer from 1 to 32757.
equipment type	Required. Enter a two- or three-character code for the device type. QFS use the following equipment types: ma Defines a QFS file system device mm Meta-data device. mr Round-robin or striped data device. g*X* Striped group data device. Striped groups start with the letter g followed by a number 1-xx, i.e., g12. See the `mcf(4) man page for specific equipment types.`
family set	Required. The family set associates all devices with the same family set name together as a QFS file system.
device state	Optional. Enter a state for the device. Valid device states are “on” and “off”.
additional parameters	Required. For each /dev/dsk/ equipment identifier there must be a corresponding /dev/rdsk identifier in this field.

The rest of this section presents sample QFS configurations, showing various steps and considerations in setting up the mcf file on a server as follows:

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

Note that all sample QFS configurations could have ASM 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 ASM System Administrator’s Guide, publication 311248302.

The sample configurations assume that QFS is loaded on the system and all QFS file systems are unmounted. All of the samples follow these basic steps:

1. Write the `/etc/fs/samfs/mcf` file.
2. Modify the `/etc/vfstab file`, if required.
3. Run `sammkfs(1M)` to make QFS file systems.

Round-Robin Disk Example

This sample configuration illustrates an QFS file system that separates the metadata on to a low-latency disk. Round-robin allocation is used on four disk drives. The file system is created using `sammkfs`.

The following assumptions are used:

- Metadata device – A single partition (s6) is used on controller 0, LUN 0.
- Data devices – Four disks are attached to controller 1. Each disk is on a separate LUN (1-4). The entire disk is used for data storage, indicated by partition 2 (s2).

Step 1: Write the mcf File

Figure 3-1 shows 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/c0t0d0s0  11   mm   qfs1    on    /dev/rdsk/c0t0d0s0
/dev/dsk/c1t1d0s0  12   mr   qfs1    on    /dev/rdsk/c1t1d0s0
/dev/dsk/c1t2d0s5  13   mr   qfs1    on    /dev/rdsk/c1t2d0s5
/dev/dsk/c1t3d0s0  14   mr   qfs1    on    /dev/rdsk/c1t3d0s0
/dev/dsk/c1t4d0s1  15   mr   qfs1    on    /dev/rdsk/c1t4d0s1
```

Figure 3-1   Round-robin mcf File

Step 2: Modify the /etc/vfstab File

The `/etc/vfstab` is set as usual for an ASM file system. Since the default QFS behavior is round-robin, no stripe width is necessary.

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

```
qfs1    -    /qfs    samfs    -    no   stripe=0
```

Step 3: Run sammkfs

Initialize the QFS file system using `sammkfs(1M).` The following example sets the DAU size to 64 blocks (remember that a QFS block is 1 kbyte or 1024 bytes). The default DAU is 16k:

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

Stripe Disk Configuration

This sample configuration illustrates an QFS file system that again separates the metadata on to a low-latency disk. File data is striped to four disk drives. The file system is created using `sammkfs(1M),` where the DAU size is specified.

The following assumptions are used:

- Metadata device – A single partition (s6) is used on controller 0, LUN 0.
- Data devices – Four disks are attached to controller 1. Each disk is on a separate LUN (1-4). All partitions are used on the entire disk drive (s2).

Step 1. Write mcf file.

Write the `/etc/fs/samfs/mcf` file using the disk configuration assumptions. Figure 3-2 shows 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/c1t1d0s2  12   mr  qfs1     on    /dev/rdsk/c1t1d0s2
/dev/dsk/c1t2d0s2  13   mr  qfs1     on    /dev/rdsk/c1t2d0s2
/dev/dsk/c1t3d0s2  14   mr  qfs1     on    /dev/rdsk/c1t3d0s2
/dev/dsk/c1t4d0s2  15   mr  qfs1     on    /dev/rdsk/c1t4d0s2
```

Figure 3-2 Stripe Disk mcf File

Step 2: Modify the /etc/vfstab File

Set the stripe width using the stripe option. This sample sets the stripe width equal to one disk allocation unit (DAU).

```
qfs1    -    /qfs    samfs    -    no   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 below).

Step 3: Run sammkfs

Initialize the QFS file system using sammkfs. The following examples sets the DAU size to 128 blocks (remember that a QFS block is 1 kbyte or 1024 bytes):

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

With this striped disk configuration, any file written to this file system would be striped across all of the devices in increments of 128k. Files less than the aggregate stripe width times the number of devices (in this example, files less than 128k * 4 disks = 512k) will still use 512k of disk space. Files larger than 512k will be have space allocated as needed in total space increments of 512k.

Metadata is written to device 10 only.

Striped Groups Configuration

Striped groups allow you to group RAID devices together for very large files. Normally, a disk allocation unit is represented by one bit in the bit maps. With striped groups, however, there is only one DAU per striped group. 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: A DAU is the minimum disk space allocated. The minimum disk space allocated in a striped group is as follows:

allocation_unit * number of disks in the group

Writing a single byte of data will fill the entire striped group.

The use of striped groups is for very specific applications. Make sure that you understand the impacts of using striped groups with your file system.

The devices within a group must of the same size. It is not possible to grow a striped group. You can add additional striped groups, however.

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

The following assumptions are used:

- Metadata device—A single partition (s6) is used on controller 0, LUN 0.
- Data devices—Four disks (two groups of two identical disks) are attached to controller 1. Each disk is on a separate LUN (1-4). All partitions are used on the entire disk drive (s2).

Step 1. Write mcf file.

Write the `/etc/fs/samfs/mcf` file using the disk configuration assumptions. Figure 3-3 shows 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/c1t1d0s2  12   g0  qfs1     on    /dev/rdsk/c1t1d0s2
/dev/dsk/c1t2d0s2  13   g0  qfs1     on    /dev/rdsk/c1t2d0s2
/dev/dsk/c0t3d0s2  14   g1  qfs1     on    /dev/rdsk/c0t3d0s2
/dev/dsk/c0t4d0s2  15   g1  qfs1     on    /dev/rdsk/c0t4d0s2
```

Figure 3-3 Striped Groups mcf File

Step 2: Modify the /etc/vfstab File

Set the stripe width using the stripe option. This sample sets the stripe width equal to zero, meaning use a round-robin allocation from striped group g0 to striped group g1:

```
qfs1    -    /qfs    samfs    -    no   stripe=0
```

Step 3: Run sammkfs

Initialize the QFS file system using sammkfs. The –a option is not used with striped groups, as the DAU is equal to the size of an allocation or the size of each group.

```
# sammkfs qfs1
```

In this example, there are two striped groups, g0 and g1, with `stripe=0` in `/etc/vfstab,` this means equipment 12 and 13 will be striped, equipment 14 and 15 will be striped, and files will round-robin around the 2 striped groups. You are really treating a striped group as a bound entity.

You cannot change these groups without a `sammkfs(1M).`

Chapter 4. QFS Utilities and Operations

Introduction

This chapter describes application interfaces to QFS. Sections include:

- Setting file attributes such as preallocation, allocation method, and stripe width using the `setfa(1)` command
- Using the Solaris `directio(3) function call`
- Data alignment between file system and RAID
- Striping the inodes file
- Using the `qfsdump/qfsrestore(1M)` utilities
- Increasing QFS file system space with samgrowfs
- Using shared reader/writer with QFS

Setting File Attributes Using setfa(1M)

QFS provides performance features that can enabled by applications on a per-file basis. This section describes how the application programmer can use these features for preallocating file space, specifying the allocation method for the file, and to specify the stripe width if using disk striping.

File attributes are set using the `setfa(1)` command. `setfa` 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` with a directory, files and directories created within that directory will 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.

Selecting File Allocation Method and Stripe Width

By default, a file created in a QFS file system uses the allocation method and stripe width specified at mount (see the mount_samfs(1M) manpage). However, a user may wish to use a different allocating for a file or directory of files using the `setfa –s` (stripe) command.

The allocation method can be either a round-robin or striped. The –s *stripe* 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.
1-512	Striped	1 – 512 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 specifying a round-robin allocation method. The command also is preallocating 100 Mbytes of space for a file called /qfs/100MB.rrobin:

```
$ setfa –s 0 –l 1000000 /qfs/100MB.rrobin
```

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

```
$ setfa –s 64 /qfs/file.stripe
```

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, avoiding a file system full condition, and that this space is allocated sequentially as defined by the filesystem. 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 –l` (the letter "el") command and specifying the length of the file in bytes. For example, to preallocate a 1 Gbyte file named /qfs/file_alloc, enter the following:

```
$ setfa –l 1000000000 /qfs/file_alloc
```

Using Direct I/O

ASM supports direct I/O in addition to buffered I/O (paging cache). If `directio(3)` is specified, this means data is transferred directly between the user's buffer and disk. Direct I/O means much less time is spent in the system.

QFS Read/Write Locks

By default, QFS disables 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.

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 QFS `qwrite` mount option bypasses the file system locking 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 will improve I/O performance by queuing multiple requests at the drive level.

The following example of using qwrite on a database file system:

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

Striped Groups - Allocation Units

When you initialize an QFS file system using `sammkfs(1M)`, the size of the disk allocation unit (DAU) can be specified in Kbyte units (1024 bytes). If you do not specify the size of the DAU, the default size is 16. Note that a power of 2 is *not* required.

If the file system contains striped groups, a DAU is *allocation* * the number of members in the striped group. This, of course, makes for a very large DAU when using striped groups.

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:

allocation_unit = RAID stripe width * 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 64k, then the optimal allocation unit is 64 * 7 = 448. The format command can be used to determine the number of data disks: divide the capacity by the size of the disks.

Striping the Inodes File

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

The stripe specification is taken from the stripe parameter on the mount command. Thus, if stripe=0, we would stay on one meta device until it is full, then switch to the next one.

Suppose you would like to stripe the meta data, but round-robin the file data. This can be accomplished using setfa(1) on the inodes file, as follows:

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

QFS Dumps and Restores

This section describes the procedures for dumping and restoring QFS file system data and control structures. The following topics are covered in this section.

- An overview of ASM control structures
- How to dump and restore file systems using `qfsdump`/`qfsrestore`

Note: The following `qfsdump/qfsrestore` restrictions should be noted. These restrictions should be removed in future revisions of QFS.

- `qfsdump` only supports full dumps of specified files and directories. Incremental dump is not available.
- `qfsdump` dumps all data of a sparse file, and `qfsrestore` will restore 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.

An Overview of QFS Control Structures

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

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 meta device. All data is stored in files 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 is dependent 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.

QFS File System Design

It is important to design your file system so that you can quickly access information as well as recover information when needed.

QFS is a multi-threaded advanced storage management system. To take advantage of these capabilities, you should create multiple file systems whenever possible.

In QFS file systems, like Solaris file systems, directory lookups use a linear search method, searching from the beginning of the file system to the end. As the number of files in a directory increases, the search time through the directory increases. Users with directories over 1000 files will see excessive search times.

These search times will also be noticed 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 1000.

Making Backups for QFS File Systems

QFS uses the `qfsdump` and `qfsrestore` utilities for backing up file systems.

`qfsdump`/`qfsrestore` creates and restores control structure and data dumps of a current directory. `qsdump` saves the relative path information for each file contained in a complete file system or in a portion of a file system. You can also use these utilities for restoring a single directory or file. `qfsdump`s are performed on mounted file systems.

StorageTek recommends performing full `qfsdump` 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 `qfsdump` 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 perform `qfsdump`s in order 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`.

How to Dump QFS File Systems

Use the `qfsdump(1M)` command to dump file systems; the `qfsrestore(1M)` command is used to restore file systems generated by `qfsdump`. See the manual page for a complete description of the syntax and options available with these commands.

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. Change to the mount point for the file system or to the directory.

```
server# cd /qfs1
```

3. Create a dump file by executing the `qfsdump` command.

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

To automate your dump procedures, use `cron(1)` to create an entry in root's cron table. The following example entry performs a dump and manages the files within a dump directory. Note that the xargs argument is "el one" not "one one":

```
10 0 * * * (find /qfsdump.directory -type f -mtime +3 /
-print| xargs -l1 rm -f); cd /qfs1; /
/opt/LSCsamfs/sbin/qfsdump -f /
/qfsdump.directory/qfs1/‘date +\%y\%m\%d‘)
```

Replace `/qfsdump.directory` with an existing directory of your choice. This entry will cause the commands to be executed each day at midnight. First, any dump file older than three days will be removed and a new dump will be created in `/qfsdump.directory/qfs1/`*`yymmdd`*.

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 `dump_file` as illustrated in the previous subsection.

1. Change into a QFS file system and create a new directory.

```
server# cd /qfs1

server# mkdir restored_qfs

server# cd restored_qfs
```

2. Restore the directory using an existing dump file.

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

How to Restore Single Files and Directories

You can also use a `qfsdump` file 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 Add Disks to a QFS File System

At some point, you may want to add disk drives or RAID devices in order to increase QFS file system. This is accomplished by updating the `mcf` file and using `samgrowfs(1M)` as described in this section. There is no need to re-make or restore the file system. Note that when adding a disks.

To add disks to an existing QFS file system follow this sequence of steps:

1. Unmount all QFS file systems.

2. Edit the `/etc/fs/samfs/mcf` file. Add the new disks *after* the existing disks for the file system. Save the changes and quit the editor.

   If the equipment identifier name in `/etc/fs/samfs/mcf` has changed the file systems will no longer be able to mounted. Instead, the message:

Warning: ASM superblock equipment identifier <*id*>s on eq <*eq*> does not match <*id*> in mcf will be logged in `/var/adm/messages`.

3. Run `samgrowfs(1M)` on the file system (named *`qfs1`* in this example) that is growing:

   # **samgrowfs *qfs1***

4. Start QFS and mount the file systems.

Appendix A. QFS Manpages

Introduction

This appendix contains a printed version of the online manual pages provided with QFS.

Appendix B. QFS Messages

This appendix contains a list of known messages specific to QFS.

```
35-neptune# sammkfs -a 48 -i 3000 -V samfs1
```

License

```
License does not support ’ma’ file system: sammkfs
aborts
You tried to run sammkfs with HPFS options on a file
system containing “ma” device types.
The license on this system does not support HPFS.
```

Message only

```
47-neptune# !35
sammkfs -a 48 -i 3000 -V samfs1
name:     samfs1
time:     Mon Aug 10 17:10:38 1998
count:    3
capacity: 0194fa60
space:    0194fa60
ord  eq   capacity      space   device
  0  11   0086fe20   0086fe20   /dev/dsk/c1t1d0s2
  1  12   0086fe20   0086fe20   /dev/dsk/c1t2d0s2
  2  21   0086fe20   0086fe20   /dev/dsk/c1t5d0s2
48-neptune#
```

```
49-neptune# sammkfs -a 48 -i 3000 samfs1
total data kilobytes       = 26540640
total data kilobytes free  = 26538768
50-neptune#
```

samu "l" screen

```
License Information                   samu
3.3.0-8 Mon Aug 10 17:28:00
License expires Fri Aug 14 17:08:49 1998
hostid = 7232855a
expiration date = Fri Aug 14 17:08:49 1998
Remote sam server feature enabled
Remote sam client feature enabled
Migration toolkit feature enabled
Fast file system feature enabled
```

Samu "m" screen

```
Mass storage status                     samu       3.3.0-8 Mon Aug 10 17:28:31
License expires Fri Aug 14 17:08:49 1998

ty    eq  status      use state ord   capacity       free mcntg  part high low
ma    10  m---------   1% on           25.311G    25.292G     8    16  60% 30%
 mr   11  ----------   1% on      0     8.437G     8.430G
 mr   13  ----------   1% on      1     8.437G     8.431G
 mr   14  ----------   1% on      2     8.437G     8.431G
```

messages show problems during the pkgadd.

```
Quick File System Solaris 2.6
(SUNW,SPARCstation-10) 3.4.0
LSC Incorporated
FATAL:  LSCsamfs is already installed on this system.
The Quick File System is already part of LSCsamfs and
is controlled by appropriate license keys.
To install the stand-alone LSCqfs package, you must
first remove LSCsamfs.
pkgadd: ERROR: request script did not complete
successfully
Installation of <LSCqfs> failed.
No changes were made to the system.
13-ROOT:
```