Fedora 14

Storage Administration Guide

Deploying and configuring single-node storage in Fedora

Josef Bacik
Kamil Dudka
Hans de Goede
Doug Ledford
Daniel Novotny
Nathan Straz
David Wysochanski
Michael Christie
Sachin Prabhu
Rob Evers
David Howells
David Lehman
Jeff Moyer
Eric Sandeen
Mike Snitzer
This guide provides instructions on how to effectively manage storage devices and file systems on Fedora 14 and later. It is intended for use by system administrators with basic to intermediate knowledge of Red Hat Enterprise Linux or Fedora.
Preface ix
  1. Document Conventions ................................................................. ix
     1.1. Typographic Conventions ...................................................... ix
     1.2. Pull-quote Conventions ......................................................... x
     1.3. Notes and Warnings ............................................................. xi
  2. We Need Feedback! ...................................................................... xi

1. Overview 1
  1.1. What's New in Fedora 14 ........................................................ 1

2. Storage Considerations During Installation 3
  2.1. Updates to Storage Configuration During Installation .................... 3
  2.2. Overview of Supported File Systems ........................................... 3
  2.3. Special Considerations ............................................................ 4

3. LVM (Logical Volume Manager) 7
  3.1. What is LVM? ........................................................................... 8
  3.2. Using system-config-lvm .......................................................... 8
     3.2.1. Utilizing Uninitialized Entities .......................................... 11
     3.2.2. Adding Unallocated Volumes to a Volume Group ............... 11
     3.2.3. Migrating Extents ............................................................. 13
     3.2.4. Adding a New Hard Disk Using LVM ................................ 15
     3.2.5. Adding a New Volume Group ........................................... 16
     3.2.6. Extending a Volume Group ............................................... 17
     3.2.7. Editing a Logical Volume .................................................. 18
     3.3. References ........................................................................... 21

4. Partitions 23
  4.1. Viewing the Partition Table ........................................................ 24
  4.2. Creating a Partition .................................................................. 25
     4.2.1. Making the Partition ........................................................ 25
     4.2.2. Formatting and Labeling the Partition .................................. 26
     4.2.3. Add to /etc/fstab .............................................................. 26
  4.3. Removing a Partition .................................................................. 27
  4.4. Resizing a Partition .................................................................. 27

5. File System Structure 29
  5.1. Why Share a Common Structure? .............................................. 29
  5.2. Overview of File System Hierarchy Standard (FHS) ....................... 29
     5.2.1. FHS Organization ............................................................. 29
  5.3. Special Fedora File Locations .................................................... 36
  5.4. The /proc Virtual File System ................................................... 37

6. The Ext3 File System 39
  6.1. Creating an Ext3 File System ...................................................... 40
  6.2. Converting to an Ext3 File System .............................................. 40
  6.3. Reverting to an Ext2 File System .............................................. 40

7. The Ext4 File System 43
  7.1. Creating an Ext4 File System ...................................................... 44
  7.2. Converting an Ext3 File System to Ext4 ..................................... 44
  7.3. Mounting an Ext4 File System ................................................... 45
  7.4. Resizing an Ext4 File System ..................................................... 46
  7.5. Other Ext4 File System Utilities ................................................ 46

8. The XFS File System 49
  8.1. Creating an XFS File System ..................................................... 49
  8.2. Mounting an XFS File System ................................................... 50
Preface

1. Document Conventions
This manual uses several conventions to highlight certain words and phrases and draw attention to specific pieces of information.

In PDF and paper editions, this manual uses typefaces drawn from the Liberation Fonts¹ set. The Liberation Fonts set is also used in HTML editions if the set is installed on your system. If not, alternative but equivalent typefaces are displayed. Note: Red Hat Enterprise Linux 5 and later includes the Liberation Fonts set by default.

1.1. Typographic Conventions
Four typographic conventions are used to call attention to specific words and phrases. These conventions, and the circumstances they apply to, are as follows.

Mono-spaced Bold
Used to highlight system input, including shell commands, file names and paths. Also used to highlight keycaps and key combinations. For example:

To see the contents of the file my_next_bestselling_novel in your current working directory, enter the cat my_next_bestselling_novel command at the shell prompt and press Enter to execute the command.

The above includes a file name, a shell command and a keycap, all presented in mono-spaced bold and all distinguishable thanks to context.

Key combinations can be distinguished from keycaps by the hyphen connecting each part of a key combination. For example:

Press Enter to execute the command.

Press Ctrl+Alt+F2 to switch to the first virtual terminal. Press Ctrl+Alt+F1 to return to your X-Windows session.

The first paragraph highlights the particular keycap to press. The second highlights two key combinations (each a set of three keycaps with each set pressed simultaneously).

If source code is discussed, class names, methods, functions, variable names and returned values mentioned within a paragraph will be presented as above, in mono-spaced bold. For example:

File-related classes include filesystem for file systems, file for files, and dir for directories. Each class has its own associated set of permissions.

Proportional Bold
This denotes words or phrases encountered on a system, including application names; dialog box text; labeled buttons; check-box and radio button labels; menu titles and sub-menu titles. For example:

Choose System → Preferences → Mouse from the main menu bar to launch Mouse Preferences. In the Buttons tab, click the Left-handed mouse check box and click

¹ https://fedorahosted.org/liberation-fonts/
Close to switch the primary mouse button from the left to the right (making the mouse suitable for use in the left hand).

To insert a special character into a gedit file, choose Applications → Accessories → Character Map from the main menu bar. Next, choose Search → Find... from the Character Map menu bar, type the name of the character in the Search field and click Next. The character you sought will be highlighted in the Character Table. Double-click this highlighted character to place it in the Text to copy field and then click the Copy button. Now switch back to your document and choose Edit → Paste from the gedit menu bar.

The above text includes application names; system-wide menu names and items; application-specific menu names; and buttons and text found within a GUI interface, all presented in proportional bold and all distinguishable by context.

Mono-spaced Bold Italic or Proportional Bold Italic

Whether mono-spaced bold or proportional bold, the addition of italics indicates replaceable or variable text. Italics denotes text you do not input literally or displayed text that changes depending on circumstance. For example:

To connect to a remote machine using ssh, type ssh username@domain.name at a shell prompt. If the remote machine is example.com and your username on that machine is john, type ssh john@example.com.

The mount -o remount file-system command remounts the named file system. For example, to remount the /home file system, the command is mount -o remount /home.

To see the version of a currently installed package, use the rpm -q package command. It will return a result as follows: package-version-release. Note the words in bold italics above — username, domain.name, file-system, package, version and release. Each word is a placeholder, either for text you enter when issuing a command or for text displayed by the system.

Aside from standard usage for presenting the title of a work, italics denotes the first use of a new and important term. For example:

Publican is a DocBook publishing system.

1.2. Pull-quote Conventions
Terminal output and source code listings are set off visually from the surrounding text.

Output sent to a terminal is set in mono-spaced roman and presented thus:

books        Desktop   documentation  drafts  mss    photos   stuff  svn
books_tests  Desktop1  downloads      images  notes  scripts  svgs

Source-code listings are also set in mono-spaced roman but add syntax highlighting as follows:

package org.jboss.book.jca.ex1;
import javax.naming.InitialContext;
1.3. Notes and Warnings

Finally, we use three visual styles to draw attention to information that might otherwise be overlooked.

### Note

Notes are tips, shortcuts or alternative approaches to the task at hand. Ignoring a note should have no negative consequences, but you might miss out on a trick that makes your life easier.

### Important

Important boxes detail things that are easily missed: configuration changes that only apply to the current session, or services that need restarting before an update will apply. Ignoring a box labeled 'Important' will not cause data loss but may cause irritation and frustration.

### Warning

Warnings should not be ignored. Ignoring warnings will most likely cause data loss.

2. We Need Feedback!

If you find a typographical error in this manual, or if you have thought of a way to make this manual better, we would love to hear from you! Please submit a report in Bugzilla: [http://bugzilla.redhat.com/bugzilla/](http://bugzilla.redhat.com/bugzilla/) against the product Fedora Documentation.

When submitting a bug report, be sure to mention the manual's identifier: *storage-administration-guide*

If you have a suggestion for improving the documentation, try to be as specific as possible when describing it. If you have found an error, please include the section number and some of the surrounding text so we can find it easily.
Overview

The Storage Administration Guide contains extensive information on supported file systems and data storage features in Fedora 14. This book is intended as a quick reference for administrators managing single-node (i.e. non-clustered) storage solutions.

1.1. What’s New in Fedora 14

This release of Fedora features several improvements in file system support and storage device management. Support for the following file systems have now been added:

- ext4
- GFS2
- XFS

Fedora 14 also features the following file system enhancements:

File System Encryption

You can now encrypt a file system at mount using eCryptfs, which provides an encryption layer on top of an actual file system. This “pseudo-file system” allows per-file and file name encryption, which offers more granular encryption than encrypted block devices. For more information about file system encryption, refer to Chapter 11, Encrypted File System.

File System Caching

FS-Cache allows you to use local storage for caching data from file systems served over the network (e.g. through NFS). This helps minimize network traffic, although it does not guarantee faster access to data over the network. FS-Cache allows a file system on a server to interact directly with a client's local cache without creating an overmounted file system. For more information about FS-Cache, refer to Chapter 10, FS-Cache.

I/O Limit Processing

The Linux I/O stack can now process I/O limit information for devices that provide it. This allows storage management tools to better optimize I/O for some devices. For more information on this, refer to Chapter 17, Storage I/O Alignment and Size.

ext4 Support

The ext4 file system is fully supported in this release. It is now the default file system as of Fedora 13, supporting an unlimited number of subdirectories. It also features more granular timestamping, extended attributes support, and quota journalling. For more information on ext4, refer to Chapter 7, The Ext4 File System.

Network Block Storage

Fibre-channel over ethernet is now supported. This allows a fibre-channel interface to use 10-Gigabit ethernet networks while preserving the fibre-channel protocol. For instructions on how to set this up, refer to Section 20.7, “Configuring a Fibre-Channel Over Ethernet Interface”.

Storage Considerations During Installation

Many storage device and file system settings can only be configured at install time. Other settings, such as file system type, can only be modified up to a certain point without requiring a reformat. As such, it is prudent that you plan your storage configuration accordingly before installing Fedora 14.

This chapter discusses several considerations when planning a storage configuration for your system. For actual installation instructions (including storage configuration during installation), refer to the Fedora 14 Installation Guide.

2.1. Updates to Storage Configuration During Installation

Installation configuration for the following settings/devices has been updated for Fedora 14:

**Fibre-Channel over Ethernet (FCoE)**

Anaconda can now configure FCoE storage devices during installation.

**Storage Device Filtering Interface**

Anaconda now has improved control over which storage devices are used during installation. You can now control which devices are available/visible to the installer, in addition to which devices are actually used for system storage. There are two paths through device filtering:

- **Basic Path**
  - For systems that only use locally attached disks and firmware RAID arrays as storage devices

- **Advanced Path**
  - For systems that use SAN (e.g. multipath, iSCSI, FCoE) devices

**Auto-partitioning and /home**

Auto-partitioning now creates a separate logical volume for the /home file system when 50GB or more is available for allocation of LVM physical volumes. The root file system (/) will be limited to a maximum of 50GB when creating a separate /home logical volume, but the /home logical volume will grow to occupy all remaining space in the volume group.

2.2. Overview of Supported File Systems

This section shows basic technical information on each file system supported by Fedora 14.

<table>
<thead>
<tr>
<th>File System</th>
<th>Max Supported Size</th>
<th>Max File Size</th>
<th>Max Subdirectories (per directory)</th>
<th>Max Depth of Symbolic Links</th>
<th>ACL Support</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ext2</td>
<td>8TB</td>
<td>2TB</td>
<td>32,000</td>
<td>8</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>Ext3</td>
<td>16TB</td>
<td>2TB</td>
<td>32,000</td>
<td>8</td>
<td>Yes</td>
<td>Chapter 6, The Ext3 File System</td>
</tr>
</tbody>
</table>
### Chapter 2. Storage Considerations During Installation

<table>
<thead>
<tr>
<th>File System</th>
<th>Max Supported Size</th>
<th>Max File Size</th>
<th>Max Subdirectories (per directory)</th>
<th>Max Depth of Symbolic Links</th>
<th>ACL Support</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ext4</td>
<td>16TB</td>
<td>16TB</td>
<td>65,000¹</td>
<td>8</td>
<td>Yes</td>
<td>Chapter 7, The Ext4 File System</td>
</tr>
<tr>
<td>XFS</td>
<td>100TB</td>
<td>16TB</td>
<td>65,000¹</td>
<td>8</td>
<td>Yes</td>
<td>Chapter 8, The XFS File System</td>
</tr>
</tbody>
</table>

¹ When the link count exceeds 65,000, it is reset to 1 and no longer increases.

**Note**

Not all file systems supported by Fedora 14 are documented in this guide. In addition, file systems (e.g. BTRFS) that are unsupported in Red Hat Enterprise Linux 6 are not documented herein either.

### 2.3. Special Considerations

This section enumerates several issues and factors to consider for specific storage configurations.

**Separate Partitions for /home, /opt, /usr/local**

If it is likely that you will upgrade your system in the future, place `/home`, `/opt`, and `/usr/local` on a separate device. This will allow you to reformat the devices/file systems containing the operating system while preserving your user and application data.

**DASD and zFCP Devices on IBM System Z**

On the IBM System Z platform, DASD and zFCP devices are configured via the Channel Command Word (CCW) mechanism. CCW paths must be explicitly added to the system and then brought online. For DASD devices, this is simply means listing the device numbers (or device number ranges) as the `DASD=` parameter at the boot command line or in a CMS configuration file.

For zFCP devices, you must list the device number, logical unit number (LUN), and world wide port name (WWPN). Once the zFCP device is initialized, it is mapped to a CCW path. The `FCP_x=` lines on the boot command line (or in a CMS configuration file) allow you to specify this information for the installer.

**Encrypting Block Devices Using LUKS**

Formatting a block device for encryption using LUKS/`dm-crypt` will destroy any existing formatting on that device. As such, you should decide which devices to encrypt (if any) before the new system's storage configuration is activated as part of the installation process.

**Stale BIOS RAID Metadata**

Moving a disk from a system configured for firmware RAID without removing the RAID metadata from the disk can prevent Anaconda from correctly detecting the disk.
iSCSI Detection and Configuration

Warning

Removing/deleting RAID metadata from disk could potentially destroy any stored data. You should back up your data before proceeding.

To delete RAID metadata from the disk, use the following command:

dmraid -r -E /device/

For more information about managing RAID devices, refer to man dmraid and Chapter 12, Redundant Array of Independent Disks (RAID).

iSCSI Detection and Configuration

For plug and play detection of iSCSI drives, configure them in the firmware of an iBFT boot-capable network interface card (NIC). CHAP authentication of iSCSI targets is supported during installation. However, iSNS discovery is not supported during installation.

FCoE Detection and Configuration

For plug and play detection of fibre-channel over ethernet (FCoE) drives, configure them in the firmware of an EDD boot-capable NIC.

DASD

Direct-access storage devices (DASD) cannot be added/configured during installation. Such devices are specified in the CMS configuration file.

Block Devices with DIF/DIX Enabled

DIF/DIX is a hardware checksum feature provided by certain SCSI host bus adapters and block devices. When DIF/DIX is enabled, errors will occur if the block device is used as a general-purpose block device. Buffered I/O or mmap(2)-based I/O will not work reliably, as there are no interlocks in the buffered write path to prevent buffered data from being overwritten after the DIF/DIX checksum has been calculated.

Because of this, the I/O will later fail with a checksum error. This problem is common to all block device (or file system-based) buffered I/O or mmap(2) I/O, so it is not possible to work around these errors caused by overwrites.

As such, block devices with DIF/DIX enabled should only be used with applications that use O_DIRECT. Such applications should use the raw block device. Alternatively, it is also safe to use the XFS filesystem on a DIF/DIX enabled block device, as long as only O_DIRECT I/O is issued through the file system. XFS is the only filesystem that does not fall back to buffered I/O when doing certain allocation operations.

The responsibility for ensuring that the I/O data does not change after the DIF/DIX checksum has been computed always lies with the application, so only applications designed for use with O_DIRECT I/O and DIF/DIX hardware should use DIF/DIX.
LVM (Logical Volume Manager)

LVM is a tool for logical volume management which includes allocating disks, striping, mirroring and resizing logical volumes.

With LVM, a hard drive or set of hard drives is allocated to one or more physical volumes. LVM physical volumes can be placed on other block devices which might span two or more disks.

The physical volumes are combined into logical volumes, with the exception of the /boot/ partition. The /boot/ partition cannot be on a logical volume group because the boot loader cannot read it. If the root (/) partition is on a logical volume, create a separate /boot/ partition which is not a part of a volume group.

Since a physical volume cannot span over multiple drives, to span over more than one drive, create one or more physical volumes per drive.

![Figure 3.1. Logical Volumes](image)

The volume groups can be divided into logical volumes, which are assigned mount points, such as /home and / and file system types, such as ext2 or ext3. When "partitions" reach their full capacity, free space from the volume group can be added to the logical volume to increase the size of the partition. When a new hard drive is added to the system, it can be added to the volume group, and partitions that are logical volumes can be increased in size.
On the other hand, if a system is partitioned with the ext3 file system, the hard drive is divided into partitions of defined sizes. If a partition becomes full, it is not easy to expand the size of the partition. Even if the partition is moved to another hard drive, the original hard drive space has to be reallocated as a different partition or not used.

**Important**

This chapter on LVM/LVM2 focuses on the use of the LVM GUI administration tool, i.e. system-config-lvm. For comprehensive information on the creation and configuration of LVM partitions in clustered and non-clustered storage, please refer to the Logical Volume Manager Administration guide also provided by Red Hat.

In addition, the Installation Guide for Fedora 14 also documents how to create and configure LVM logical volumes during installation. For more information, refer to the Create LVM Logical Volume section of the Installation Guide for Fedora 14.

### 3.1. What is LVM2?

LVM version 2, or LVM2, was the default for previous versions of Fedora which used the device mapper driver contained in the 2.6 kernel. LVM2 can be upgraded from versions of Fedora running the 2.4 kernel.

### 3.2. Using system-config-lvm

The LVM utility allows you to manage logical volumes within X windows or graphically. You can access the application by selecting from your menu panel System > Administration > Logical Volume Management. Alternatively you can start the Logical Volume Management utility by typing system-config-lvm from a terminal.

In the example used in this section, the following are the details for the volume group that was created during the installation:

<table>
<thead>
<tr>
<th>Volume Group</th>
<th>Mount Point</th>
<th>Extents</th>
</tr>
</thead>
<tbody>
<tr>
<td>boot - (Ext3)</td>
<td>/boot</td>
<td>312</td>
</tr>
<tr>
<td>LogVol00 - (LVM)</td>
<td>(/) directory</td>
<td>312 extents</td>
</tr>
<tr>
<td>LogVol02 - (LVM)</td>
<td>(/home)</td>
<td>128 extents</td>
</tr>
</tbody>
</table>
LogVol03 - (LVM) swap (28 extents).

The logical volumes above were created in disk entity /dev/hda2 while /boot was created in /dev/hda1. The system also consists of ‘Uninitialised Entities’ which are illustrated in Figure 3.7, “Uninitialized Entities”. The figure below illustrates the main window in the LVM utility. The logical and the physical views of the above configuration are illustrated below. The three logical volumes exist on the same physical volume (hda2).

Figure 3.3. Main LVM Window

The figure below illustrates the physical view for the volume. In this window, you can select and remove a volume from the volume group or migrate extents from the volume to another volume group. Steps to migrate extents are discussed in Figure 3.12, “Migrate Extents”.

Figure 3.4. Physical View Window

The figure below illustrates the logical view for the selected volume group. The logical volume size is also indicated with the individual logical volume sizes illustrated.
On the left side column, you can select the individual logical volumes in the volume group to view more details about each. In this example the objective is to rename the logical volume name for ‘LogVol03’ to ‘Swap’. To perform this operation select the respective logical volume and click on the **Edit Properties** button. This will display the Edit Logical Volume window from which you can modify the Logical volume name, size (in extents) and also use the remaining space available in a logical volume group. The figure below illustrates this.

Please note that this logical volume cannot be changed in size as there is currently no free space in the volume group. If there was remaining space, this option would be enabled (see Figure 3.21, “Edit logical volume”). Click on the **OK** button to save your changes (this will remount the volume). To cancel your changes click on the **Cancel** button. To revert to the last snapshot settings click on the **Revert** button. A snapshot can be created by clicking on the **Create Snapshot** button on the LVM utility window. If the selected logical volume is in use by the system (for example) the / (root) directory, this task will not be successful as the volume cannot be unmounted.
3.2.1. Utilizing Uninitialized Entities

'Uninitialized Entities' consist of unpartitioned space and non LVM file systems. In this example partitions 3, 4, 5, 6 and 7 were created during installation and some unpartitioned space was left on the hard disk. Please view each partition and ensure that you read the 'Properties for Disk Entity' on the right column of the window to ensure that you do not delete critical data. In this example partition 1 cannot be initialized as it is /boot. Uninitialized entities are illustrated below.

![Figure 3.7. Uninitialized Entities](image)

In this example, partition 3 will be initialized and added to an existing volume group. To initialize a partition or unpartitioned space, select the partition and click on the Initialize Entity button. Once initialized, a volume will be listed in the 'Unallocated Volumes' list.

3.2.2. Adding Unallocated Volumes to a Volume Group

Once initialized, a volume will be listed in the 'Unallocated Volumes' list. The figure below illustrates an unallocated partition (Partition 3). The respective buttons at the bottom of the window allow you to:

- create a new volume group,
- add the unallocated volume to an existing volume group,
- remove the volume from LVM.

To add the volume to an existing volume group, click on the Add to Existing Volume Group button.
Chapter 3. LVM (Logical Volume Manager)

Figure 3.8. Unallocated Volumes

Clicking on the **Add to Existing Volume Group** button will display a pop up window listing the existing volume groups to which you can add the physical volume you are about to initialize. A volume group may span across one or more hard disks. In this example only one volume group exists as illustrated below.

Figure 3.9. Add physical volume to volume group

Once added to an existing volume group the new logical volume is automatically added to the unused space of the selected volume group. You can use the unused space to:

- create a new logical volume (click on the **Create New Logical Volume(s)** button,

- select one of the existing logical volumes and increase the extents (see *Section 3.2.6, “Extending a Volume Group”*),

- select an existing logical volume and remove it from the volume group by clicking on the **Remove Selected Logical Volume(s)** button. Please note that you cannot select unused space to perform this operation.
The figure below illustrates the logical view of 'VolGroup00' after adding the new volume group.

![Logical view of volume group](image)

Figure 3.10. Logical view of volume group

In the figure below, the uninitialized entities (partitions 3, 5, 6 and 7) were added to 'VolGroup00'.

![Logical view of volume group](image)

Figure 3.11. Logical view of volume group

### 3.2.3. Migrating Extents

To migrate extents from a physical volume, select the volume and click on the **Migrate Selected Extent(s) From Volume** button. Please note that you need to have a sufficient number of free extents to migrate extents within a volume group. An error message will be displayed if you do not have a sufficient number of free extents. To resolve this problem, please extend your volume group (see **Section 3.2.6, “Extending a Volume Group”**). If a sufficient number of free extents is detected in
the volume group, a pop up window will be displayed from which you can select the destination for the extents or automatically let LVM choose the physical volumes (PVs) to migrate them to. This is illustrated below.

![Migrate extents](image)

**Figure 3.12. Migrate Extents**

The figure below illustrates a migration of extents in progress. In this example, the extents were migrated to ‘Partition 3’.

![Migrating extents in progress](image)

**Figure 3.13. Migrating extents in progress**

Once the extents have been migrated, unused space is left on the physical volume. The figure below illustrates the physical and logical view for the volume group. Please note that the extents of LogVol00
which were initially in hda2 are now in hda3. Migrating extents allows you to move logical volumes in case of hard disk upgrades or to manage your disk space better.

3.2.4. Adding a New Hard Disk Using LVM

In this example, a new IDE hard disk was added. The figure below illustrates the details for the new hard disk. From the figure below, the disk is uninitialized and not mounted. To initialize a partition, click on the Initialize Entity button. For more details, see Section 3.2.1, “Utilizing Uninitialized Entities”. Once initialized, LVM will add the new volume to the list of unallocated volumes as illustrated in Figure 3.16, “Create new volume group”.

Figure 3.14. Logical and physical view of volume group

Figure 3.15. Uninitialized hard disk
3.2.5. Adding a New Volume Group

Once initialized, LVM will add the new volume to the list of unallocated volumes where you can add it to an existing volume group or create a new volume group. You can also remove the volume from LVM. The volume if removed from LVM will be listed in the list of ‘Uninitialized Entities’ as illustrated in Figure 3.15, “Uninitialized hard disk”. In this example, a new volume group was created as illustrated below.

![Create new volume group](image)

Figure 3.16. Create new volume group

Once created a new volume group will be displayed in the list of existing volume groups as illustrated below. The logical view will display the new volume group with unused space as no logical volumes have been created. To create a logical volume, select the volume group and click on the Create New Logical Volume button as illustrated below. Please select the extents you wish to use on the volume group. In this example, all the extents in the volume group were used to create the new logical volume.
Figure 3.17. Create new logical volume

The figure below illustrates the physical view of the new volume group. The new logical volume named ‘Backups’ in this volume group is also listed.

Figure 3.18. Physical view of new volume group

3.2.6. Extending a Volume Group

In this example, the objective was to extend the new volume group to include an uninitialized entity (partition). This was to increase the size or number of extents for the volume group. To extend the
volume group, click on the **Extend Volume Group** button. This will display the 'Extend Volume Group' window as illustrated below. On the 'Extend Volume Group' window, you can select disk entities (partitions) to add to the volume group. Please ensure that you check the contents of any 'Uninitialized Disk Entities' (partitions) to avoid deleting any critical data (see Figure 3.15, "Uninitialized hard disk"). In the example, the disk entity (partition) /dev/hda6 was selected as illustrated below.

![Select disk entities](image1)

Figure 3.19. Select disk entities

Once added, the new volume will be added as 'Unused Space' in the volume group. The figure below illustrates the logical and physical view of the volume group after it was extended.

![Logical and physical view of an extended volume group](image2)

Figure 3.20. Logical and physical view of an extended volume group

### 3.2.7. Editing a Logical Volume

The LVM utility allows you to select a logical volume in the volume group and modify its name, size and specify file system options. In this example, the logical volume named 'Backups' was extended onto the remaining space for the volume group.
Clicking on the **Edit Properties** button will display the 'Edit Logical Volume' popup window from which you can edit the properties of the logical volume. On this window, you can also mount the volume after making the changes and mount it when the system is rebooted. Please note that you should indicate the mount point. If the mount point you specify does not exist, a popup window will be displayed prompting you to create it. The 'Edit Logical Volume' window is illustrated below.

![Edit Logical Volume](image)

Figure 3.21. Edit logical volume

If you wish to mount the volume, select the 'Mount' checkbox indicating the preferred mount point. To mount the volume when the system is rebooted, select the 'Mount when rebooted' checkbox. In this example, the new volume will be mounted in `/mnt/backups`. This is illustrated in the figure below.
The figure below illustrates the logical and physical view of the volume group after the logical volume was extended to the unused space. Please note in this example that the logical volume named 'Backups' spans across two hard disks. A volume can be striped across two or more physical devices using LVM.
3.3. References

Use these sources to learn more about LVM.

Installed Documentation

- `rpm -qd lvm2` — This command shows all the documentation available from the lvm package, including man pages.

- `lvm help` — This command shows all LVM commands available.

Useful Websites

- [http://sources.redhat.com/lvm2](http://sources.redhat.com/lvm2) — LVM2 webpage, which contains an overview, link to the mailing lists, and more.

Partitions

The utility `parted` allows users to:

- View the existing partition table
- Change the size of existing partitions
- Add partitions from free space or additional hard drives

By default, the `parted` package is included when installing Fedora. To start `parted`, log in as root and type the command `parted /dev/sda` at a shell prompt (where `/dev/sda` is the device name for the drive you want to configure).

If you want to remove or resize a partition, the device on which that partition resides must not be in use. Creating a new partition on a device which is in use—while possible—is not recommended.

For a device to not be in use, none of the partitions on the device can be mounted, and any swap space on the device must not be enabled.

As well, the partition table should not be modified while it is in use because the kernel may not properly recognize the changes. If the partition table does not match the actual state of the mounted partitions, information could be written to the wrong partition, resulting in lost and overwritten data.

The easiest way to achieve this is to boot your system in rescue mode. When prompted to mount the file system, select `Skip`.

Alternately, if the drive does not contain any partitions in use (system processes that use or lock the file system from being unmounted), you can unmount them with the `umount` command and turn off all the swap space on the hard drive with the `swapoff` command.

Table 4.1, “`parted` commands” contains a list of commonly used `parted` commands. The sections that follow explain some of these commands and arguments in more detail.

Table 4.1. `parted` commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>check minor-num</td>
<td>Perform a simple check of the file system</td>
</tr>
<tr>
<td>cp from to</td>
<td>Copy file system from one partition to another;</td>
</tr>
<tr>
<td></td>
<td><code>from</code> and <code>to</code> are the minor numbers of the</td>
</tr>
<tr>
<td></td>
<td>partitions</td>
</tr>
<tr>
<td>help</td>
<td>Display list of available commands</td>
</tr>
<tr>
<td>mklabel label</td>
<td>Create a disk label for the partition table</td>
</tr>
<tr>
<td>mkfs minor-num file-system-type</td>
<td>Create a file system of type <code>file-system-type</code></td>
</tr>
<tr>
<td>mkpart part-type fs-type start-mb end-mb</td>
<td>Make a partition without creating a new file system</td>
</tr>
<tr>
<td>mkpartfs part-type fs-type start-mb end-mb</td>
<td>Make a partition and create the specified file system</td>
</tr>
<tr>
<td>move minor-num start-mb end-mb</td>
<td>Move the partition</td>
</tr>
<tr>
<td>name minor-num name</td>
<td>Name the partition for Mac and PC98 disklabels only</td>
</tr>
</tbody>
</table>
4.1. Viewing the Partition Table

After starting parted, use the command print to view the partition table. A table similar to the following appears:

Model: ATA ST3160812AS (scsi)
Disk /dev/sda: 160GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos

<table>
<thead>
<tr>
<th>Number</th>
<th>Start</th>
<th>End</th>
<th>Size</th>
<th>Type</th>
<th>File system</th>
<th>Flags</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>32.3kB</td>
<td>107MB</td>
<td>107MB</td>
<td>primary</td>
<td>ext3</td>
<td>boot</td>
</tr>
<tr>
<td>2</td>
<td>107MB</td>
<td>105GB</td>
<td>105GB</td>
<td>primary</td>
<td>ext3</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>105GB</td>
<td>107GB</td>
<td>2147MB</td>
<td>primary</td>
<td>linux-swap</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>107GB</td>
<td>160GB</td>
<td>52.9GB</td>
<td>extended</td>
<td>root</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>107GB</td>
<td>133GB</td>
<td>26.2GB</td>
<td>logical</td>
<td>ext3</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>133GB</td>
<td>133GB</td>
<td>107MB</td>
<td>logical</td>
<td>ext3</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>133GB</td>
<td>160GB</td>
<td>26.6GB</td>
<td>logical</td>
<td>lvm</td>
<td></td>
</tr>
</tbody>
</table>

The first line contains the disk type, manufacturer, model number and interface, and the second line displays the disk label type. The remaining output below the fourth line shows the partition table.

In the partition table, the Minor number is the partition number. For example, the partition with minor number 1 corresponds to /dev/sda1. The Start and End values are in megabytes. Valid Type are metadata, free, primary, extended, or logical. The Filesystem is the file system type, which can be any of the following:

- ext2
- ext3
- fat16
- fat32
- hfs
- jfs
- linux-swap
• ntfs
• reiserfs
• hp-ufs
• sun-ufs
• xfs

If a Filesystem of a device shows no value, this means that its file system type is unknown.

The Flags column lists the flags set for the partition. Available flags are boot, root, swap, hidden, raid, lvm, or lba.

**Tip**

To select a different device without having to restart `parted`, use the `select` command followed by the device name (for example, `/dev/sda`). Doing so allows you to view or configure the partition table of a device.

### 4.2. Creating a Partition

**Warning**

Do not attempt to create a partition on a device that is in use.

Before creating a partition, boot into rescue mode (or unmount any partitions on the device and turn off any swap space on the device).

Start `parted`, where `/dev/sda` is the device on which to create the partition:

**Warning**

To select a different device without having to restart `parted`, use the `select` command followed by the device name (for example, `/dev/sda`). Doing so allows you to view or configure the partition table of a device.

**Warning**

Do not attempt to create a partition on a device that is in use.

Start `parted`, where `/dev/sda` is the device on which to create the partition:

**parted** /dev/sda

View the current partition table to determine if there is enough free space:

**print**

If there is not enough free space, you can resize an existing partition. Refer to Section 4.4, "Resizing a Partition" for details.

### 4.2.1. Making the Partition

From the partition table, determine the start and end points of the new partition and what partition type it should be. You can only have four primary partitions (with no extended partition) on a device. If you need more than four partitions, you can have three primary partitions, one extended partition, and multiple logical partitions within the extended. For an overview of disk partitions, refer to the appendix *An Introduction to Disk Partitions* in the Fedora 14 *Installation Guide*.

For example, to create a primary partition with an ext3 file system from 1024 megabytes until 2048 megabytes on a hard drive type the following command:

**mkpart primary ext3 1024 2048**
Tip

If you use the `mkpartfs` command instead, the file system is created after the partition is created. However, `parted` does not support creating an ext3 file system. Thus, if you wish to create an ext3 file system, use `mkpart` and create the file system with the `mkfs` command as described later.

The changes start taking place as soon as you press `Enter`, so review the command before executing to it.

After creating the partition, use the `print` command to confirm that it is in the partition table with the correct partition type, file system type, and size. Also remember the minor number of the new partition so that you can label any file systems on it. You should also view the output of `cat /proc/partitions` to make sure the kernel recognizes the new partition.

### 4.2.2. Formatting and Labeling the Partition

The partition still does not have a file system. Create the file system:

```
/sbin/mkfs -t ext3 /dev/sda6
```

**Warning**

Formatting the partition permanently destroys any data that currently exists on the partition.

Next, give the file system on the partition a label. For example, if the file system on the new partition is `/dev/sda6` and you want to label it `/work`, use:

```
e2label /dev/sda6 /work
```

By default, the installation program uses the mount point of the partition as the label to make sure the label is unique. You can use any label you want.

Afterwards, create a mount point (e.g. `/work`) as root.

### 4.2.3. Add to `/etc/fstab`

As root, edit the `/etc/fstab` file to include the new partition using the partition's UUID. Use the `blkid -L label` command to retrieve the partition's UUID. The new line should look similar to the following:

```
UUID=93a0429d-0318-45c0-8320-9676ebf1ca79           /work                 ext3    defaults
```

The first column should contain `UUID=` followed by the file system's UUID. The second column should contain the mount point for the new partition, and the next column should be the file system type (for example, ext3 or swap). If you need more information about the format, read the man page with the command `man fstab`.

If the fourth column is the word `defaults`, the partition is mounted at boot time. To mount the partition without rebooting, as root, type the command:
4.3. Removing a Partition

**Warning**

Do not attempt to remove a partition on a device that is in use.

Before removing a partition, boot into rescue mode (or unmount any partitions on the device and turn off any swap space on the device).

Start `parted`, where `/dev/sda` is the device on which to remove the partition:

```
parted /dev/sda
```

View the current partition table to determine the minor number of the partition to remove:

```
print
```

Remove the partition with the command `rm`. For example, to remove the partition with minor number 3:

```
rm 3
```

The changes start taking place as soon as you press `Enter`, so review the command before committing to it.

After removing the partition, use the `print` command to confirm that it is removed from the partition table. You should also view the output of

```
cat /proc/partitions
```

to make sure the kernel knows the partition is removed.

The last step is to remove it from the `/etc/fstab` file. Find the line that declares the removed partition, and remove it from the file.

4.4. Resizing a Partition

**Warning**

Do not attempt to resize a partition on a device that is in use.

Before resizing a partition, boot into rescue mode (or unmount any partitions on the device and turn off any swap space on the device).

Start `parted`, where `/dev/sda` is the device on which to resize the partition:

```
parted /dev/sda
```

View the current partition table to determine the minor number of the partition to resize as well as the start and end points for the partition:
Chapter 4. Partitions

print

To resize the partition, use the `resize` command followed by the minor number for the partition, the starting place in megabytes, and the end place in megabytes. For example:

```bash
resize 3 1024 2048
```

⚠️ **Warning**

A partition cannot be made larger than the space available on the device

After resizing the partition, use the `print` command to confirm that the partition has been resized correctly, is the correct partition type, and is the correct file system type.

After rebooting the system into normal mode, use the command `df` to make sure the partition was mounted and is recognized with the new size.
Chapter 5.

File System Structure

5.1. Why Share a Common Structure?

The file system structure is the most basic level of organization in an operating system. Almost all of the ways an operating system interacts with its users, applications, and security model are dependent on how the operating system organizes files on storage devices. Providing a common file system structure ensures users and programs can access and write files.

File systems break files down into two logical categories:

- Shareable vs. unsharable files
- Variable vs. static files

Shareable files can be accessed locally and by remote hosts; unsharable files are only available locally. Variable files, such as documents, can be changed at any time; static files, such as binaries, do not change without an action from the system administrator.

Categorizing files in this manner helps correlate the function of each file with the permissions assigned to the directories which hold them. How the operating system and its users interact with a file determines the directory in which it is placed, whether that directory is mounted with read-only or read/write permissions, and the level of access each user has to that file. The top level of this organization is crucial; access to the underlying directories can be restricted, otherwise security problems could arise if, from the top level down, access rules do not adhere to a rigid structure.

5.2. Overview of File System Hierarchy Standard (FHS)

Fedora uses the Filesystem Hierarchy Standard (FHS) file system structure, which defines the names, locations, and permissions for many file types and directories.

The FHS document is the authoritative reference to any FHS-compliant file system, but the standard leaves many areas undefined or extensible. This section is an overview of the standard and a description of the parts of the file system not covered by the standard.

The two most important elements of FHS compliance are:

- Compatibility with other FHS-compliant systems
- The ability to mount a /usr/ partition as read-only. This is especially crucial, since /usr/ contains common executables and should not be changed by users. In addition, since /usr/ is mounted as read-only, it should be mountable from the CD-ROM drive or from another machine via a read-only NFS mount.

5.2.1. FHS Organization

The directories and files noted here are a small subset of those specified by the FHS document. Refer to the latest FHS document for the most complete information at http://www.pathname.com/fhs/.

5.2.1.1. Gathering File System Information

The df command reports the system's disk space usage. Its output looks similar to the following:

```bash
$ df
```

```bash
Filesystem   1K-blocks   Used Available Use% Mounted on
/dev/loop0     122880   51200  71680  44% /
```

```bash
- /dev/loop0: A loopback device mounted to the root of the file system. The size of this device is 122880 blocks, with 51200 blocks in use, leaving 71680 blocks available. The usage is 44%.
```
By default, `df` shows the partition size in 1 kilobyte blocks and the amount of used/available disk space in kilobytes. To view the information in megabytes and gigabytes, use the command `df -h`. The `-h` argument stands for “human-readable” format. The output for `df -h` looks similar to the following:

<table>
<thead>
<tr>
<th>Filesystem</th>
<th>Size</th>
<th>Used</th>
<th>Avail</th>
<th>Use%</th>
<th>Mounted on</th>
</tr>
</thead>
<tbody>
<tr>
<td>/dev/mapper/VolGroup00-LogVol00</td>
<td>12G</td>
<td>6.0G</td>
<td>4.6G</td>
<td>57%</td>
<td>/dev/sda1</td>
</tr>
<tr>
<td>none</td>
<td>316M</td>
<td>0</td>
<td>316M</td>
<td>0%</td>
<td>/dev/shm</td>
</tr>
</tbody>
</table>

**Note**

The mounted partition `/dev/shm` represents the system's virtual memory file system.

The `du` command displays the estimated amount of space being used by files in a directory, displaying the disk usage of each subdirectory. The last line in the output of `du` shows the total disk usage of the directory; to see only the total disk usage of a directory in human-readable format, use `du -hs`. For more options, refer to `man du`.

To view the system's partitions and disk space usage in a graphical format, use the Gnome System Monitor by clicking on Applications > System Tools > System Monitor or using the command `gnome-system-monitor`. Select the File Systems tab to view the system's partitions. The figure below illustrates the File Systems tab.
5.2.1.2. The /boot/ Directory

The /boot/ directory contains static files required to boot the system, e.g. the Linux kernel. These files are essential for the system to boot properly.

**Warning**

Do not remove the /boot/ directory. Doing so renders the system unbootable.

5.2.1.3. The /dev/ Directory

The /dev/ directory contains device nodes that represent the following device types:

- Devices attached to the system
- Virtual devices provided by the kernel

These device nodes are essential for the system to function properly. The udevd daemon creates and removes device nodes in /dev/ as needed.

Devices in the /dev/ directory and subdirectories are either character (providing only a serial stream of input/output, e.g. mouse or keyboard) or block (accessible randomly, e.g. hard drive, floppy drive). If you have GNOME or KDE installed, some storage devices are automatically detected when connected (e.g via USB) or inserted (e.g via CD or DVD drive), and a popup window displaying the contents appears.
Table 5.1. Examples of common files in the /dev

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/dev/hda</td>
<td>The master device on primary IDE channel.</td>
</tr>
<tr>
<td>/dev/hdb</td>
<td>The slave device on primary IDE channel.</td>
</tr>
<tr>
<td>/dev/tty0</td>
<td>The first virtual console.</td>
</tr>
<tr>
<td>/dev/tty1</td>
<td>The second virtual console.</td>
</tr>
<tr>
<td>/dev/sda</td>
<td>The first device on primary SCSI or SATA channel.</td>
</tr>
<tr>
<td>/dev/lp0</td>
<td>The first parallel port.</td>
</tr>
</tbody>
</table>

5.2.1.4. The /etc/ Directory

The /etc/ directory is reserved for configuration files that are local to the machine. It should contain no binaries; any binaries should be moved to /bin/ or /sbin/.

For example, the /etc/skel/ directory stores "skeleton" user files, which are used to populate a home directory when a user is first created. Applications also store their configuration files in this directory and may reference them when executed. The /etc/exports file controls which file systems to export to remote hosts.

5.2.1.5. The /lib/ Directory

The /lib/ directory should only contain libraries needed to execute the binaries in /bin/ and /sbin/. These shared library images are used to boot the system or execute commands within the root file system.

5.2.1.6. The /media/ Directory

The /media/ directory contains subdirectories used as mount points for removeable media such as USB storage media, DVDs, CD-ROMs, and Zip disks.

5.2.1.7. The /mnt/ Directory

The /mnt/ directory is reserved for temporarily mounted file systems, such as NFS file system mounts. For all removeable storage media, use the /media/ directory. Automatically detected removeable media will be mounted in the /media directory.

Note

The /mnt directory must not be used by installation programs.

5.2.1.8. The /opt/ Directory

The /opt/ directory is normally reserved for software and add-on packages that are not part of the default installation. A package that installs to /opt/ creates a directory bearing its name, e.g. /opt(packagename/). In most cases, such packages follow a predictable subdirectory structure; most store their binaries in /opt(packagename/bin/ and their man pages in /opt(packagename/man/, and so on.
5.2.1.9. The /proc/ Directory
The /proc/ directory contains special files that either extract information from the kernel or send
information to it. Examples of such information include system memory, cpu information, and hardware
configuration. For more information about /proc/, refer to Section 5.4, “The /proc Virtual File
System”.

5.2.1.10. The /sbin/ Directory
The /sbin/ directory stores binaries essential for booting, restoring, recovering, or repairing the
system. The binaries in /sbin/ require root privileges to use. In addition, /sbin/ contains binaries
used by the system before the /usr/ directory is mounted; any system utilities used after /usr/ is
mounted is typically placed in /usr/sbin/.

At a minimum, the following programs should be stored in /sbin/:

• arp
• clock
• halt
• init
• fsck.*
• grub
• ifconfig
• mingetty
• mkfs.*
• mkswap
• reboot
• route
• shutdown
• swapoff
• swapon

5.2.1.11. The /srv/ Directory
The /srv/ directory contains site-specific data served by a Fedora system. This directory gives users
the location of data files for a particular service, such as FTP, WWW, or CVS. Data that only pertains
to a specific user should go in the /home/ directory.

5.2.1.12. The /sys/ Directory
The /sys/ directory utilizes the new sysfs virtual file system specific to the 2.6 kernel. With the
increased support for hot plug hardware devices in the 2.6 kernel, the /sys/ directory contains
information similar to that held by /proc/, but displays a hierarchical view device information specific
to hot plug devices.
5.2.1.13. The `/usr/` Directory

The `/usr/` directory is for files that can be shared across multiple machines. The `/usr/` directory is often on its own partition and is mounted read-only. At a minimum, `/usr/` should contain the following subdirectories:

- `/usr/bin`, used for binaries
- `/usr/etc`, used for system-wide configuration files
- `/usr/games`
- `/usr/include`, used for C header files
- `/usr/kerberos`, used for Kerberos-related binaries and files
- `/usr/lib`, used for object files and libraries that are not designed to be directly utilized by shell scripts or users
- `/usr/libexec`, contains small helper programs called by other programs
- `/usr/sbin`, stores system administration binaries that do not belong to `/sbin/
- `/usr/share`, stores files that are not architecture-specific
- `/usr/src`, stores source code
- `/usr/tmp` -> `/var/tmp`

The `/usr/` directory should also contain a `/local/` subdirectory. As per the FHS, this subdirectory is used by the system administrator when installing software locally, and should be safe from being overwritten during system updates. The `/usr/local` directory has a structure similar to `/usr/`, and contains the following subdirectories:

- `/usr/local/bin`
- `/usr/local/etc`
- `/usr/local/games`
- `/usr/local/include`
- `/usr/local/lib`
- `/usr/local/libexec`
- `/usr/local/sbin`
- `/usr/local/share`
- `/usr/local/src`
- `/usr/local/src`

Fedora’s usage of `/usr/local/` differs slightly from the FHS. The FHS states that `/usr/local/` should be used to store software that should remain safe from system software upgrades. Since the RPM Package Manager can perform software upgrades safely, it is not necessary to protect files by storing them in `/usr/local/`.

Instead, Fedora uses `/usr/local/` for software local to the machine. For instance, if the `/usr/` directory is mounted as a read-only NFS share from a remote host, it is still possible to install a package or program under the `/usr/local/` directory.
5.2.1.14. The /var/ Directory
Since the FHS requires Linux to mount /usr/ as read-only, any programs that write log files or need spool/ or lock/ directories should write them to the /var/ directory. The FHS states /var/ is for variable data files, which include spool directories/files, logging data, transient/temporary files, and the like.

Below are some of the directories found within the /var/ directory:

- /var/account/
- /var/arpwatch/
- /var/cache/
- /var/crash/
- /var/db/
- /var/empty/
- /var/ftp/
- /var/gdm/
- /var/kerberos/
- /var/lib/
- /var/local/
- /var/lock/
- /var/log/
- /var/mail -> /var/spool/mail/
- /var/mailman/
- /var/named/
- /var/nis/
- /var/opt/
- /var/preserve/
- /var/run/
- /var/spool/
- /var/tmp/
- /var/tux/
- /var/www/
- /var/yp/

System log files, such as messages and lastlog, go in the /var/log/ directory. The /var/lib/rpm/ directory contains RPM system databases. Lock files go in the /var/lock/ directory, usually
in directories for the program using the file. The /var/spool/ directory has subdirectories that store data files for some programs. These subdirectories include:

- /var/spool/at/
- /var/spool/clientmqueue/
- /var/spool/cron/
- /var/spool/cups/
- /var/spool/exim/
- /var/spool/lpd/
- /var/spool/mail/
- /var/spool/mailman/
- /var/spool/mqueue/
- /var/spool/news/
- /var/spool/postfix/
- /var/spool/repackage/
- /var/spool/rwho/
- /var/spool/samba/
- /var/spool/squid/
- /var/spool/squirrelmail/
- /var/spool/up2date/
- /var/spool/uucp
- /var/spool/uucppublic/
- /var/spool/vbox/

### 5.3. Special Fedora File Locations

Fedora extends the FHS structure slightly to accommodate special files.

Most files pertaining to RPM are kept in the /var/lib/rpm/ directory. For more information on RPM, refer to man rpm.

The /var/cache/yum/ directory contains files used by the Package Updater, including RPM header information for the system. This location may also be used to temporarily store RPMs downloaded while updating the system. For more information about Red Hat Network, refer to the documentation online at https://rhn.redhat.com/.

Another location specific to Fedora is the /etc/sysconfig/ directory. This directory stores a variety of configuration information. Many scripts that run at boot time use the files in this directory.
5.4. The /proc Virtual File System

Unlike most file systems, /proc contains neither text nor binary files. Instead, it houses virtual files; hence, /proc is normally referred to as a virtual file system. These virtual files are typically zero bytes in size, even if they contain a large amount of information.

The /proc file system is not used for storage per se. Its main purpose is to provide a file-based interface to hardware, memory, running processes, and other system components. You can retrieve real-time information on many system components by viewing the corresponding /proc file. Some of the files within /proc can also be manipulated (by both users and applications) to configure the kernel.

The following /proc files are relevant in managing and monitoring system storage:

- /proc/devices
  Displays various character and block devices currently configured

- /proc/filesystems
  Lists all file system types currently supported by the kernel

- /proc/mdstat
  Contains current information on multiple-disk or RAID configurations on the system, if they exist

- /proc/mounts
  Lists all mounts currently used by the system

- /proc/partitions
  Contains partition block allocation information

For more information about the /proc file system, refer to the Fedora Deployment Guide.
The Ext3 File System

The ext3 file system is essentially an enhanced version of the ext2 file system. These improvements provide the following advantages:

Availability

After an unexpected power failure or system crash (also called an unclean system shutdown), each mounted ext2 file system on the machine must be checked for consistency by the e2fsck program. This is a time-consuming process that can delay system boot time significantly, especially with large volumes containing a large number of files. During this time, any data on the volumes is unreachable.

The journaling provided by the ext3 file system means that this sort of file system check is no longer necessary after an unclean system shutdown. The only time a consistency check occurs using ext3 is in certain rare hardware failure cases, such as hard drive failures. The time to recover an ext3 file system after an unclean system shutdown does not depend on the size of the file system or the number of files; rather, it depends on the size of the journal used to maintain consistency. The default journal size takes about a second to recover, depending on the speed of the hardware.

Data Integrity

The ext3 file system prevents loss of data integrity in the event that an unclean system shutdown occurs. The ext3 file system allows you to choose the type and level of protection that your data receives. By default, the ext3 volumes are configured to keep a high level of data consistency with regard to the state of the file system.

Speed

Despite writing some data more than once, ext3 has a higher throughput in most cases than ext2 because ext3’s journaling optimizes hard drive head motion. You can choose from three journaling modes to optimize speed, but doing so means trade-offs in regards to data integrity if the system was to fail.

Easy Transition

It is easy to migrate from ext2 to ext3 and gain the benefits of a robust journaling file system without reformatting. Refer to Section 6.2, “Converting to an Ext3 File System” for more on how to perform this task.

As of Fedora 13, ext3 features the following updates:

Default Inode Sizes Changed

The default size of the on-disk inode has increased for more efficient storage of extended attributes, for example ACLs or SELinux attributes. Along with this change, the default number of inodes created on a file system of a given size has been decreased. The inode size may be selected with the mke2fs -I option, or specified in /etc/mke2fs.conf to set system-wide defaults for mke2fs.

Note

If you upgrade to Fedora 14 with the intention of keeping any ext3 file systems intact, you do not need to remake the file system.
New Mount Option: data_err
A new mount option has been added: `data_err=abort`. This option instructs ext3 to abort the journal if an error occurs in a file data (as opposed to metadata) buffer in `data=ordered` mode. This option is disabled by default (i.e. set as `data_err=ignore`).

More Efficient Storage Use
When creating a file system (i.e. `mkfs`), `mke2fs` will attempt to “discard” or “trim” blocks not used by the file system metadata. This helps to optimize SSDs or thinly-provisioned storage. To suppress this behavior, use the `mke2fs -K` option.

The following sections walk you through the steps for creating and tuning ext3 partitions. For ext2 partitions, skip the partitioning and formatting sections below and go directly to Section 6.2, “Converting to an Ext3 File System”.

6.1. Creating an Ext3 File System
After installation, it is sometimes necessary to create a new ext3 file system. For example, if you add a new disk drive to the system, you may want to partition the drive and use the ext3 file system.

The steps for creating an ext3 file system are as follows:

1. Format the partition with the ext3 file system using `mkfs`.
2. Label the file system using `e2label`.

6.2. Converting to an Ext3 File System
The `tune2fs` allows you to convert an ext2 file system to ext3.

Note
Always use the `e2fsck` utility to check your file system before and after using `tune2fs`. A default installation of Fedora 14 uses ext4 for all file systems.

To convert an ext2 file system to ext3, log in as root and type the following command in a terminal:

```
tune2fs -j block_device
```

where `block_device` contains the ext2 file system you wish to convert.

A valid block device could be one of two types of entries:

- A mapped device — A logical volume in a volume group, for example, `/dev/mapper/VolGroup00-LogVol02`.
- A static device — A traditional storage volume, for example, `/dev/sdbX`, where `sdb` is a storage device name and `X` is the partition number.

Issue the `df` command to display mounted file systems.

6.3. Reverting to an Ext2 File System
For simplicity, the sample commands in this section use the following value for the block device:
Reverting to an Ext2 File System

/dev/mapper/VolGroup00-LogVol02

If you wish to revert a partition from ext3 to ext2 for any reason, you must first unmount the partition by logging in as root and typing,

`umount /dev/mapper/VolGroup00-LogVol02`

Next, change the file system type to ext2 by typing the following command as root:

`tune2fs -O ^has_journal /dev/mapper/VolGroup00-LogVol02`

Check the partition for errors by typing the following command as root:

`e2fsck -y /dev/mapper/VolGroup00-LogVol02`

Then mount the partition again as ext2 file system by typing:

`mount -t ext2 /dev/mapper/VolGroup00-LogVol02 /mount/point`

In the above command, replace `/mount/point` with the mount point of the partition.

**Note**

If a `.journal` file exists at the root level of the partition, delete it.

You now have an ext2 partition.

If you want to permanently change the partition to ext2, remember to update the `/etc/fstab` file.
The Ext4 File System

The ext4 file system is a scalable extension of the ext3 file system, which was the default file system in previous versions of Fedora. Ext4 is the default file system of Fedora 14, and can support files and file systems of up to 16 terabytes in size. It also supports an unlimited number of sub-directories (the ext3 file system only supports up to 32,000). Further, ext4 is backward compatible with ext3 and ext2, allowing these older versions to be mounted with the ext4 driver.

Main Features
Ext4 uses extents (as opposed to the traditional block mapping scheme used by ext2 and ext3), which improves performance when using large files and reduces metadata overhead for large files. In addition, ext4 also labels unallocated block groups and inode table sections accordingly, which allows them to be skipped during a file system check. This makes for quicker file system checks, which becomes more beneficial as the file system grows in size.

Allocation Features
The ext4 file system features the following allocation schemes:

- Persistent pre-allocation
- Delayed allocation
- Multi-block allocation
- Stripe-aware allocation

Because of delayed allocation and other performance optimizations, ext4's behavior of writing files to disk is different from ext3. In ext4, a program's writes to the file system are not guaranteed to be on-disk unless the program issues an `fsync()` call afterwards.

By default, ext3 automatically forces newly created files to disk almost immediately even without `fsync()`. This behavior hid bugs in programs that did not use `fsync()` to ensure that written data was on-disk. The ext4 file system, on the other hand, often waits several seconds to write out changes to disk, allowing it to combine and reorder writes for better disk performance than ext3.

**Warning**

If a system crashes while ext4 is waiting to write out changes to disk, the write will fail (i.e. newly created files will not be on-disk). To prevent this, add an `fsync()` call to any programs that depend on writes being on-disk.

Other Ext4 Features
The Ext4 file system also supports the following:

- *Extended attributes (xattr)*, which allows the system to associate several additional name/value pairs per file.
- *Quota journaling*, which avoids the need for lengthy quota consistency checks after a crash.
- *"No journaling" mode*, which allows users to disable journaling for a slight improvement albeit at the cost of file system integrity
- *Subsecond timestamps*
Chapter 7. The Ext4 File System

7.1. Creating an Ext4 File System

To create an ext4 file system, use the `mkfs.ext4` command. In general, the default options are optimal for most usage scenarios, as in:

```
mkfs.ext4 /dev/device
```

Below is a sample output of this command, which displays the resulting file system geometry and features:

```
mke2fs 1.41.9 (22-Aug-2009)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
1954064 inodes, 7813614 blocks
390680 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=4294967296
239 block groups
32768 blocks per group, 32768 fragments per group
8176 inodes per group
Superblock backups stored on blocks:
  32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
  4096000
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done
```

For striped block devices (e.g. RAID5 arrays), the stripe geometry can be specified at the time of file system creation. Using proper stripe geometry greatly enhances performance of an ext4 file system.

When creating file systems on lvm or md volumes, `mkfs.ext4` chooses an optimal geometry. This may also be true on some hardware RAIDs which export geometry information to the operating system.

To specify stripe geometry, use the `-E` option of `mkfs.ext4` (i.e. extended file system options) with the following sub-options:

```
stride=value
    Specifies the RAID chunk size.

stripe-width=value
    Specifies the number of data disks in a RAID device, or the number of stripe units in the stripe.
```

For both sub-options, value must be specified in file system block units. For example, to create a file system with a 64k stride (i.e. 16 x 4096) on a 4k-block file system, use the following command:

```
mkfs.ext4 -E stride=16,stripe-width=64 /dev/device
```

For more information about creating file systems, refer to `man mkfs.ext4`.

7.2. Converting an Ext3 File System to Ext4

Many of the ext4 file system enhancements over ext3 result from modified metadata structures configured on the disk during file system creation. However, an existing ext3 file system can be upgraded to take advantage of other improvements in ext4.
Note

Whenever possible, create a new ext4 file system and migrate your data to it instead of converting from ext3 to ext4. This ensures a better metadata layout, allowing for the enhanced performance natively provided by ext4.

To enable ext4 features on an existing ext3 file system, begin by using the `tune2fs` command in the following manner:

```
tune2fs -O extents,uninit_bg /dev/device
```

The `-O` option sets, clears, or initializes a comma-delimited list of file system features. With the `extents` parameter, the file system will now use extents instead of the indirect block scheme for storing data blocks in an inode (but only for files created after activating extents feature). The `uninit_bg` parameter allows the kernel to mark unused block groups accordingly.

After using `tune2fs` to modify the file system, perform a file system check using the following command:

```
e2fsck -f /dev/device
```

Note that without the file system check, the converted file system cannot be mounted. During the course of conversion, `e2fsck` may print the following warning:

```
One or more block group descriptor checkumfs are invalid
```

This warning is generally benign, as `e2fsck` will repair any invalid block group descriptors it encounters during the conversion process.

An ext3 file system converted to ext4 in the manner described in this section can no longer be mounted as ext3. Refer to Section 7.3, “Mounting an Ext4 File System” for information on how to mount an ext3 file system as ext4 without converting.

In addition, an ext2 file system cannot be converted directly to ext4; it should be converted to ext3, at which point it can be converted to ext4 (or mounted using the ext4 driver). For more information on converting an ext2 file system to ext3, refer to Section 6.2, “Converting to an Ext3 File System”.

For more information on converting an ext3 file system to ext4, refer to `man tune2fs` and `man e2fsck`.

7.3. Mounting an Ext4 File System

An ext4 file system can be mounted with no extra options. For example:

```
mount /dev/device /mount/point
```

The ext4 file system also supports several mount options to influence behavior. For example, the `acl` parameter enables access control lists, while the `user_xattr` parameter enables user extended attributes. To enable both options, use their respective parameters with `-o`, as in:
Chapter 7. The Ext4 File System

mount -o acl,user_xattr /dev/device /mount/point

The tune2fs utility also allows administrators to set default mount options in the file system superblock. For more information on this, refer to man tune2fs.

Write Barriers

By default, ext4 uses write barriers to ensure file system integrity even when power is lost to a device with write caches enabled. For devices without write caches, or with battery-backed write caches, disable barriers using the nobARRIER option, as in:

mount -o nobARRIER /dev/device /mount/point

For more information about write barriers, refer to Chapter 16, Write Barriers.

Mounting an Ext3 File System as Ext4

An ext3 file system can also be mounted as ext4 without changing the format, allowing it to be mounted as ext3 again in the future. To do so, run the following command (where device is an ext3 file system):

mount -t ext4 /dev/device /mount/point

Doing so will only allow the ext3 file system to use ext4-specific features that do not require a file format conversion. These features include delayed allocation and multi-block allocation, and exclude features such as extent mapping.

For more information about mounting an ext4 file system, refer to man mount.

7.4. Resizing an Ext4 File System

Before growing an ext4 file system, ensure that the underlying block device is of an appropriate size to hold the file system later. Use the appropriate resizing methods for the affected block device.

An ext4 file system may be grown while mounted using the resize2fs command, as in:

resize2fs /mount/point size

The resize2fs command can also decrease the size of an unmounted ext4 file system, as in:

resize2fs /dev/device size

When resizing an ext4 file system, the resize2fs utility reads the size in units of file system block size, unless a suffix indicating a specific unit is used. The following suffixes indicate specific units:

- s — 512kb sectors
- K — kilobytes
- M — megabytes
- G — gigabytes

For more information about resizing an ext4 file system, refer to man resize2fs.

7.5. Other Ext4 File System Utilities

Fedora 14 also features other utilities for managing ext4 file systems:
e2fsck
   Used to repair an ext4 file system. This tool checks and repairs an ext4 file system more efficiently than ext3, thanks to updates in the ext4 disk structure.

e2label
   Changes the label on an ext4 file system. This tool can also work on ext2 and ext3 file systems.

quota
   Controls and reports on disk space (blocks) and file (inode) usage by users and groups on an ext4 file system. For more information on using quota, refer to man quota and Section 14.1, “Configuring Disk Quotas”.

As demonstrated earlier in Section 7.3, “Mounting an Ext4 File System”, the tune2fs utility can also adjust configurable file system parameters for ext2, ext3, and ext4 file systems. In addition, the following tools are also useful in debugging and analyzing ext4 file systems:

debugfs
   Debugs ext2, ext3, or ext4 file systems.

e2image
   Saves critical ext2, ext3, or ext4 file system metadata to a file.

For more information about these utilities, refer to their respective man pages.
The XFS File System

XFS is a highly scalable, high-performance file system which was originally designed at Silicon Graphics, Inc. It was created to support extremely large file systems (up to 16 exabytes), files (8 exabytes) and directory structures (tens of millions of entries).

Main Features

XFS supports metadata journaling, which facilitates quicker crash recovery. The XFS file system can also be defragmented and enlarged while mounted and active. In addition, Fedora 14 supports backup and restore utilities specific to XFS.

Allocation Features

XFS features the following allocation schemes:

- Extent-based allocation
- Stripe-aware allocation policies
- Delayed allocation
- Space pre-allocation

Delayed allocation and other performance optimizations affect XFS the same way that they do ext4. Namely, a program’s writes to an XFS file system are not guaranteed to be on-disk unless the program issues an fsync() call afterwards.

For more information on the implications of delayed allocation on a file system, refer to Allocation Features in Chapter 7, The Ext4 File System. The workaround for ensuring writes to disk applies to XFS as well.

Other XFS Features

The XFS file system also supports the following:

- Extended attributes (xattr), which allows the system to associate several additional name/value pairs per file.
- Quota journalling, which avoids the need for lengthy quota consistency checks after a crash.
- Project/directory quotas, allowing quota restrictions over a directory tree.
- Subsecond timestamps

8.1. Creating an XFS File System

To create an XFS file system, use the mkfs.xfs /dev/device command. In general, the default options are optimal for common use.

When using mkfs.xfs on a block device containing an existing file system, use the -f option to force an overwrite of that file system.

Below is a sample output of the mkfs.xfs command:

| meta-data=/dev/device                          | isize=256  | agcount=4, agsize=3277258 blks |
| data                                           | sectsz=512  | attr=2                         |
| naming  =version 2                             | bsize=4096  | blocks=13100032, imaxpct=25    |
|                                                 | sunit=0     | swidth=0 blks                  |
Chapter 8. The XFS File System

For striped block devices (e.g., RAID5 arrays), the stripe geometry can be specified at the time of file system creation. Using proper stripe geometry greatly enhances the performance of an XFS filesystem.

When creating filesystems on lvm or md volumes, `mkfs.xfs` chooses an optimal geometry. This may also be true on some hardware RAIDs which export geometry information to the operating system.

To specify stripe geometry, use the following `mkfs.xfs` sub-options:

- **su=value**
  - Specifies a stripe unit or RAID chunk size. The `value` must be specified in bytes, with an optional `k`, `m`, or `g` suffix.

- **sw=value**
  - Specifies the number of data disks in a RAID device, or the number of stripe units in the stripe.

The following example specifies a chunk size of 64k on a RAID device containing 4 stripe units:

```
mkfs.xfs -d su=64k,sw=4 /dev/device
```

For more information about creating XFS file systems, refer to `man mkfs.xfs`.

### 8.2. Mounting an XFS File System

An XFS file system can be mounted with no extra options, for example:

```
mount /dev/device /mount/point
```

XFS also supports several mount options to influence behavior.

By default, XFS allocates inodes to reflect their on-disk location. However, because some 32-bit userspace applications are not compatible with inode numbers greater than $2^{32}$, XFS will allocate all inodes in disk locations which result in 32-bit inode numbers. This can lead to decreased performance on very large filesystems (i.e. larger than 2 terabytes), because inodes are skewed to the beginning of the block device, while data is skewed towards the end.

To address this, use the `inode64` mount option. This option configures XFS to allocate inodes and data across the entire file system, which can improve performance:

```
mount -o inode64 /dev/device /mount/point
```

### Write Barriers

By default, XFS uses write barriers to ensure file system integrity even when power is lost to a device with write caches enabled. For devices without write caches, or with battery-backed write caches, disable barriers using the `nobARRIER` option, as in:
mount -o nobARRIER /dev/device /mount/point

For more information about write barriers, refer to Chapter 16, Write Barriers.

8.3. XFS Quota Management

The XFS quota subsystem manages limits on disk space (blocks) and file (inode) usage. XFS quotas control and/or report on usage of these items on a user, group, or directory/project level. Also, note that while user, group, and directory/project quotas are enabled independently, group and project quotas are mutually exclusive.

When managing on a per-directory or per-project basis, XFS manages the disk usage of directory hierarchies associated with a specific project. In doing so, XFS recognizes cross-organizational “group” boundaries between projects. This provides a level of control that is broader than what is available when managing quotas for users or groups.

XFS quotas are enabled at mount time, with specific mount options. Each mount option can also be specified as noenforce; this will allow usage reporting without enforcing any limits. Valid quota mount options are:

• uquota/uqnoenforce - User quotas
• gquota/gqnoenforce - Group quotas
• pquota/pqnoenforce - Project quota

Once quotas are enabled, the xfs_quota tool can be used to set limits and report on disk usage. By default, xfs_quota is run interactively, and in basic mode. Basic mode sub-commands simply report usage, and are available to all users. Basic xfs_quota sub-commands include:

quota username/userID
   Show usage and limits for the given username or numeric userID

df
   Shows free and used counts for blocks and inodes.

In contrast, xfs_quota also has an expert mode. The sub-commands of this mode allow actual configuration of limits, and are available only to users with elevated privileges. To use expert mode sub-commands interactively, run xfs_quota -x. Expert mode sub-commands include:

report /path
   Reports quota information for a specific file system.

limit
   Modify quota limits.

For a complete list of sub-commands for either basic or expert mode, use the sub-command help.

All sub-commands can also be run directly from a command line using the -c option, with -x for expert sub-commands. For example, to display a sample quota report for /home (on /dev/blockdevice), use the command xfs_quota -cx 'report -h' /home. This will display output similar to the following:

<table>
<thead>
<tr>
<th>User quota on /home (/dev/blockdevice)</th>
<th>Blocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>User ID      Used  Soft  Hard  Warn/Grace</td>
<td></td>
</tr>
</tbody>
</table>
To set a soft and hard inode count limit of 500 and 700 respectively for user john (whose home directory is /home/john), use the following command:

```
xfs_quota -x -c 'limit isoft=500 ihard=700 /home/john'
```

By default, the `limit` sub-command recognizes targets as users. When configuring the limits for a group, use the `-g` option (as in the previous example). Similarly, use `-p` for projects.

Soft and hard block limits can also be configured using `bsoft/bhard` instead of `isoft/ihard`. For example, to set a soft and hard block limit of 1000m and 1200m, respectively, to group accounting on the /target/path file system, use the following command:

```
xfs_quota -x -c 'limit -g bsoft=1000m bhard=1200m accounting' /target/path
```

**Note**

While real-time blocks (`rtbhard/rtbsoft`) are described in `man xfs_quota` as valid units when setting quotas, the real-time sub-volume is not enabled in this release. As such, the `rtbhard` and `rtbsoft` options are not applicable.

### Setting Project Limits

Before configuring limits for project-controlled directories, add them first to `/etc/projects`. Project names can be added to `/etc/projectid` to map project IDs to project names. Once a project is added to `/etc/projects`, initialize its project directory using the following command:

```
xfs_quota -c 'project -s projectname'
```

Quotas for projects with initialized directories can then be configured, as in:

```
xfs_quota -x -c 'limit -p bsoft=1000m bhard=1200m projectname'
```

Generic quota configuration tools (e.g. `quota`, `repquota`, `edquota`) may also be used to manipulate XFS quotas. However, these tools cannot be used with XFS project quotas.

For more information about setting XFS quotas, refer to `man xfs_quota`.

### 8.4. Increasing the Size of an XFS File System

An XFS file system may be grown while mounted using the `xfs_growfs` command, as in:

```
xfs_growfs /mount/point -D size
```

The `-D size` option grows the file system to the specified size (expressed in file system blocks). Without the `-D size` option, `xfs_growfs` will grow the file system to the maximum size supported by the device.

Before growing an XFS file system with `-D size`, ensure that the underlying block device is of an appropriate size to hold the file system later. Use the appropriate resizing methods for the affected block device.
Note

While XFS file systems can be grown while mounted, their size cannot be reduced at all.

For more information about growing a file system, refer to `man xfs_growfs`.

8.5. Repairing an XFS File System

To repair an XFS file system, use `xfs_repair`, as in:

```
xfs_repair /dev/device
```

The `xfs_repair` utility is highly scalable, and is designed to repair even very large file systems with many inodes efficiently. Note that unlike other Linux file systems, `xfs_repair` does not run at boot time, even when an XFS file system was not cleanly unmounted. In the event of an unclean unmount, `xfs_repair` simply replays the log at mount time, ensuring a consistent file system.

Note

The `xfs_repair` utility cannot repair an XFS file system with a dirty log. To clear the log, mount and unmount the XFS file system. If the log is corrupt and cannot be replayed, use the `-L` option ("force log zeroing") to clear the log, i.e. `xfs_repair -L /dev/device`. Note, however, that this may result in further corruption or data loss.

For more information about repairing an XFS file system, refer to `man xfs_repair`.

8.6. Suspending an XFS File System

To suspend or resume write activity to a file system, use `xfs_freeze`. Suspending write activity allows hardware-based device snapshots to be used to capture the file system in a consistent state.

Note

The `xfs_freeze` utility is provided by the `xfsprogs` package, which is only available on x86_64.

To suspend (i.e. freeze) an XFS file system, use:

```
xfs_freeze -f /mount/point
```

To unfreeze an XFS file system, use:

```
xfs_freeze -u /mount/point
```

When taking an LVM snapshot, it is not necessary to use `xfs_freeze` to suspend the file system first. Rather, the LVM management tools will automatically suspend the XFS file system before taking the snapshot.
Chapter 8. The XFS File System

**Note**

You can also use the `xfs_freeze` utility to freeze/unfreeze an ext3, ext4, GFS2, XFS, and BTRFS, file system. The syntax for doing so is also the same.

For more information about freezing and unfreezing an XFS file system, refer to `man xfs_freeze`.

8.7. Backup and Restoration of XFS File Systems

XFS file system backup and restoration involves two utilities: `xfsdump` and `xfsrestore`.

To backup or dump an XFS file system, use the `xfsdump` utility. Fedora 14 supports backups to tape drives or regular file images, and also allows multiple dumps to be written to the same tape. The `xfsdump` utility also allows a dump to span multiple tapes, although only one dump can be written to a regular file. In addition, `xfsdump` supports incremental backups, and can exclude files from a backup using size, subtree, or inode flags to filter them.

In order to support incremental backups, `xfsdump` uses *dump levels* to determine a base dump to which a specific dump is relative. The `-l` option specifies a dump level (0-9). To perform a full backup, perform a level 0 dump on the file system (i.e. `/path/to/filesystem`), as in:

```
xfsdump -l 0 -f /dev/device /path/to/filesystem
```

**Note**
The `-f` option specifies a destination for a backup. For example, the `/dev/st0` destination is normally used for tape drives. An `xfsdump` destination can be a tape drive, regular file, or remote tape device.

In contrast, an incremental backup will only dump files that changed since the last level 0 dump. A level 1 dump is the first incremental dump after a full dump; the next incremental dump would be level 2, and so on, to a maximum of level 9. So, to perform a level 1 dump to a tape drive:

```
xfsdump -l 1 -f /dev/st0 /path/to/filesystem
```

Conversely, the `xfsrestore` utility restores file systems from dumps produced by `xfsdump`. The `xfsrestore` utility has two modes: a default *simple* mode, and a *cumulative* mode. Specific dumps are identified by session *ID* or *label*. As such, restoring a dump requires its corresponding session ID or label. To display the session ID and labels of all dumps (both full and incremental), use the `-I` option, as in:

```
xfsrestore -I
```

This will provide output similar to the following:

```
file system 0:
  fs id: 45e9af35-efd2-4244-87bc-4762e476cbab
  session 0:
    mount point: bear-05:/mnt/test
device: bear-05:/dev/sdb2
time: Fri Feb 26 16:55:21 2010
  session label: "my_dump_session_label"
  session id: b74a3586-e52e-4a4a-8775-c3334fa8ea2c
  level: 0
```
**xfsrestore Simple Mode**

The *simple* mode allows users to restore an entire file system from a *level 0* dump. After identifying a *level 0* dump's session ID (i.e. *session-ID*), restore it fully to */path/to/destination* using:

```
 xfsrestore -f /dev/st0 -S session-ID /path/to/destination
```

**Note**

The `-f` option specifies the location of the dump, while the `-S` or `-L` option specifies which specific dump to restore. The `-S` option is used to specify a session ID, while the `-L` option is used for session labels. The `-I` option displays both session labels and IDs for each dump.

**xfsrestore Cumulative Mode**

The *cumulative* mode of *xfsrestore* allows file system restoration from a specific incremental backup, i.e. *level 1* to *level 9*. To restore a file system from an incremental backup, simply add the `-r` option, as in:

```
 xfsrestore -f /dev/st0 -S session-ID -r /path/to/destination
```

**Interactive Operation**

The *xfsrestore* utility also allows specific files from a dump to be extracted, added, or deleted. To use *xfsrestore* interactively, use the `-i` option, as in:

```
 xfsrestore -f /dev/st0 -i
```

The interactive dialogue will begin after *xfsrestore* finishes reading the specified device. Available commands in this dialogue include *cd, ls, add, delete, and extract*; for a complete list of commands, use `help`.

For more information about dumping and restoring XFS file systems, refer to `man xfsdump` and `man xfsrestore`.

**8.8. Other XFS File System Utilities**

Fedora 14 also features other utilities for managing XFS file systems:
Chapter 8. The XFS File System

xfs_fsr
Used to defragment mounted XFS file systems. When invoked with no arguments, xfs_fsr defragments all regular files in all mounted XFS file systems. This utility also allows users to suspend a defragmentation at a specified time and resume from where it left off later.

In addition, xfs_fsr also allows the defragmentation of only one file, as in xfs_fsr /path/to/file. Periodic defragmentation of an entire file system is not advised, as this is normally not warranted.

xfs_bmap
Prints the map of disk blocks used by files in an XFS filesystem. This map list each extent used by a specified file, as well as regions in the file with no corresponding blocks (i.e. holes).

xfs_info
Prints XFS file system information.

xfs_admin
Changes the parameters of an XFS file system. The xfs_admin utility can only modify parameters of unmounted devices/file systems.

xfs_copy
Copies the contents of an entire XFS file system to one or more targets in parallel.

The following utilities are also useful in debugging and analyzing XFS file systems:

xfs_metadump
Copies XFS file system metadata to a file. The xfs_metadump utility should only be used to copy unmounted, read-only, or frozen/suspended file systems; otherwise, generated dumps could be corrupted or inconsistent.

xfs_mdrestore
Restores and XFS metadump image (generated using xfs_metadump) to a file system image.

xfs_db
Debugs an XFS file system.

For more information about these utilities, refer to their respective man pages.
Network File System (NFS)

A Network File System (NFS) allows remote hosts to mount file systems over a network and interact with those file systems as though they are mounted locally. This enables system administrators to consolidate resources onto centralized servers on the network.

This chapter focuses on fundamental NFS concepts and supplemental information.

9.1. How It Works

Currently, there are three versions of NFS. NFS version 2 (NFSv2) is older and is widely supported. NFS version 3 (NFSv3) supports safe asynchronous writes and a more robust error handling than NFSv2; it also supports 64-bit file sizes and offsets, allowing clients to access more than 2Gb of file data.

NFS version 4 (NFSv4) works through firewalls and on the Internet, no longer requires an rpcbind service, supports ACLs, and utilizes stateful operations. Fedora supports NFSv2, NFSv3, and NFSv4 clients. When mounting a file system via NFS, Fedora uses NFSv4 by default, if the server supports it.

All versions of NFS can use Transmission Control Protocol (TCP) running over an IP network, with NFSv4 requiring it. NFSv2 and NFSv3 can use the User Datagram Protocol (UDP) running over an IP network to provide a stateless network connection between the client and server.

When using NFSv2 or NFSv3 with UDP, the stateless UDP connection (under normal conditions) has less protocol overhead than TCP. This can translate into better performance on very clean, non-congested networks. However, because UDP is stateless, if the server goes down unexpectedly, UDP clients continue to saturate the network with requests for the server. In addition, when a frame is lost with UDP, the entire RPC request must be retransmitted; with TCP, only the lost frame needs to be resent. For these reasons, TCP is the preferred protocol when connecting to an NFS server.

The mounting and locking protocols have been incorporated into the NFSv4 protocol. The server also listens on the well-known TCP port 2049. As such, NFSv4 does not need to interact with rpcbind, rpc.lockd, and rpc.statd daemons. The rpc.mountd daemon is still required on the NFS server so set up the exports, but is not involved in any over-the-wire operations.

TCP is the default transport protocol for NFS version 2 and 3 under Fedora. UDP can be used for compatibility purposes as needed, but is not recommended for wide usage. NFSv4 requires TCP.

All the RPC/NFS daemon have a `-p` command line option that can set the port, making firewall configuration easier.

After TCP wrappers grant access to the client, the NFS server refers to the `/etc/exports` configuration file to determine whether the client is allowed to access any exported file systems. Once verified, all file and directory operations are available to the user.

---

1 The rpcbind service replaces portmap, which was used in previous versions of Fedora to map RPC program numbers to IP address port number combinations. For more information, refer to Section 9.1.1, “Required Services”.
Important

In order for NFS to work with a default installation of Fedora with a firewall enabled, configure IPTables with the default TCP port 2049. Without proper IPTables configuration, NFS will not function properly.

The NFS initialization script and *rpc.nfsd* process now allow binding to any specified port during system start up. However, this can be error-prone if the port is unavailable, or if it conflicts with another daemon.

9.1.1. Required Services

Fedora uses a combination of kernel-level support and daemon processes to provide NFS file sharing. All NFS versions rely on Remote Procedure Calls (RPC) between clients and servers. RPC services under Fedora 14 are controlled by the *rpcbind* service. To share or mount NFS file systems, the following services work together, depending on which version of NFS is implemented:

Note

The *portmap* service was used to map RPC program numbers to IP address port number combinations in earlier versions of Fedora. This service is now replaced by *rpcbind* in Fedora 14 to enable IPv6 support. For more information about this change, refer to the following links:


- nfs
  
  The *service nfs start* starts the NFS server and the appropriate RPC processes to service requests for shared NFS file systems.

- nfslock
  
  The *service nfslock start* activates a mandatory service that starts the appropriate RPC processes which allow NFS clients to lock files on the server.

- rpcbind
  
  The *rpcbind* accepts port reservations from local RPC services. These ports are then made available (or advertised) so the corresponding remote RPC services can access them. *rpcbind* responds to requests for RPC services and sets up connections to the requested RPC service. This is not used with NFSv4.

The following RPC processes facilitate NFS services:

- rpc.mountd
  
  This process receives mount requests from NFS clients and verifies that the requested file system is currently exported. This process is started automatically by the *nfs* service and does not require user configuration.

- rpc.nfsd
  
  The *rpc.nfsd* allows explicit NFS versions and protocols the server advertises to be defined. It works with the Linux kernel to meet the dynamic demands of NFS clients, such as providing server threads each time an NFS client connects. This process corresponds to the *nfs* service.
rpc.lockd
   **rpc.lockd** allows NFS clients to lock files on the server. If **rpc.lockd** is not started, file locking will fail. **rpc.lockd** implements the Network Lock Manager (NLM) protocol. This process corresponds to the **nfslock** service. This is not used with NFSv4.

rpc.statd
   This process implements the Network Status Monitor (NSM) RPC protocol, which notifies NFS clients when an NFS server is restarted without being gracefully brought down. **rpc.statd** is started automatically by the **nfslock** service, and does not require user configuration. This is not used with NFSv4.

rpc.rquotad
   This process provides user quota information for remote users. **rpc.rquotad** is started automatically by the **nfs** service and does not require user configuration.

rpc.idmapd
   **rpc.idmapd** provides NFSv4 client and server upcalls, which map between on-the-wire NFSv4 names (which are strings in the form of `user@domain`) and local UIDs and GIDs. For **idmapd** to function with NFSv4, the `/etc/idmapd.conf` must be configured. This service is required for use with NFSv4, although not when all hosts share the same DNS domain name.

### 9.2. NFS Client Configuration

The **mount** command mounts NFS shares on the client side. Its format is as follows:

```
mount -t nfs -o options host:/remote/export /local/directory
```

This command uses the following variables:

**options**
   A comma-delimited list of mount options; refer to Section 9.4, "Common NFS Mount Options" for details on valid NFS mount options.

**server**
   The hostname, IP address, or fully qualified domain name of the server exporting the file system you wish to mount

**/remote/export**
   The file system / directory being exported from **server**, i.e. the directory you wish to mount

**/local/directory**
   The client location where **/remote/export** should be mounted

The NFS protocol version used in Fedora 14 is identified by the **mount** options **nfsvers** or **vers**. By default, **mount** will use NFSv4 with **mount -t nfs**. If the server does not support NFSv4, the client will automatically step down to a version supported by the server. If you use the **nfsvers/vers** option to pass a particular version not supported by the server, the mount will fail. The file system type **nfs4** is also available for legacy reasons; this is equivalent to running **mount -t nfs -o nfsvers=4 host:/remote/export /local/directory**.

Refer to **man mount** for more details.

If an NFS share was mounted manually, the share will not be automatically mounted upon reboot. Fedora offers two methods for mounting remote file systems automatically at boot time: the `/etc/fstab` file and the **autofs** service. Refer to Section 9.2.1, "Mounting NFS File Systems using `/etc/fstab" and Section 9.3, "autofs" for more information.
9.2.1. Mounting NFS File Systems using /etc/fstab

An alternate way to mount an NFS share from another machine is to add a line to the /etc/fstab file. The line must state the hostname of the NFS server, the directory on the server being exported, and the directory on the local machine where the NFS share is to be mounted. You must be root to modify the /etc/fstab file.

The general syntax for the line in /etc/fstab is as follows:

```
server:/usr/local/pub    /pub   nfs    rsize=8192,wsize=8192,timeo=14,intr
```

The mount point /pub must exist on the client machine before this command can be executed. After adding this line to /etc/fstab on the client system, use the command `mount /pub`, and the mount point /pub is mounted from the server.

The /etc/fstab file is referenced by the netfs service at boot time, so lines referencing NFS shares have the same effect as manually typing the mount command during the boot process.

A valid /etc/fstab entry to mount an NFS export should contain the following information:

```
server:/remote/export /local/directory nfs options 0 0
```

The variables server, /remote/export, /local/directory, and options are the same ones used when manually mounting an NFS share. Refer to Section 9.2, “NFS Client Configuration” for a definition of each variable.

**Note**

The mount point /local/directory must exist on the client before /etc/fstab is read. Otherwise, the mount will fail.

For more information about /etc/fstab, refer to man fstab.

9.3. autofs

One drawback to using /etc/fstab is that, regardless of how infrequently a user accesses the NFS mounted file system, the system must dedicate resources to keep the mounted file system in place. This is not a problem with one or two mounts, but when the system is maintaining mounts to many systems at one time, overall system performance can be affected. An alternative to /etc/fstab is to use the kernel-based automount utility. An automounter consists of two components:

- a kernel module that implements a file system
- a user-space daemon that performs all of the other functions

The automount utility can mount and unmount NFS file systems automatically (on-demand mounting), therefore saving system resources. It can be used to mount other file systems including AFS, SMBFS, CIFS, and local file systems.

autofs uses /etc/auto.master (master map) as its default primary configuration file. This can be changed to use another supported network source and name using the autofs configuration (in /etc/sysconfig/autofs) in conjunction with the Name Service Switch (NSS) mechanism. An instance of the autofs version 4 daemon was run for each mount point configured in the master map and so it could be run manually from the command line for any given mount point. This is not possible
Improvements in autofs Version 5 over Version 4

with autofs version 5, because it uses a single daemon to manage all configured mount points; as such, all automounters must be configured in the master map. This is in line with the usual requirements of other industry standard automounters. Mount point, hostname, exported directory, and options can all be specified in a set of files (or other supported network sources) rather than configuring them manually for each host.

9.3.1. Improvements in autofs Version 5 over Version 4

autofs version 5 features the following enhancements over version 4:

Direct map support

Direct maps in autofs provide a mechanism to automatically mount file systems at arbitrary points in the file system hierarchy. A direct map is denoted by a mount point of `/-' in the master map. Entries in a direct map contain an absolute path name as a key (instead of the relative path names used in indirect maps).

Lazy mount and unmount support

Multi-mount map entries describe a hierarchy of mount points under a single key. A good example of this is the `hosts` map, commonly used for automounting all exports from a host under `/net/host` as a multi-mount map entry. When using the `-hosts` map, an `ls` of `/net/host` will mount autofs trigger mounts for each export from host and mount and expire them as they are accessed. This can greatly reduce the number of active mounts needed when accessing a server with a large number of exports.

Enhanced LDAP support

The Lightweight Directory Access Protocol (LDAP) support in autofs version 5 has been enhanced in several ways with respect to autofs version 4. The autofs configuration file (`/etc/sysconfig/autofs`) provides a mechanism to specify the autofs schema that a site implements, thus precluding the need to determine this via trial and error in the application itself. In addition, authenticated binds to the LDAP server are now supported, using most mechanisms supported by the common LDAP server implementations. A new configuration file has been added for this support: `/etc/autofs_ldap_auth.conf`. The default configuration file is self-documenting, and uses an XML format.

Proper use of the Name Service Switch (`nsswitch`) configuration.

The Name Service Switch configuration file exists to provide a means of determining from where specific configuration data comes. The reason for this configuration is to allow administrators the flexibility of using the back-end database of choice, while maintaining a uniform software interface to access the data. While the version 4 automounter is becoming increasingly better at handling the NSS configuration, it is still not complete. Autofs version 5, on the other hand, is a complete implementation.

Refer to `man nsswitch.conf` for more information on the supported syntax of this file. Please note that not all NSS databases are valid map sources and the parser will reject ones that are invalid. Valid sources are files, `yp, nis, nisplus, ldap`, and `hesiod`.

Multiple master map entries per autofs mount point

One thing that is frequently used but not yet mentioned is the handling of multiple master map entries for the direct mount point `/-`. The map keys for each entry are merged and behave as one map.

An example is seen in the connectathon test maps for the direct mounts below:

```
/- /tmp/auto_dcthon
/- /tmp/auto_test3_direct
```
Chapter 9. Network File System (NFS)

9.3.2. autofs Configuration

The primary configuration file for the automounter is /etc/auto.master, also referred to as the master map which may be changed as described in the Section 9.3.1, “Improvements in autofs Version 5 over Version 4”. The master map lists autofs-controlled mount points on the system, and their corresponding configuration files or network sources known as automount maps. The format of the master map is as follows:

```
mount-point map-name options
```

The variables used in this format are:

- **mount-point**
  - The autofs mount point e.g /home.

- **map-name**
  - The name of a map source which contains a list of mount points, and the file system location from which those mount points should be mounted. The syntax for a map entry is described below.

- **options**
  - If supplied, these will apply to all entries in the given map provided they don't themselves have options specified. This behavior is different from autofs version 4 where options where cumulative. This has been changed to implement mixed environment compatibility.

The following is a sample line from /etc/auto.master file (displayed with cat /etc/auto.master):

```
/home /etc/auto.misc
```

The general format of maps is similar to the master map, however the “options” appear between the mount point and the location instead of at the end of the entry as in the master map:

```
mount-point [options] location
```

The variables used in this format are:

- **mount-point**
  - This refers to the autofs mount point. This can be a single directory name for an indirect mount or the full path of the mount point for direct mounts. Each direct and indirect map entry key (mount-point above) may be followed by a space separated list of offset directories (sub directory names each beginning with a “/”) making them what is known as a multi-mount entry.

- **options**
  - Whenever supplied, these are the mount options for the map entries that do not specify their own options.

- **location**
  - This refers to the file system location such as a local file system path (preceded with the Sun map format escape character “:” for map names beginning with ‘/’), an NFS file system or other valid file system location.
The following is a sample of contents from a map file (i.e. /etc/auto.misc):

| payroll -fstype=nfs personnel:/dev/hda3 |
| sales -fstype=ext3 :/dev/hda4 |

The first column in a map file indicates the `autofs` mount point (sales and payroll from the server called personnel). The second column indicates the options for the `autofs` mount while the third column indicates the source of the mount. Following the above configuration, the `autofs` mount points will be /home/payroll and /home/sales. The `-fstype=` option is often omitted and is generally not needed for correct operation.

The automounter will create the directories if they do not exist. If the directories exist before the automounter was started, the automounter will not remove them when it exits. You can start or restart the automount daemon by issuing either of the following two commands:

```
service autofs start
service autofs restart
```

Using the above configuration, if a process requires access to an `autofs` unmounted directory such as /home/payroll/2006/July.sxc, the automount daemon automatically mounts the directory. If a timeout is specified, the directory will automatically be unmounted if the directory is not accessed for the timeout period.

You can view the status of the automount daemon by issuing the following command:

```
service autofs status
```

### 9.3.3. Overriding or Augmenting Site Configuration Files

It can be useful to override site defaults for a specific mount point on a client system. For example, consider the following conditions:

- Automounter maps are stored in NIS and the /etc/nsswitch.conf file has the following directive:

  ```
  automount:    files nis
  ```

- The `auto.master` file contains the following

  ```
  +auto.master
  ```

- The NIS `auto.master` map file contains the following:

  ```
  /home auto.home
  ```

- The NIS `auto.home` map contains the following:

  ```
  beth   fileserver.example.com:/export/home/beth
  joe    fileserver.example.com:/export/home/joe
  *      fileserver.example.com:/export/home/
  ```

- The file map /etc/auto.home does not exist.
Given these conditions, let's assume that the client system needs to override the NIS map auto.home and mount home directories from a different server. In this case, the client will need to use the following /etc/auto.master map:

```
/home -/etc/auto.home
+auto.master
```

And the /etc/auto.home map contains the entry:

```
*    labserver.example.com:/export/home/&
```

Because the automounter only processes the first occurrence of a mount point, /home will contain the contents of /etc/auto.home instead of the NIS auto.home map.

Alternatively, if you just want to augment the site-wide auto.home map with a few entries, create a /etc/auto.home file map, and in it put your new entries and at the end, include the NIS auto.home map. Then the /etc/auto.home file map might look similar to:

```
mydir someserver:/export/mydir
+auto.home
```

Given the NIS auto.home map listed above, ls /home would now output:

```
beth joes mydir
```

This last example works as expected because autofs knows not to include the contents of a file map of the same name as the one it is reading. As such, autofs moves on to the next map source in the nsswitch configuration.

### 9.3.4. Using LDAP to Store Automounter Maps

LDAP client libraries must be installed on all systems configured to retrieve automounter maps from LDAP. In Fedora, the openldap package should be installed automatically as a dependency of the automounter. To configure LDAP access, modify /etc/openldap/ldap.conf. Ensure that BASE, URI, and schema are set appropriately for your site.

The most recently established schema for storing automount maps in LDAP is described by rfc2307bis. To use this schema it is necessary to set it in the autofs configuration (/etc/sysconfig/autofs) by removing the comment characters from the schema definition. For example:

```
DEFAULT_MAP_OBJECT_CLASS="automountMap"
DEFAULT_ENTRY_OBJECT_CLASS="automount"
DEFAULT_MAP_ATTRIBUTE="automountMapName"
DEFAULT_ENTRY_ATTRIBUTE="automountKey"
DEFAULT_VALUE_ATTRIBUTE="automountInformation"
```

Ensure that these are the only schema entries not commented in the configuration. Note that the automountKey replaces the cn attribute in the rfc2307bis schema. An LDIF of a sample configuration is described below:

```
# extended LDIF
#
# LDAPv3
# base <> with scope subtree
# filter: ((&(objectclass=automountMap)(automountMapName=auto.master)))
```
9.4. Common NFS Mount Options

Beyond mounting a file system via NFS on a remote host, you can also specify other options at
mount time to make the mounted share easier to use. These options can be used with manual `mount`
commands, `/etc/fstab` settings, and `autofs`.

The following are options commonly used for NFS mounts:
intr
  Allows NFS requests to be interrupted if the server goes down or cannot be reached.

lookupcache=mode
  Specifies how the kernel should manage its cache of directory entries for a given mount point.
  Valid arguments for mode are all, none, or pos/positive.

nfsvers=version
  Specifies which version of the NFS protocol to use, where version is 2, 3, or 4. This is useful
  for hosts that run multiple NFS servers. If no version is specified, NFS uses the highest version
  supported by the kernel and mount command.

  The option vers is identical to nfsvers, and is included in this release for compatibility reasons.

noacl
  Turns off all ACL processing. This may be needed when interfacing with older versions of Fedora,
  Red Hat Enterprise Linux, Red Hat Linux, or Solaris, since the most recent ACL technology is not
  compatible with older systems.

nolock
  Disables file locking. This setting is occasionally required when connecting to older NFS servers.

noexec
  Prevents execution of binaries on mounted file systems. This is useful if the system is mounting a
  non-Linux file system containing incompatible binaries.

nosuid
  Disables set-user-identifier or set-group-identifier bits. This prevents remote users
  from gaining higher privileges by running a setuid program.

port=num
  port=num — Specifies the numeric value of the NFS server port. If num is 0 (the default), then
  mount queries the remote host's rpcbind service for the port number to use. If the remote host's
  NFS daemon is not registered with its rpcbind service, the standard NFS port number of TCP
  2049 is used instead.

rsize=num and wsize=num
  These settings speed up NFS communication for reads (rsize) and writes (wsize) by setting
  a larger data block size (num, in bytes), to be transferred at one time. Be careful when changing
  these values; some older Linux kernels and network cards do not work well with larger block sizes.
  For NFSv2 or NFSv3, the default values for both parameters is set to 8192. For NFSv4, the default
  values for both parameters is set to 32768.

sec=mode
  Specifies the type of security to utilize when authenticating an NFS connection. Its default setting
  is sec=sys, which uses local UNIX UIDs and GIDs by using AUTH_SYS to authenticate NFS
  operations.

  sec=krb5 uses Kerberos V5 instead of local UNIX UIDs and GIDs to authenticate users.

  sec=krb5i uses Kerberos V5 for user authentication and performs integrity checking of NFS
  operations using secure checksums to prevent data tampering.

  sec=krb5p uses Kerberos V5 for user authentication, integrity checking, and encrypts NFS traffic
  to prevent traffic sniffing. This is the most secure setting, but it also involves the most performance
  overhead.
Starting and Stopping NFS

TCP

Instructs the NFS mount to use the TCP protocol.

UDP

Instructs the NFS mount to use the UDP protocol.

For a complete list of options and more detailed information on each one, refer to `man mount` and `man nfs`. For more information on using NFS via TCP or UDP protocols, refer to Section 9.9, “Using NFS over TCP”.

9.5. Starting and Stopping NFS

To run an NFS server, the `rpcbind` service must be running. To verify that `rpcbind` is active, use the following command:

```
service rpcbind status
```

**Note**

Using `service` command to start, stop, or restart a daemon requires root privileges.

If the `rpcbind` service is running, then the `nfs` service can be started. To start an NFS server, use the following command as root:

```
service nfs start
```

**Note**

`nfslock` must also be started for both the NFS client and server to function properly. To start NFS locking, use the following command:

```
service nfslock start
```

If NFS is set to start at boot, ensure that `nfslock` also starts by running `chkconfig --list nfslock`. If `nfslock` is not set to `on`, this implies that you will need to manually run the `service nfslock start` each time the computer starts. To set `nfslock` to automatically start on boot, use `chkconfig nfslock on`.

`nfslock` is only needed for NFSv2 and NFSv3.

To stop the server, use:

```
service nfs stop
```

The `restart` option is a shorthand way of stopping and then starting NFS. This is the most efficient way to make configuration changes take effect after editing the configuration file for NFS. To restart the server, as root, type:

```
service nfs restart
```

The `condrestart` (conditional restart) option only starts `nfs` if it is currently running. This option is useful for scripts, because it does not start the daemon if it is not running. To conditionally restart the server, as root, type:
service nfs condrestart
To reload the NFS server configuration file without restarting the service, as root, type:

service nfs reload

9.6. NFS Server Configuration
There are two ways to configure an NFS server:

- By manually editing the NFS configuration file, i.e. /etc/exports
- Through the command line, i.e. through exportfs

9.6.1. The /etc/exports Configuration File
The /etc/exports file controls which file systems are exported to remote hosts and specifies options. It follows the following syntax rules:

- Blank lines are ignored.
- To add a comment, start a line with the hash mark (#).
- You can wrap long lines with a backslash (\).
- Each exported file system should be on its own individual line.
- Any lists of authorized hosts placed after an exported file system must be separated by space characters.
- Options for each of the hosts must be placed in parentheses directly after the host identifier, without any spaces separating the host and the first parenthesis.

Each entry for an exported file system has the following structure:

```
export  host(options)
```

The aforementioned structure uses the following variables:

- **export**
  - The directory being exported
- **host**
  - The host or network to which the export is being shared
- **options**
  - The options to be used for host

You can specify multiple hosts, along with specific options for each host. To do so, list them on the same line as a space-delimited list, with each hostname followed by its respective options (in parentheses), as in:

```
export  host1(options1)  host2(options2)  host3(options3)
```

For information on different methods for specifying hostnames, refer to Section 9.6.4, “Hostname Formats”.
In its simplest form, the /etc/exports file only specifies the exported directory and the hosts permitted to access it, as in the following example:

```
/exported/directory bob.example.com
```

Here, bob.example.com can mount /exported/directory/ from the NFS server. Because no options are specified in this example, NFS will use default settings, which are:

- **ro**
  - The exported file system is read-only. Remote hosts cannot change the data shared on the file system. To allow hosts to make changes to the file system (i.e. read/write), specify the **rw** option.

- **sync**
  - The NFS server will not reply to requests before changes made by previous requests are written to disk. To enable asynchronous writes instead, specify the option **async**.

- **wdelay**
  - The NFS server will delay writing to the disk if it suspects another write request is imminent. This can improve performance as it reduces the number of times the disk must be accessed by separate write commands, thereby reducing write overhead. To disable this, specify the **no_wdelay**; note that **no_wdelay** is only available if the default **sync** option is also specified.

- **root_squash**
  - This prevents root users connected remotely from having root privileges; instead, the NFS server will assign them the user ID nfsnobody. This effectively "squashes" the power of the remote root user to the lowest local user, preventing possible unauthorized writes on the remote server. To disable root squashing, specify the option **no_root_squash**.

To squash every remote user (including root), use the **all_squash** option. To specify the user and group IDs that the NFS server should assign to remote users from a particular host, use the **anonuid** and **anongid** options, respectively, as in:

```
export host(anonuid=uid,anongid=gid)
```

Here, **uid** and **gid** are user ID number and group ID number, respectively. The **anonuid** and **anongid** options allow you to create a special user/group account for remote NFS users to share.

**Important**

By default, access control lists (ACLs) are supported by NFS. To disable this feature, specify the **no_acl** option when exporting the file system.

Each default for every exported file system must be explicitly overridden. For example, if the **rw** option is not specified, then the exported file system is shared as read-only. The following is a sample line from /etc/exports which overrides two default options:

```
/another/exported/directory 192.168.0.3(rw,async)
```

In this example 192.168.0.3 can mount /another/exported/directory/ read/write and all writes to disk are asynchronous. For more information on exporting options, refer to man exportfs.

Additionally, other options are available where no default value is specified. These include the ability to disable sub-tree checking, allow access from insecure ports, and allow insecure file locks (necessary
Chapter 9. Network File System (NFS)

for certain early NFS client implementations). Refer to man exports for details on these less-used options.

Warning

The format of the /etc/exports file is very precise, particularly in regards to use of the space character. Remember to always separate exported file systems from hosts and hosts from one another with a space character. However, there should be no other space characters in the file except on comment lines.

For example, the following two lines do not mean the same thing:

/home bob.example.com(rw)
/home bob.example.com (rw)

The first line allows only users from bob.example.com read/write access to the /home directory. The second line allows users from bob.example.com to mount the directory as read-only (the default), while the rest of the world can mount it read/write.

9.6.2. The exportfs Command

Every file system being exported to remote users via NFS, as well as the access level for those file systems, are listed in the /etc/exports file. When the nfs service starts, the /usr/sbin/exportfs command launches and reads this file, passes control to rpc.mountd (if NFSv2 or NFSv3) for the actual mounting process, then to rpc.nfsd where the file systems are then available to remote users.

When issued manually, the /usr/sbin/exportfs command allows the root user to selectively export or unexport directories without restarting the NFS service. When given the proper options, the /usr/sbin/exportfs command writes the exported file systems to /var/lib/nfs/xtab. Since rpc.mountd refers to the xtab file when deciding access privileges to a file system, changes to the list of exported file systems take effect immediately.

The following is a list of commonly-used options available for /usr/sbin/exportfs:

- r
  Causes all directories listed in /etc/exports to be exported by constructing a new export list in /etc/lib/nfs/xtab. This option effectively refreshes the export list with any changes made to /etc/exports.

- a
  Causes all directories to be exported or unexported, depending on what other options are passed to /usr/sbin/exportfs. If no other options are specified, /usr/sbin/exportfs exports all file systems specified in /etc/exports.

- o file-systems
  Specifies directories to be exported that are not listed in /etc/exports. Replace file-systems with additional file systems to be exported. These file systems must be formatted in the same way they are specified in /etc/exports. Refer to Section 9.6.1, “The /etc/exports Configuration File” for more information on /etc/exports syntax. This option is often used to test an exported file system before adding it permanently to the list of file systems to be exported.

- i
  Ignores /etc/exports; only options given from the command line are used to define exported file systems.
-u
Unexports all shared directories. The command /usr/sbin/exportfs -ua suspends NFS file sharing while keeping all NFS daemons up. To re-enable NFS sharing, use exportfs -r.

-v
Verbose operation, where the file systems being exported or unexported are displayed in greater detail when the exportfs command is executed.

If no options are passed to the exportfs command, it displays a list of currently exported file systems. For more information about the exportfs command, refer to man exportfs.

9.6.2.1. Using exportfs with NFSv4
The exportfs command is used to maintain the NFS table of exported file systems. When used with no arguments, exportfs shows all the exported directories.

Since NFSv4 no longer utilizes the MOUNT protocol, which was used with the NFSv2 and NFSv3 protocols, the mounting of file systems has changed.

An NFSv4 client now has the ability to see all of the exports served by the NFSv4 server as a single file system, called the NFSv4 pseudo-file system. On Fedora, the pseudo-file system is identified as a single, real file system, identified at export with the fsid=0 option.

9.6.3. Running NFS Behind a Firewall
NFS requires rpcbind, which dynamically assigns ports for RPC services and can cause problems for configuring firewall rules. To allow clients to access NFS shares behind a firewall, edit the /etc/sysconfig/nfs configuration file to control which ports the required RPC services run on.

The /etc/sysconfig/nfs may not exist by default on all systems. If it does not exist, create it and add the following variables, replacing port with an unused port number (alternatively, if the file exists, un-comment and change the default entries as required):

MOUNTD_PORT=port
Controls which TCP and UDP port mountd (rpc.mountd) uses.

STATD_PORT=port
Controls which TCP and UDP port status (rpc.statd) uses.

LOCKD_TCP_PORT=port
Controls which TCP port nlockmgr (rpc.lockd) uses.

LOCKD_UDPPORT=port
Controls which UDP port nlockmgr (rpc.lockd) uses.

If NFS fails to start, check /var/log/messages. Normally, NFS will fail to start if you specify a port number that is already in use. After editing /etc/sysconfig/nfs, restart the NFS service using service nfs restart. Run the rpcinfo -p command to confirm the changes.

To configure a firewall to allow NFS, perform the following steps:
1. Allow TCP and UDP port 2049 for NFS.
2. Allow TCP and UDP port 111 (rpcbind/sunrpc).
3. Allow the TCP and UDP port specified with MOUNTD_PORT="port"
4. Allow the TCP and UDP port specified with STATD_PORT="port"
5. Allow the TCP port specified with `LOCKD_TCPPORT="port"`

6. Allow the UDP port specified with `LOCKD_UDPPORT="port"`

### 9.6.4. Hostname Formats

The host(s) can be in the following forms:

**Single machine**

- A fully-qualified domain name (that can be resolved by the server), hostname (that can be resolved by the server), or an IP address.

**Series of machines specified via wildcards**

- Use the `*` or `?` character to specify a string match. Wildcards are not to be used with IP addresses; however, they may accidentally work if reverse DNS lookups fail. When specifying wildcards in fully qualified domain names, dots (.) are not included in the wildcard. For example, `*.example.com` includes `one.example.com` but does not include `one.two.example.com`.

**IP networks**

- Use `a.b.c.d/z`, where `a.b.c.d` is the network and `z` is the number of bits in the netmask (for example, 192.168.0.0/24). Another acceptable format is `a.b.c.d/netmask`, where `a.b.c.d` is the network and `netmask` is the netmask (for example, 192.168.100.8/255.255.255.0).

**Netgroups**

- Use the format `@group-name`, where `group-name` is the NIS netgroup name.

### 9.7. Securing NFS

NFS is well-suited for sharing entire file systems with a large number of known hosts in a transparent manner. However, with ease-of-use comes a variety of potential security problems. Consider the following sections when exporting NFS file systems on a server or mounting them on a client. Doing so minimizes NFS security risks and better protects data on the server.

#### 9.7.1. Host Access in NFSv2 or NFSv3

NFS controls who can mount an exported file system based on the host making the mount request, not the user that actually uses the file system. Hosts must be given explicit rights to mount the exported file system. Access control is not possible for users, other than through file and directory permissions. In other words, once a file system is exported via NFS, any user on any remote host connected to the NFS server can access the shared data. To limit the potential risks, administrators often allow read-only access or squash user permissions to a common user and group ID. Unfortunately, these solutions prevent the NFS share from being used in the way it was originally intended.

Additionally, if an attacker gains control of the DNS server used by the system exporting the NFS file system, the system associated with a particular hostname or fully qualified domain name can be pointed to an unauthorized machine. At this point, the unauthorized machine is the system permitted to mount the NFS share, since no username or password information is exchanged to provide additional security for the NFS mount.

Wildcards should be used sparingly when exporting directories via NFS, as it is possible for the scope of the wildcard to encompass more systems than intended.

You can also to restrict access to the `rpcbind` service via TCP wrappers. Creating rules with `iptables` can also limit access to ports used by `rpcbind`, `rpc.mountd`, and `rpc.nfsd`.

---

1. `rpcbind`
For more information on securing NFS and `rpcbind`, refer to `man iptables`.

### 9.7.2. Host Access in NFSv4

The release of NFSv4 brought a revolution to authentication and security to NFS exports. NFSv4 mandates the implementation of the `RPCSEC_GSS` kernel module, the Kerberos version 5 GSS-API mechanism, SPKM-3, and LIPKEY. With NFSv4, the mandatory security mechanisms are oriented towards authenticating individual users, and not client machines as used in NFSv2 and NFSv3. As such, for security reasons, it is recommended that you choose NFSv4 over other versions whenever possible.

*Note*

It is assumed that a Kerberos ticket-granting server (KDC) is installed and configured correctly, prior to configuring an NFSv4 server. Kerberos is a network authentication system which allows clients and servers to authenticate to each other through use of symmetric encryption and a trusted third party, the KDC.

NFSv4 includes ACL support based on the Microsoft Windows NT model, not the POSIX model, because of the former's features and wide deployment. NFSv2 and NFSv3 do not have support for native ACL attributes.

Another important security feature of NFSv4 is the removal of the use of the `MOUNT` protocol for mounting file systems. This protocol presented possible security holes because of the way that it processed file handles.


### 9.7.3. File Permissions

Once the NFS file system is mounted read/write by a remote host, the only protection each shared file has is its permissions. If two users that share the same user ID value mount the same NFS file system, they can modify each others files. Additionally, anyone logged in as root on the client system can use the `su -` command to access any files via the NFS share.

By default, access control lists (ACLs) are supported by NFS under Fedora. It is recommended that you keep this feature enabled.

By default, NFS uses *root squashing* when exporting a file system. This sets the user ID of anyone accessing the NFS share as the root user on their local machine to *nobody*. Root squashing is controlled by the default option `root_squash`; for more information about this option, refer to Section 9.6.1, "The `/etc/exports Configuration File". If possible, never disable root squashing.

When exporting an NFS share as read-only, consider using the `all_squash` option. This option makes every user accessing the exported file system take the user ID of the `nfsnobody` user.
Chapter 9. Network File System (NFS)

9.8. NFS and rpcbind

Note
The following section only applies to NFSv2 or NFSv3 implementations that require the **rpcbind** service for backward compatibility.

The **rpcbind** utility maps RPC services to the ports on which they listen. RPC processes notify **rpcbind** when they start, registering the ports they are listening on and the RPC program numbers they expect to serve. The client system then contacts **rpcbind** on the server with a particular RPC program number. The **rpcbind** service redirects the client to the proper port number so it can communicate with the requested service.

Because RPC-based services rely on **rpcbind** to make all connections with incoming client requests, **rpcbind** must be available before any of these services start.

The **rpcbind** service uses TCP wrappers for access control, and access control rules for **rpcbind** affect all RPC-based services. Alternatively, it is possible to specify access control rules for each of the NFS RPC daemons. The **man** pages for **rpc.mountd** and **rpc.statd** contain information regarding the precise syntax for these rules.

9.8.1. Troubleshooting NFS and rpcbind

Because **rpcbind** provides coordination between RPC services and the port numbers used to communicate with them, it is useful to view the status of current RPC services using **rpcbind** when troubleshooting. The **rpcinfo** command shows each RPC-based service with port numbers, an RPC program number, a version number, and an IP protocol type (TCP or UDP).

To make sure the proper NFS RPC-based services are enabled for **rpcbind**, issue the following command as root:

```
rpcinfo -p
```

The following is sample output from this command:

```
program vers proto port
100021  1  udp  32774  nlockmgr
100021  3  udp  32774  nlockmgr
100021  4  udp  32774  nlockmgr
100021  1  tcp  34437  nlockmgr
100021  3  tcp  34437  nlockmgr
100021  4  tcp  34437  nlockmgr
100011  1  udp   819   rquotad
100011  2  udp   819   rquotad
100011  1  tcp   822   rquotad
100011  2  tcp   822   rquotad
100003  2  udp  2049   nfs
100003  3  udp  2049   nfs
100003  2  tcp  2049   nfs
100003  3  tcp  2049   nfs
100005  1  udp  836   mountd
100005  1  tcp  839   mountd
100005  2  udp  836   mountd
100005  2  tcp  839   mountd
100005  3  udp  836   mountd
100005  3  tcp  839   mountd
```
If one of the NFS services does not start up correctly, `rpcbind` will be unable to map RPC requests from clients for that service to the correct port. In many cases, if NFS is not present in `rpcinfo` output, restarting NFS causes the service to correctly register with `rpcbind` and begin working. For instructions on starting NFS, refer to Section 9.5, “Starting and Stopping NFS”.

For more information and a list of options on `rpcinfo`, refer to its man page.

### 9.9. Using NFS over TCP

The default transport protocol for NFS is TCP; however, the Fedora kernel includes support for NFS over UDP. To use NFS over UDP, include the `mount` option `-o udp` when mounting the NFS-exported file system on the client system. Note that NFSv4 on UDP is not standards-compliant, since UDP does not feature congestion control; as such, NFSv4 on UDP is not supported.

There are three ways to configure an NFS file system export:

- On demand via the command line (client side)

- Automatically via the `/etc/fstab` file (client side)

- Automatically via `autofs` configuration files, such as `/etc/auto.master` and `/etc/auto.misc` (server side with NIS)

For example, on demand via the command line (client side):

```bash
mount -o udp shadowman.example.com:/misc/export /misc/local
```

When the NFS mount is specified in `/etc/fstab` (client side):

```bash
server:/usr/local/pub /pub nfs rsize=8192,wsize=8192,timeo=14,intr,udp
```

When the NFS mount is specified in an `autofs` configuration file for a NIS server, available for NIS enabled workstations:

```bash
myproject -rw,soft,intr,rsize=8192,wsize=8192,udp penguin.example.net:/proj52
```

Since the default is TCP, if the `-o udp` option is not specified, the NFS-exported file system is accessed via TCP.

The advantages of using TCP include the following:

- UDP only acknowledges packet completion, while TCP acknowledges every packet. This results in a performance gain on heavily-loaded networks that use TCP when mounting shares.

- TCP has better congestion control than UDP. On a very congested network, UDP packets are the first packets that are dropped. This means that if NFS is writing data (in 8K chunks) all of that 8K must be retransmitted over UDP. Because of TCP's reliability, only parts of that 8K data are transmitted at a time.

- TCP also has better error detection. When a TCP connection breaks (due to the server being unavailable) the client stops sending data and restarts the connection process once the server becomes available. Since UDP is connectionless, the client continues to pound the network with data until the server re-establishes a connection.

The main disadvantage with TCP is that there is a very small performance hit due to the overhead associated with the protocol.
Chapter 9. Network File System (NFS)

9.10. References

Administering an NFS server can be a challenge. Many options, including quite a few not mentioned in this chapter, are available for exporting or mounting NFS shares. Consult the following sources for more information.

**Installed Documentation**

- `/usr/share/doc/nfs-utils-version/` — This directory contains a wealth of information about the NFS implementation for Linux, including a look at various NFS configurations and their impact on file transfer performance.

- `man mount` — Contains a comprehensive look at mount options for both NFS server and client configurations.

- `man fstab` — Gives details for the format of the `/etc/fstab` file used to mount file systems at boot-time.

- `man nfs` — Provides details on NFS-specific file system export and mount options.

- `man exports` — Shows common options used in the `/etc/exports` file when exporting NFS file systems.

**Useful Websites**

- [http://nfs.sourceforge.net/](http://nfs.sourceforge.net/) — The home of the Linux NFS project and a great place for project status updates.


- [http://www.nfsv4.org](http://www.nfsv4.org) — The home of NFS version 4 and all related standards.

- [http://www.vanemery.com/Linux/NFSv4/NFSv4-no-rpcsec.html](http://www.vanemery.com/Linux/NFSv4/NFSv4-no-rpcsec.html) — Describes the details of NFSv4 with Fedora Core 2, which includes the 2.6 kernel.


**Related Books**

- *Managing NFS and NIS* by Hal Stern, Mike Eisler, and Ricardo Labiaga; O'Reilly & Associates — Makes an excellent reference guide for the many different NFS export and mount options available.

- *NFS Illustrated* by Brent Callaghan; Addison-Wesley Publishing Company — Provides comparisons of NFS to other network file systems and shows, in detail, how NFS communication occurs.
FS-Cache

FS-Cache is a persistent local cache that can be used by file systems to take data retrieved from over the network and cache it on local disk. This helps minimize network traffic for users accessing data from a file system mounted over the network (for example, NFS).

The following diagram is a high-level illustration of how FS-Cache works:

![Figure 10.1. FS-Cache Overview](image)

FS-Cache is designed to be as transparent as possible to the users and administrators of a system. Unlike cachefs on Solaris, FS-Cache allows a file system on a server to interact directly with a client's local cache without creating an overmounted file system. With NFS, a mount option instructs the client to mount the NFS share with FS-cache enabled.

FS-Cache does not alter the basic operation of a file system that works over the network - it merely provides that file system with a persistent place in which it can cache data. For instance, a client can still mount an NFS share whether or not FS-Cache is enabled. In addition, cached NFS can handle files that won't fit into the cache (whether individually or collectively) as files can be partially cached and do not have to be read completely up front. FS-Cache also hides all I/O errors that occur in the cache from the client file system driver.

To provide caching services, FS-Cache needs a cache back-end. A cache back-end is a storage driver configured to provide caching services (i.e. cachefiles). In this case, FS-Cache requires a mounted block-based file system that can supports bmap and extended attributes (e.g. ext3) as its cache back-end.
FS-Cache cannot arbitrarily cache any file system, whether through the network or otherwise: the shared file system’s driver must be altered to allow interaction with FS-Cache, data storage/retrieval, and metadata setup and validation. FS-Cache needs indexing keys and coherency data from the cached file system to support persistence: indexing keys to match file system objects to cache objects, and coherency data to determine whether the cache objects are still valid.

10.1. Performance Guarantee

FS-Cache does not guarantee increased performance. Rather, using a cache back-end incurs a performance penalty: for example, cached NFS shares add disk accesses to cross-network lookups. While FS-Cache tries to be as asynchronous as possible, there are synchronous paths (e.g. reads) where this isn’t possible.

For example, using FS-Cache to cache an NFS share between two computers over an otherwise unladen GigE network will not demonstrate any performance improvements on file access. Rather, NFS requests would be satisfied faster from server memory rather than from local disk.

The use of FS-Cache, therefore, is a compromise between various factors. If FS-Cache is being used to cache NFS traffic, for instance, it may slow the client down a little, but massively reduce the network and server loading by satisfying read requests locally without consuming network bandwidth.

10.2. Setting Up a Cache

Currently, Fedora 14 only provides the cachefiles caching back-end. The cachefilesd daemon initiates and manages cachefiles. The /etc/cachefilesd.conf file controls how cachefiles provides caching services. To configure a cache back-end of this type, the cachefilesd package must be installed.

The first setting to configure in a cache back-end is which directory to use as a cache. To configure this, use the following parameter:

```
dir /path/to/cache
```

Typically, the cache back-end directory is set in /etc/cachefilesd.conf as /var/cache/fscache, as in:

```
dir /var/cache/fscache
```

FS-Cache will store the cache in the file system that hosts /path/to/cache. On a laptop, it is advisable to use the the root file system (/) as the host file system, but for a desktop machine it would be more prudent to mount a disk partition specifically for the cache.

File systems that support functionalities required by FS-Cache cache back-end include the Fedora 14 implementations of the following file systems:

- ext3 (with extended attributes enabled)
- ext4
- BTRFS
- XFS

The host file system must support user-defined extended attributes; FS-Cache uses these attributes to store coherency maintenance information. To enable user-defined extended attributes for ext3 file systems (i.e. device), use:
Using the Cache With NFS

```
tune2fs -o user_xattr /dev/device
```

Alternatively, extended attributes for a file system can be enabled at mount time, as in:

```
mount /dev/device /path/to/cache -o user_xattr
```

The cache back-end works by maintaining a certain amount of free space on the partition hosting the cache. It grows and shrinks the cache in response to other elements of the system using up free space, making it safe to use on the root file system (for example, on a laptop). FS-Cache sets defaults on this behavior, which can be configured via cache cull limits. For more information about configuring cache cull limits, refer to Section 10.4, “Setting Cache Cull Limits”.

Once the configuration file is in place, start up the `cachefilesd` daemon:

```
service cachefilesd start
```

To configure `cachefilesd` to start at boot time, execute the following command as root:

```
chkconfig cachefilesd on
```

### 10.3. Using the Cache With NFS

NFS will not use the cache unless explicitly instructed. To configure an NFS mount to use FS-Cache, include the `-o fsc` option to the `mount` command, as in:

```
mount nfs-share:/ /mount/point -o fsc
```

All access to files under `/mount/point` will go through the cache, unless the file is opened for direct I/O or writing (refer to Section 10.3.2, “Cache Limitations With NFS” for more information). NFS indexes cache contents using NFS file handle, not the file name; this means that hard-linked files share the cache correctly.

Caching is supported in version 2, 3, and 4 of NFS. However, each version uses different branches for caching.

#### 10.3.1. Cache Sharing

There are several potential issues to do with NFS cache sharing. Because the cache is persistent, blocks of data in the cache are indexed on a sequence of four keys:

- Level 1: Server details
- Level 2: Some mount options; security type; FSID; uniquifier
- Level 3: File Handle
- Level 4: Page number in file

To avoid coherency management problems between superblocks, all NFS superblocks that wish to cache data have unique Level 2 keys. Normally, two NFS mounts with same source volume and options will share a superblock, and thus share the caching, even if they mount different directories within that volume. Take the following two `mount` commands:

```
mount home0:/disk0/fred /home/fred -o fsc
```
```
mount home0:/disk0/jim /home/jim -o fsc
```
Chapter 10. FS-Cache

Here, `~/fred` and `~/jim` will likely share the superblock as they have the same options, especially if they come from the same volume/partition on the NFS server (`home0`). Now, consider the next two subsequent mount commands:

```bash
mount home0:/disk0/fred /home/fred -o fsc,rsize=230
mount home0:/disk0/jim /home/jim -o fsc,rsize=231
```

In this case, `~/fred` and `~/jim` will not share the superblock as they have different network access parameters, which are part of the Level 2 key. The same goes for the following mount sequence:

```bash
mount home0:/disk0/fred /home/fred1 -o fsc,rsize=230
mount home0:/disk0/fred /home/fred2 -o fsc,rsize=231
```

Here, the contents of the two subtrees (`~/fred1` and `~/fred2`) will be cached twice.

Another way to avoid superblock sharing is to suppress it explicitly with the `nosharecache` parameter. Using the same example:

```bash
mount home0:/disk0/fred /home/fred -o nosharecache,fsc
mount home0:/disk0/jim /home/jim -o nosharecache,fsc
```

However, in this case only one of the superblocks will be permitted to use cache since there is nothing to distinguish the Level 2 keys of `home0:/disk0/fred` and `home0:/disk0/jim`. To address this, add a `unique identifier` on at least one of the mounts, i.e. `fsc=unique-identifier`. For example:

```bash
mount home0:/disk0/fred /home/fred -o nosharecache,fsc
mount home0:/disk0/jim /home/jim -o nosharecache,fsc=jim
```

Here, the unique identifier `jim` will be added to the Level 2 key used in the cache for `~/jim`.

### 10.3.2. Cache Limitations With NFS

Opening a file from a shared file system for direct I/O will automatically bypass the cache. This is because this type of access must be direct to the server.

Opening a file from a shared file system for writing will not work on NFS version 2 and 3. The protocols of these versions do not provide sufficient coherency management information for the client to detect a concurrent write to the same file from another client.

As such, opening a file from a shared file system for either direct I/O or writing will flush the cached copy of the file. FS-Cache will not cache the file again until it is no longer opened for direct I/O or writing.

Furthermore, this release of FS-Cache only caches regular NFS files. FS-Cache will not cache directories, symlinks, device files, FIFOs and sockets.

### 10.4. Setting Cache Cull Limits

The `cachefilesd` daemon works by caching remote data from shared file systems to free space on the disk. This could potentially consume all available free space, which could be bad if the disk also housed the root partition. To control this, `cachefilesd` tries to maintain a certain amount of free space by discarding old objects (i.e. accessed less recently) from the cache. This behavior is known as `cache culling`.
When dealing with file system size, the CacheFiles culling behavior is controlled by three settings in /etc/cachefilesd.conf:

**brun** $N\%$

If the amount of free space rises above $N\%$ of total disk capacity, cachefilesd disables culling.

**bcull** $N\%$

If the amount of free space falls below $N\%$ of total disk capacity, cachefilesd starts culling.

**bstop** $N\%$

If the amount of free space falls below $N\%$, cachefilesd will no longer allocate disk space until culling raises the amount of free space above $N\%$.

Some file systems have a limit on the number of files they can actually support (for example, ext3 can only support up to 32,000 files). This makes it possible for CacheFiles to reach the file system's maximum number of supported files without triggering **bcull** or **bstop**. To address this, cachefilesd also tries to keep the number of files below a file system's limit. This behavior is controlled by the following settings:

**frun** $N\%$

If the number of files the file system can further accommodate falls below $N\%$ of its maximum file limit, cachefilesd disables culling. For example, with **frun 5%**, cachefilesd will disable culling on an ext3 file system if it can accommodate more than 1,600 files, or if the number of files falls below 95% of its limit, i.e. 30,400 files.

**fcull** $N\%$

If the number of files the file system can further accommodate rises above $N\%$ of its maximum file limit, cachefilesd starts culling. For example, with **fcull 5%**, cachefilesd will start culling on an ext3 file system if it can only accommodate 1,600 more files, or if the number of files exceeds 95% of its limit, i.e. 30,400 files.

**fstop** $N\%$

If the number of files the file system can further accommodate rises above $N\%$ of its maximum file limit, cachefilesd will no longer allocate disk space until culling drops the number of files to below $N\%$ of the limit. For example, with **fstop 5%**, cachefilesd will no longer accommodate disk space until culling drops the number of files below 95% of its limit, i.e. 30,400 files.

The default value of $N$ for each setting is as follows:

- **brun/frun** — 10%
- **bcull/fcull** — 7%
- **bstop/fstop** — 3%

When configuring these settings, the following must hold true:

$$0 \leq \texttt{bstop} < \texttt{bcull} < \texttt{brun} < 100$$

$$0 \leq \texttt{fstop} < \texttt{fcull} < \texttt{frun} < 100$$

### 10.5. Statistical Information

FS-Cache also keeps track of general statistical information. To view this information, use:

```
cat /proc/fs/fscache/stats
```
FS-Cache statistics includes information on decision points and object counters. For more details on the statistics provided by FS-Cache, refer to the following kernel document:

/usr/share/doc/kernel-doc-version/Documentation/filesystems/caching/fscache.txt

10.6. References
For more information on cachefilesd and how to configure it, refer to man cachefilesd and man cachefilesd.conf. The following kernel documents also provide additional information:

- /usr/share/doc/cachefilesd-0.5/README
- /usr/share/man/man5/cachefilesd.conf.5.gz
- /usr/share/man/man8/cachefilesd.8.gz

For general information about FS-Cache, including details on its design contraints, available statistics, and capabilities, refer to the following kernel document:

/usr/share/doc/kernel-doc-version/Documentation/filesystems/caching/fscache.txt
Chapter 11.

Encrypted File System

Fedora 14 now supports eCryptfs, a "pseudo-file system" which provides data and filename encryption on a per-file basis. The term "pseudo-file system" refers to the fact that eCryptfs does not have an on-disk format; rather, it is a file system layer that resides on top of an actual file system. The eCryptfs layer provides encryption capabilities.

eCryptfs works like a bind mount, as it intercepts file operations that write to the underlying (i.e. encrypted) file system. The eCryptfs layer adds a header to the metadata of files in the underlying file system. This metadata describes the encryption for that file, and eCryptfs encrypts file data before it is passed to the encrypted file system. Optionally, eCryptfs can also encrypt filenames.

eCryptfs is not an on-disk file system; as such, there is no need to create it via tools such as mkfs. Instead, eCryptfs is initiated by issuing a special mount command. To manage file systems protected by eCryptfs, the ecryptfs-utils package must be installed first.

11.1. Mounting a File System as Encrypted

The easiest way to encrypt a file system with eCryptfs and mount it is interactively. To start this process, execute the following command:

```
mount -t ecryptfs /source /destination
```

Encrypting a directory hierarchy (i.e. /source) with eCryptfs means mounting it to a mount point encrypted by eCryptfs (i.e. /destination). All file operations to /destination will be passed encrypted to the underlying /source file system. In some cases, however, it may be possible for a file operation to modify /source directly without passing through the eCryptfs layer; this could lead to inconsistencies.

This is why for most environments, both /source and /destination should be identical. For example:

```
mount -t ecryptfs /home /home
```

This effectively means encrypting a file system and mounting it on itself. Doing so helps ensure that all file operations to /home pass through the eCryptfs layer.

During the interactive encryption/mount process, mount will allow the following settings to be configured:

- Encryption key type; openssl, tspi, or passphrase. When choosing passphrase, mount will ask for one.
- Cipher; aes, blowfish, des3_ede, cast6, or cast5.
- Key bytesize; 16, 32, 24
- Whether or not plaintext passthrough is enabled
- Whether or not filename encryption is enabled

After the last step of an interactive mount, mount will display all the selections made and perform the mount. This output consists of the command-line option equivalents of each chosen setting. For example, mounting /home with a key type of passphrase, aes cipher, key bytesize of 16 with both plaintext passthrough and filename encryption disabled, the output would be:

```
Attempting to mount with the following options:
```
Chapter 11. Encrypted File System

<table>
<thead>
<tr>
<th>encryptfs_unlink_sigs</th>
</tr>
</thead>
<tbody>
<tr>
<td>encryptfs_key_bytes=16</td>
</tr>
<tr>
<td>encryptfs_cipher=aes</td>
</tr>
<tr>
<td>encryptfs_sig=c7fed37c0a341e19</td>
</tr>
</tbody>
</table>

Mounted eCryptfs

The options in this display can then be passed directly to the command line to encrypt and mount a file system using the same configuration. To do so, use each option as an argument to the `-o` option of `mount`. For example:

```bash
mount -t ecryptfs /home /home -o encryptfs_unlink_sigs,encryptfs_key_bytes=16,encryptfs_cipher=aes,encryptfs_sig=c7fed37c0a341e19
```

11.2. Additional Information

For more information on eCryptfs and its mount options, refer to `man encryptfs` (provided by the `ecryptfs-utils` package). The following Kernel document (provided by the `kernel-doc` package) also provides additional information on eCryptfs:

```
/usr/share/doc/kernel-doc-version/Documentation/filesystems/ecryptfs.txt
```

---

1 This is a single command split into multiple lines, to accommodate printed and PDF versions of this document. All concatenated lines — preceded by the backslash (`\`) — should be treated as one command, sans backslashes.
Chapter 12.

Redundant Array of Independent Disks (RAID)

The basic idea behind RAID is to combine multiple small, inexpensive disk drives into an array to accomplish performance or redundancy goals not attainable with one large and expensive drive. This array of drives appears to the computer as a single logical storage unit or drive.

12.1. What is RAID?

RAID allows information to be spread across several disks. RAID uses techniques such as disk striping (RAID Level 0), disk mirroring (RAID Level 1), and disk striping with parity (RAID Level 5) to achieve redundancy, lower latency, increased bandwidth, and maximized ability to recover from hard disk crashes.

RAID distributes data across each drive in the array by breaking it down into consistently-sized chunks (commonly 256K or 512k, although other values are acceptable). Each chunk is then written to a hard drive in the RAID array according to the RAID level employed. When the data is read, the process is reversed, giving the illusion that the multiple drives in the array are actually one large drive.

12.2. Who Should Use RAID?

System Administrators and others who manage large amounts of data would benefit from using RAID technology. Primary reasons to deploy RAID include:

- Enhances speed
- Increases storage capacity using a single virtual disk
- Minimizes disk failure

12.3. RAID Types

There are three possible RAID approaches: Firmware RAID, Hardware RAID and Software RAID.

Firmware RAID

Firmware RAID (also known as ATARAID) is a type of software RAID where the RAID sets can be configured using a firmware-based menu. The firmware used by this type of RAID also hooks into the BIOS, allowing you to boot from its RAID sets. Different vendors use different on-disk metadata formats to mark the RAID set members. The Intel Matrix RAID is a good example of a firmware RAID system.

Hardware RAID

The hardware-based array manages the RAID subsystem independently from the host. It presents a single disk per RAID array to the host.

A Hardware RAID device may be internal or external to the system, with internal devices commonly consisting of a specialized controller card that handles the RAID tasks transparently to the operating system and with external devices commonly connecting to the system via SCSI, fiber channel, iSCSI, InfiniBand, or other high speed network interconnect and presenting logical volumes to the system.
Chapter 12. Redundant Array of Independent Disks (RAID)

RAID controller cards function like a SCSI controller to the operating system, and handle all the actual drive communications. The user plugs the drives into the RAID controller (just like a normal SCSI controller) and then adds them to the RAID controllers configuration, and the operating system won't know the difference.

Software RAID

Software RAID implements the various RAID levels in the kernel disk (block device) code. It offers the cheapest possible solution, as expensive disk controller cards or hot-swap chassis are not required. Software RAID also works with cheaper IDE disks as well as SCSI disks. With today's faster CPUs, Software RAID also generally outperforms Hardware RAID.

The Linux kernel contains a multi-disk (MD) driver that allows the RAID solution to be completely hardware independent. The performance of a software-based array depends on the server CPU performance and load.

Here are some of the key features of the Linux software RAID stack:

- Multi-threaded design
- Portability of arrays between Linux machines without reconstruction
- Backgrounded array reconstruction using idle system resources
- Hot-swappable drive support
- Automatic CPU detection to take advantage of certain CPU features such as streaming SIMD support
- Automatic correction of bad sectors on disks in an array
- Regular consistency checks of RAID data to ensure the health of the array
- Proactive monitoring of arrays with email alerts sent to a designated email address on important events
- Write-intent bitmaps which drastically speed resync events by allowing the kernel to know precisely which portions of a disk need to be resynced instead of having to resync the entire array
- Resync checkpointing so that if you reboot your computer during a resync, at startup the resync will pick up where it left off and not start all over again
- The ability to change parameters of the array after installation. For example, you can grow a 4-disk raid5 array to a 5-disk raid5 array when you have a new disk to add. This grow operation is done live and does not require you to reinstall on the new array.

12.4. RAID Levels and Linear Support

RAID supports various configurations, including levels 0, 1, 4, 5, 6, 10, and linear. These RAID types are defined as follows:

Level 0

RAID level 0, often called "striping," is a performance-oriented striped data mapping technique. This means the data being written to the array is broken down into strips and written across the

---

1 A hot-swap chassis allows you to remove a hard drive without having to power-down your system.
member disks of the array, allowing high I/O performance at low inherent cost but provides no redundancy.

Many RAID level 0 implementations will only stripe the data across the member devices up to the size of the smallest device in the array. This means that if you have multiple devices with slightly different sizes, each device will get treated as though it is the same size as the smallest drive. Therefore, the common storage capacity of a level 0 array is equal to the capacity of the smallest member disk in a Hardware RAID or the capacity of smallest member partition in a Software RAID multiplied by the number of disks or partitions in the array.

Level 1
RAID level 1, or "mirroring," has been used longer than any other form of RAID. Level 1 provides redundancy by writing identical data to each member disk of the array, leaving a "mirrored" copy on each disk. Mirroring remains popular due to its simplicity and high level of data availability. Level 1 operates with two or more disks, and provides very good data reliability and improves performance for read-intensive applications but at a relatively high cost. \(^2\)

The storage capacity of the level 1 array is equal to the capacity of the smallest mirrored hard disk in a Hardware RAID or the smallest mirrored partition in a Software RAID. Level 1 redundancy is the highest possible among all RAID types, with the array being able to operate with only a single disk present.

Level 4
Level 4 uses parity \(^3\) concentrated on a single disk drive to protect data. Because the dedicated parity disk represents an inherent bottleneck on all write transactions to the RAID array, level 4 is seldom used without accompanying technologies such as write-back caching, or in specific circumstances where the system administrator is intentionally designing the software RAID device with this bottleneck in mind (such as an array that will have little to no write transactions once the array is populated with data). RAID level 4 is so rarely used that it is not available as an option in Anaconda. However, it could be created manually by the user if truly needed.

The storage capacity of Hardware RAID level 4 is equal to the capacity of the smallest member partition multiplied by the number of partitions minus one. Performance of a RAID level 4 array will always be asymmetrical, meaning reads will outperform writes. This is because writes consume extra CPU and main memory bandwidth when generating parity, and then also consume extra bus bandwidth when writing the actual data to disks because you are writing not only the data, but also the parity. Reads need only read the data and not the parity unless the array is in a degraded state. As a result, reads generate less traffic to the drives and across the busses of the computer for the same amount of data transfer under normal operating conditions.

Level 5
This is the most common type of RAID. By distributing parity across all of an array's member disk drives, RAID level 5 eliminates the write bottleneck inherent in level 4. The only performance bottleneck is the parity calculation process itself. With modern CPUs and Software RAID, that is usually not a bottleneck at all since modern CPUs can generate parity very fast. However, if

---

\(^2\) RAID level 1 comes at a high cost because you write the same information to all of the disks in the array, provides data reliability, but in a much less space-efficient manner than parity based RAID levels such as level 5. However, this space inefficiency comes with a performance benefit: parity-based RAID levels consume considerably more CPU power in order to generate the parity while RAID level 1 simply writes the same data more than once to the multiple RAID members with very little CPU overhead. As such, RAID level 1 can outperform the parity-based RAID levels on machines where software RAID is employed and CPU resources on the machine are consistently taxed with operations other than RAID activities.

\(^3\) Parity information is calculated based on the contents of the rest of the member disks in the array. This information can then be used to reconstruct data when one disk in the array fails. The reconstructed data can then be used to satisfy I/O requests to the failed disk before it is replaced and to repopulate the failed disk after it has been replaced.
you have a sufficiently large number of member devices in a software RAID5 array such that the combined aggregate data transfer speed across all devices is high enough, then this bottleneck can start to come into play.

As with level 4, level 5 has asymmetrical performance, with reads substantially outperforming writes. The storage capacity of RAID level 5 is calculated the same way as with level 4.

Level 6
This is a common level of RAID when data redundancy and preservation, and not performance, are the paramount concerns, but where the space inefficiency of level 1 is not acceptable. Level 6 uses a complex parity scheme to be able to recover from the loss of any two drives in the array. This complex parity scheme creates a significantly higher CPU burden on software RAID devices and also imposes an increased burden during write transactions. As such, not only is level 6 asymmetrical in performance like levels 4 and 5, but it is considerably more asymmetrical.

The total capacity of a RAID level 6 array is calculated similarly to RAID level 5 and 4, except that you must subtract 2 devices (instead of 1) from the device count for the extra parity storage space.

Level 10
This RAID level attempts to combine the performance advantages of level 0 with the redundancy of level 1. It also helps to alleviate some of the space wasted in level 1 arrays with more than 2 devices. With level 10, it is possible to create a 3-drive array configured to store only 2 copies of each piece of data, which then allows the overall array size to be 1.5 times the size of the smallest devices instead of only equal to the smallest device (like it would be with a 3-device, level 1 array).

The number of options available when creating level 10 arrays (as well as the complexity of selecting the right options for a specific use case) make it impractical to create during installation. It is possible to create one manually using the command line `mdadm` tool. For details on the options and their respective performance trade-offs, refer to `man md`.

Linear RAID
Linear RAID is a simple grouping of drives to create a larger virtual drive. In linear RAID, the chunks are allocated sequentially from one member drive, going to the next drive only when the first is completely filled. This grouping provides no performance benefit, as it is unlikely that any I/O operations will be split between member drives. Linear RAID also offers no redundancy and, in fact, decreases reliability — if any one member drive fails, the entire array cannot be used. The capacity is the total of all member disks.

12.5. Linux RAID Subsystems
RAID in Linux is composed of the following subsystems:

Linux Hardware RAID controller drivers
Hardware RAID controllers have no specific RAID subsystem in Linux. Because they use special RAID chipsets, hardware RAID controllers come with their own drivers; these drivers allow the system to detect the RAID sets as regular disks.

mdraid
The `mdraid` subsystem was designed as a software RAID solution for Linux; it is also the preferred solution for software RAID under Linux. This subsystem uses its own metadata format, generally refered to as native `mdraid` metadata.
**mdraid** also supports other metadata formats, known as external metadata. Fedora 14 uses **mdraid** with external metadata to access ISW / IMSM (Intel firmware RAID) sets. **mdraid** sets are configured and controlled through the **mdadm** utility.

**dmraid**

*Device-mapper RAID* or **dmraid** refers to device-mapper kernel code that offers the mechanism to piece disks together into a RAID set. This same kernel code does not provide any RAID configuration mechanism.

**dmraid** is configured entirely in user-space, making it easy to support various on-disk metadata formats. As such, **dmraid** is used on a wide variety of firmware RAID implementations. **dmraid** also supports Intel firmware RAID, although Fedora 14 uses **mdraid** to access Intel firmware RAID sets.

### 12.6. RAID Support in the Installer

The **Anaconda** installer will automatically detect any hardware and firmware RAID sets on a system, making them available for installation. **Anaconda** also supports software RAID using **mdraid**, and can recognize existing **mdraid** sets.

**Anaconda** provides utilities for creating RAID sets during installation; however, these utilities only allow partitions (as opposed to entire disks) to be members of new sets. To use an entire disk for a set, simply create a partition on it spanning the entire disk, and use that partition as the RAID set member.

When the root file system uses a RAID set, **Anaconda** will add special kernel command-line options to the bootloader configuration telling the **initrd** which RAID set(s) to activate before searching for the root file system.

For instructions on configuring RAID during installation, refer to the Fedora 14 *Installation Guide*.

### 12.7. Configuring RAID Sets

Most RAID sets are configured during creation, typically through the firmware menu or from the installer. In some cases, you may need to create or modify RAID sets after installing the system, preferably without having to reboot the machine and enter the firmware menu to do so.

Some hardware RAID controllers allow you to configure RAID sets on-the-fly or even define completely new sets after adding extra disks. This requires the use of driver-specific utilities, as there is no standard API for this. Refer to your hardware RAID controller's driver documentation for information on this.

**mdadm**

The **mdadm** command-line tool is used to manage software RAID in Linux, i.e. **mdraid**. For information on the different **mdadm** modes and options, refer to **man mdadm**. The **man** page also contains useful examples for common operations like creating, monitoring, and assembling software RAID arrays.

**dmraid**

As the name suggests, **dmraid** is used to manage device-mapper RAID sets. The **dmraid** tool finds ATARAID devices using multiple metadata format handlers, each supporting various formats. For a complete list of supported formats, run **dmraid** -l.

As mentioned earlier in *Section 12.5, "Linux RAID Subsystems"*, the **dmraid** tool cannot configure RAID sets after creation. For more information about using **dmraid**, refer to **man dmraid**.
### 12.8. Advanced RAID Device Creation

In some cases, you may wish to install the operating system on an array that can't be created after the installation completes. Usually, this means setting up the `/boot` or root file system arrays on a complex RAID device; in such cases, you may need to use array options that are not supported by Anaconda. To work around this, perform the following procedure:

1. Insert the install disk as you normally would.

2. During the initial boot up, select **Rescue Mode** instead of **Install** or **Upgrade**. When the system fully boots into **Rescue mode**, the user will be presented with a command line terminal.

3. From this terminal, use **parted** to create RAID partitions on the target hard drives. Then, use **mdadm** to manually create raid arrays from those partitions using any and all settings and options available. For more information on how to do these, refer to **Chapter 4, Partitions**, **man parted**, and **man mdadm**.

4. Once the arrays are created, you can optionally create file systems on the arrays as well. Refer to **Section 2.2, “Overview of Supported File Systems”** for basic technical information on file systems supported by Fedora 14.

5. Reboot the computer and this time select **Install** or **Upgrade** to install as normal. As Anaconda searches the disks in the system, it will find the pre-existing RAID devices.

6. When asked about how to use the disks in the system, select **Custom Layout** and click **Next**. In the device listing, the pre-existing MD RAID devices will be listed.

7. Select a RAID device, click **Edit** and configure its mount point and (optionally) the type of file system it should use (if you didn't create one earlier) then click **Done**. Anaconda will perform the install to this pre-existing RAID device, preserving the custom options you selected when you created it in **Rescue Mode**.

#### Note

The limited **Rescue Mode** of the installer does not include **man** pages. Both the **man mdadm** and **man md** contain useful information for creating custom RAID arrays, and may be needed throughout the workaround. As such, it can be helpful to either have access to a machine with these **man** pages present, or to print them out prior to booting into **Rescue Mode** and creating your custom arrays.
13.1. What is Swap Space?

Swap space in Linux is used when the amount of physical memory (RAM) is full. If the system needs more memory resources and the RAM is full, inactive pages in memory are moved to the swap space. While swap space can help machines with a small amount of RAM, it should not be considered a replacement for more RAM. Swap space is located on hard drives, which have a slower access time than physical memory.

Swap space can be a dedicated swap partition (recommended), a swap file, or a combination of swap partitions and swap files.

Swap should equal 2x physical RAM for up to 2 GB of physical RAM, and then an additional 1x physical RAM for any amount above 2 GB, but never less than 32 MB.

So, if:

\[
\text{M} = \text{Amount of RAM in GB}, \quad \text{and} \quad \text{S} = \text{Amount of swap in GB}, \text{then}
\]

\[
\begin{align*}
\text{If } \text{M} &< 2 \\
S & = \text{M} \times 2 \\
\text{Else} \\
S & = \text{M} + 2
\end{align*}
\]

Using this formula, a system with 2 GB of physical RAM would have 4 GB of swap, while one with 3 GB of physical RAM would have 5 GB of swap. Creating a large swap space partition can be especially helpful if you plan to upgrade your RAM at a later time.

For systems with really large amounts of RAM (more than 32 GB) you can likely get away with a smaller swap partition (around 1x, or less, of physical RAM).

**Important**

File systems and LVM2 volumes assigned as swap space should not be in use when being modified. Any attempts to modify swap will fail if a system process or the kernel is using swap space. Use the `free` and `cat /proc/swaps` commands to verify how much and where swap is in use.

You should modify swap space while the system is booted in rescue mode; for instructions on how to boot in rescue mode, refer to the Installation Guide. When prompted to mount the file system, select Skip.

13.2. Adding Swap Space

Sometimes it is necessary to add more swap space after installation. For example, you may upgrade the amount of RAM in your system from 128 MB to 256 MB, but there is only 256 MB of swap space. It might be advantageous to increase the amount of swap space to 512 MB if you perform memory-intensive operations or run applications that require a large amount of memory.

You have three options: create a new swap partition, create a new swap file, or extend swap on an existing LVM2 logical volume. It is recommended that you extend an existing logical volume.
Chapter 13. Swap Space

13.2.1. Extending Swap on an LVM2 Logical Volume

By default, Fedora 14 uses all available space during installation. If this is the case with your system, then you must first add a new physical volume to the volume group used by the swap space. For instructions on how to do so, refer to Section 3.2.2, “Adding Unallocated Volumes to a Volume Group”.

After adding additional storage to the swap space’s volume group, it is now possible to extend it. To do so, perform the following procedure (assuming /dev/VolGroup00/LogVol01 is the volume you want to extend by 256MB):

1. Disable swapping for the associated logical volume:
   
   ```bash
   swapoff -v /dev/VolGroup00/LogVol01
   ```

2. Resize the LVM2 logical volume by 256 MB:

   ```bash
   lvresize /dev/VolGroup00/LogVol01 -L +256M
   ```

3. Format the new swap space:

   ```bash
   mkswap /dev/VolGroup00/LogVol01
   ```

4. Enable the extended logical volume:

   ```bash
   swapon -v /dev/VolGroup00/LogVol01
   ```

To test if the logical volume was successfully extended, use `cat /proc/swaps` or `free` to inspect the swap space.

13.2.2. Creating an LVM2 Logical Volume for Swap

To add a swap volume group (assuming /dev/VolGroup00/LogVol02 is the swap volume you want to add):

1. Create the LVM2 logical volume of size 256 MB:

   ```bash
   lvcreate VolGroup00 -n LogVol02 -L 256M
   ```

2. Format the new swap space:

   ```bash
   mkswap /dev/VolGroup00/LogVol02
   ```

3. Add the following entry to the `/etc/fstab` file:

   ```bash
   /dev/VolGroup00/LogVol02 swap swap defaults 0 0
   ```

4. Enable the extended logical volume:

   ```bash
   swapon -v /dev/VolGroup00/LogVol02
   ```

To test if the logical volume was successfully created, use `cat /proc/swaps` or `free` to inspect the swap space.

13.2.3. Creating a Swap File

To add a swap file:

1. Determine the size of the new swap file in megabytes and multiply by 1024 to determine the number of blocks. For example, the block size of a 64 MB swap file is 65536.
2. At a shell prompt as root, type the following command with count being equal to the desired block size:

   \texttt{dd if=/dev/zero of=/swapfile bs=1024 count=65536}

3. Setup the swap file with the command:

   \texttt{mkswap /swapfile}

4. To enable the swap file immediately but not automatically at boot time:

   \texttt{swapon /swapfile}

5. To enable it at boot time, edit \texttt{/etc/fstab} to include the following entry:

   \texttt{/swapfile swap swap defaults 0 0}

   The next time the system boots, it enables the new swap file.

To test if the new swap file was successfully created, use \texttt{cat /proc/swaps} or \texttt{free} to inspect the swap space.

### 13.3. Removing Swap Space

Sometimes it can be prudent to reduce swap space after installation. For example, say you downgraded the amount of RAM in your system from 1 GB to 512 MB, but there is 2 GB of swap space still assigned. It might be advantageous to reduce the amount of swap space to 1 GB, since the larger 2 GB could be wasting disk space.

You have three options: remove an entire LVM2 logical volume used for swap, remove a swap file, or reduce swap space on an existing LVM2 logical volume.

#### 13.3.1. Reducing Swap on an LVM2 Logical Volume

To reduce an LVM2 swap logical volume (assuming \texttt{/dev/VolGroup00/LogVol01} is the volume you want to reduce):

1. Disable swapping for the associated logical volume:

   \texttt{swapoff -v /dev/VolGroup00/LogVol01}

2. Reduce the LVM2 logical volume by 512 MB:

   \texttt{lvreduce /dev/VolGroup00/LogVol01 -L -512M}

3. Format the new swap space:

   \texttt{mkswap /dev/VolGroup00/LogVol01}

4. Enable the extended logical volume:

   \texttt{swapon -v /dev/VolGroup00/LogVol01}

To test if the swap's logical volume size was successfully reduced, use \texttt{cat /proc/swaps} or \texttt{free} to inspect the swap space.
13.3.2. Removing an LVM2 Logical Volume for Swap
To remove a swap volume group (assuming `/dev/VolGroup00/LogVol02` is the swap volume you want to remove):

1. Disable swapping for the associated logical volume:
   
   ```bash
   swapoff -v /dev/VolGroup00/LogVol02
   ```

2. Remove the LVM2 logical volume of size 512 MB:
   
   ```bash
   lvremove /dev/VolGroup00/LogVol02
   ```

3. Remove the following entry from the `/etc/fstab` file:
   
   ```bash
   /dev/VolGroup00/LogVol02 swap swap defaults 0 0
   ```

To test if the logical volume size was successfully removed, use `cat /proc/swaps` or `free` to inspect the swap space.

13.3.3. Removing a Swap File
To remove a swap file:

1. At a shell prompt as root, execute the following command to disable the swap file (where `/swapfile` is the swap file):
   
   ```bash
   swapoff -v /swapfile
   ```

2. Remove its entry from the `/etc/fstab` file.

3. Remove the actual file:
   
   ```bash
   rm /swapfile
   ```

13.4. Moving Swap Space
To move swap space from one location to another, follow the steps for removing swap space, and then follow the steps for adding swap space.
Disk Quotas

Disk space can be restricted by implementing disk quotas which alert a system administrator before a user consumes too much disk space or a partition becomes full.

Disk quotas can be configured for individual users as well as user groups. This makes it possible to manage the space allocated for user-specific files (such as email) separately from the space allocated to the projects a user works on (assuming the projects are given their own groups).

In addition, quotas can be set not just to control the number of disk blocks consumed but to control the number of inodes (data structures that contain information about files in UNIX file systems). Because inodes are used to contain file-related information, this allows control over the number of files that can be created.

The quota RPM must be installed to implement disk quotas.

14.1. Configuring Disk Quotas

To implement disk quotas, use the following steps:

1. Enable quotas per file system by modifying the /etc/fstab file.
2. Remount the file system(s).
3. Create the quota database files and generate the disk usage table.
4. Assign quota policies.

Each of these steps is discussed in detail in the following sections.

14.1.1. Enabling Quotas

As root, using a text editor, edit the /etc/fstab file. Add the usrquota and/or grpquota options to the file systems that require quotas:

```
/dev/VolGroup00/LogVol00 / ext3 defaults 1 1
LABEL=/boot /boot ext3 defaults 1 2
none /dev/pts devpts gid=5,mode=620 0 0
none /dev/shm tmpfs defaults 0 0
none /proc proc defaults 0 0
none /sys sysfs defaults 0 0
/dev/VolGroup00/LogVol02 /home ext3 defaults,usrquota,grpquota 1 2
/dev/VolGroup00/LogVol01 swap swap defaults 0 0 ...
```

In this example, the /home file system has both user and group quotas enabled.

**Note**

The following examples assume that a separate /home partition was created during the installation of Fedora. The root (/) partition can be used for setting quota policies in the /etc/fstab file.

14.1.2. Remounting the File Systems

After adding the usrquota and/or grpquota options, remount each file system whose fstab entry has been modified. If the file system is not in use by any process, use one of the following methods:
Chapter 14. Disk Quotas

- Issue the `umount` command followed by the `mount` command to remount the file system. Refer to the `man` page for both `umount` and `mount` for the specific syntax for mounting and unmounting various file system types.

- Issue the `mount -o remount file-system` command (where `file-system` is the name of the file system) to remount the file system. For example, to remount the `/home` file system, the command to issue is `mount -o remount /home`.

If the file system is currently in use, the easiest method for remounting the file system is to reboot the system.

14.1.3. Creating the Quota Database Files

After each quota-enabled file system is remounted, the system is capable of working with disk quotas. However, the file system itself is not yet ready to support quotas. The next step is to run the `quotacheck` command.

The `quotacheck` command examines quota-enabled file systems and builds a table of the current disk usage per file system. The table is then used to update the operating system's copy of disk usage. In addition, the file system's disk quota files are updated.

To create the quota files (`aquota.user` and `aquota.group`) on the file system, use the `-c` option of the `quotacheck` command. For example, if user and group quotas are enabled for the `/home` file system, create the files in the `/home` directory:

```
quotacheck -cug /home
```

The `-c` option specifies that the quota files should be created for each file system with quotas enabled, the `-u` option specifies to check for user quotas, and the `-g` option specifies to check for group quotas.

If neither the `-u` or `-g` options are specified, only the user quota file is created. If only `-g` is specified, only the group quota file is created.

After the files are created, run the following command to generate the table of current disk usage per file system with quotas enabled:

```
quotacheck -avug
```

The options used are as follows:

- `a` Check all quota-enabled, locally-mounted file systems

- `v` Display verbose status information as the quota check proceeds

- `u` Check user disk quota information

- `g` Check group disk quota information

After `quotacheck` has finished running, the quota files corresponding to the enabled quotas (user and/or group) are populated with data for each quota-enabled locally-mounted file system such as `/home`. 
14.1.4. Assigning Quotas per User

The last step is assigning the disk quotas with the `edquota` command.

To configure the quota for a user, as root in a shell prompt, execute the command:

```
edquota username
```

Perform this step for each user who needs a quota. For example, if a quota is enabled in `/etc/fstab` for the `/home` partition (`/dev/VolGroup00/LogVol02` in the example below) and the command `edquota testuser` is executed, the following is shown in the editor configured as the default for the system:

```
Disk quotas for user testuser (uid 501):
Filesystem                blocks     soft     hard   inodes   soft   hard
/dev/VolGroup00/LogVol02  440436   0        0     37418      0      0
```

**Note**

The text editor defined by the `EDITOR` environment variable is used by `edquota`. To change the editor, set the `EDITOR` environment variable in your `~/.bash_profile` file to the full path of the editor of your choice.

The first column is the name of the file system that has a quota enabled for it. The second column shows how many blocks the user is currently using. The next two columns are used to set soft and hard block limits for the user on the file system. The `inodes` column shows how many inodes the user is currently using. The last two columns are used to set the soft and hard inode limits for the user on the file system.

The hard block limit is the absolute maximum amount of disk space that a user or group can use. Once this limit is reached, no further disk space can be used.

The soft block limit defines the maximum amount of disk space that can be used. However, unlike the hard limit, the soft limit can be exceeded for a certain amount of time. That time is known as the grace period. The grace period can be expressed in seconds, minutes, hours, days, weeks, or months.

If any of the values are set to 0, that limit is not set. In the text editor, change the desired limits. For example:

```
Disk quotas for user testuser (uid 501):
Filesystem                blocks     soft     hard   inodes   soft   hard
/dev/VolGroup00/LogVol02  440436  500000   550000    37418      0      0
```

To verify that the quota for the user has been set, use the command:

```
quota testuser
```

14.1.5. Assigning Quotas per Group

Quotas can also be assigned on a per-group basis. For example, to set a group quota for the `devel` group (the group must exist prior to setting the group quota), use the command:

```
edquota -g devel
```
This command displays the existing quota for the group in the text editor:

```
filesystem    blocks    soft     hard    inodes    soft    hard
/dev/VolGroup00/LogVol02  440400       0        0     37418       0       0
```

Modify the limits, then save the file.

To verify that the group quota has been set, use the command:

```
quota -g devel
```

### 14.1.6. Setting the Grace Period for Soft Limits

If a given quota has soft limits, you can edit the grace period (i.e. the amount of time a soft limit can be exceeded) with the following command:

```
edquota -t
```

This command works on quotas for inodes or blocks, for either users or groups. While other `edquota` commands operate on quotas for a particular user or group, the `-t` option operates on every file system with quotas enabled.

### 14.2. Managing Disk Quotas

If quotas are implemented, they need some maintenance — mostly in the form of watching to see if the quotas are exceeded and making sure the quotas are accurate.

Of course, if users repeatedly exceed their quotas or consistently reach their soft limits, a system administrator has a few choices to make depending on what type of users they are and how much disk space impacts their work. The administrator can either help the user determine how to use less disk space or increase the user's disk quota.

#### 14.2.1. Enabling and Disabling

It is possible to disable quotas without setting them to 0. To turn all user and group quotas off, use the following command:

```
quotaoff -vaug
```

If neither the `-u` or `-g` options are specified, only the user quotas are disabled. If only `-g` is specified, only group quotas are disabled. The `-v` switch causes verbose status information to display as the command executes.

To enable quotas again, use the `quotaon` command with the same options.

For example, to enable user and group quotas for all file systems, use the following command:

```
quotaon -vaug
```

To enable quotas for a specific file system, such as `/home`, use the following command:

```
quotaon -vug /home
```

If neither the `-u` or `-g` options are specified, only the user quotas are enabled. If only `-g` is specified, only group quotas are enabled.
### 14.2.2. Reporting on Disk Quotas

Creating a disk usage report entails running the `repquota` utility. For example, the command `repquota /home` produces this output:

```
*** Report for user quotas on device /dev/mapper/VolGroup00-LogVol02
Block grace time: 7days; Inode grace time: 7days

<table>
<thead>
<tr>
<th>User</th>
<th>used</th>
<th>soft</th>
<th>hard</th>
<th>grace</th>
<th>used</th>
<th>soft</th>
<th>hard</th>
<th>grace</th>
</tr>
</thead>
<tbody>
<tr>
<td>root</td>
<td>--</td>
<td>36</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>kristin</td>
<td>--</td>
<td>540</td>
<td>0</td>
<td>0</td>
<td>125</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>testuser</td>
<td>-- 440400</td>
<td>500000</td>
<td>550000</td>
<td>37418</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

To view the disk usage report for all (option `-a`) quota-enabled file systems, use the command:

`repquota -a`

While the report is easy to read, a few points should be explained. The `--` displayed after each user is a quick way to determine whether the block or inode limits have been exceeded. If either soft limit is exceeded, a `+` appears in place of the corresponding `-`; the first `-` represents the block limit, and the second represents the inode limit.

The `grace` columns are normally blank. If a soft limit has been exceeded, the column contains a time specification equal to the amount of time remaining on the grace period. If the grace period has expired, `none` appears in its place.

### 14.2.3. Keeping Quotas Accurate

Whenever a file system is not unmounted cleanly (due to a system crash, for example), it is necessary to run `quotacheck`. However, `quotacheck` can be run on a regular basis, even if the system has not crashed. Safe methods for periodically running `quotacheck` include:

**Ensuring quotacheck runs on next reboot**

**Best method for most systems**

This method works best for (busy) multiuser systems which are periodically rebooted.

As root, place a shell script into the `/etc/cron.daily/` or `/etc/cron.weekly/` directory—or schedule one using the `crontab -e` command—that contains the `touch /forcequotacheck` command. This creates an empty `forcequotacheck` file in the root directory, which the system init script looks for at boot time. If it is found, the init script runs `quotacheck`. Afterward, the init script removes the `/forcequotacheck` file; thus, scheduling this file to be created periodically with `cron` ensures that `quotacheck` is run during the next reboot.

For more information about `cron`, refer to `man cron`.

**Running quotacheck in single user mode**

An alternative way to safely run `quotacheck` is to (re-)boot the system into single-user mode to prevent the possibility of data corruption in quota files and run the following commands:

```
quotaoff -vaug /file_system
quotacheck -vaug /file_system
```
quotaon -vaug /file_system

Running quotacheck on a running system

If necessary, it is possible to run quotacheck on a machine during a time when no users are logged in, and thus have no open files on the file system being checked. Run the command quotacheck -vaug file_system; this command will fail if quotacheck cannot remount the given file_system as read-only. Note that, following the check, the file system will be remounted read-write.

**Warning**

Running quotacheck on a live file system mounted read-write is not recommended due to the possibility of quota file corruption.

Refer to man cron for more information about configuring cron.

### 14.3. References

For more information on disk quotas, refer to the man pages of the following commands:

- quotacheck
- edquota
- repquota
- quota
- quotaon
- quotaoff
Access Control Lists

Files and directories have permission sets for the owner of the file, the group associated with the file, and all other users for the system. However, these permission sets have limitations. For example, different permissions cannot be configured for different users. Thus, Access Control Lists (ACLs) were implemented.

The Fedora kernel provides ACL support for the ext3 file system and NFS-exported file systems. ACLs are also recognized on ext3 file systems accessed via Samba.

Along with support in the kernel, the acl package is required to implement ACLs. It contains the utilities used to add, modify, remove, and retrieve ACL information.

The cp and mv commands copy or move any ACLs associated with files and directories.

15.1. Mounting File Systems

Before using ACLs for a file or directory, the partition for the file or directory must be mounted with ACL support. If it is a local ext3 file system, it can mounted with the following command:

`mount -t ext3 -o acl device-name partition`

For example:

`mount -t ext3 -o acl /dev/VolGroup00/LogVol02 /work`

Alternatively, if the partition is listed in the `/etc/fstab` file, the entry for the partition can include the acl option:

```
LABEL=/work      /work       ext3    acl        1 2
```

If an ext3 file system is accessed via Samba and ACLs have been enabled for it, the ACLs are recognized because Samba has been compiled with the `--with-acl-support` option. No special flags are required when accessing or mounting a Samba share.

15.1.1. NFS

By default, if the file system being exported by an NFS server supports ACLs and the NFS client can read ACLs, ACLs are utilized by the client system.

To disable ACLs on NFS shares when configuring the server, include the no_acl option in the `/etc/exports` file. To disable ACLs on an NFS share when mounting it on a client, mount it with the no_acl option via the command line or the `/etc/fstab` file.

15.2. Setting Access ACLs

There are two types of ACLs: access ACLs and default ACLs. An access ACL is the access control list for a specific file or directory. A default ACL can only be associated with a directory; if a file within the directory does not have an access ACL, it uses the rules of the default ACL for the directory. Default ACLs are optional.

ACLs can be configured:

1. Per user
2. Per group
Chapter 15. Access Control Lists

3. Via the effective rights mask

4. For users not in the user group for the file

The `setfacl` utility sets ACLs for files and directories. Use the `-m` option to add or modify the ACL of a file or directory:

```
setfacl -m rules files
```

Rules (`rules`) must be specified in the following formats. Multiple rules can be specified in the same command if they are separated by commas.

- **u:uid:perms**
  - Sets the access ACL for a user. The user name or UID may be specified. The user may be any valid user on the system.

- **g:gid:perms**
  - Sets the access ACL for a group. The group name or GID may be specified. The group may be any valid group on the system.

- **m:perms**
  - Sets the effective rights mask. The mask is the union of all permissions of the owning group and all of the user and group entries.

- **o:perms**
  - Sets the access ACL for users other than the ones in the group for the file.

Permissions (`perms`) must be a combination of the characters `r`, `w`, and `x` for read, write, and execute.

If a file or directory already has an ACL, and the `setfacl` command is used, the additional rules are added to the existing ACL or the existing rule is modified.

For example, to give read and write permissions to user andrius:

```
setfacl -m u:andrius:rw /project/somefile
```

To remove all the permissions for a user, group, or others, use the `-x` option and do not specify any permissions:

```
setfacl -x rules files
```

For example, to remove all permissions from the user with UID 500:

```
setfacl -x u:500 /project/somefile
```

### 15.3. Setting Default ACLs

To set a default ACL, add `d:` before the rule and specify a directory instead of a file name.

For example, to set the default ACL for the `/share/` directory to read and execute for users not in the user group (an access ACL for an individual file can override it):

```
setfacl -m d:o:rx /share
```

### 15.4. Retrieving ACLs

To determine the existing ACLs for a file or directory, use the `getfacl` command. In the example below, the `getfacl` is used to determine the existing ACLs for a file.
getfacl home/john/picture.png

The above command returns the following output:

```
# file: home/john/picture.png
# owner: john
# group: john
user::rw-
group::r--
other::r--
```

If a directory with a default ACL is specified, the default ACL is also displayed as illustrated below. For example, `getfacl home/sales/` will display similar output:

```
# file: home/sales/
# owner: john
# group: john
user::rw-
user:barryg:r--
group::r--
mask::r--
other::r--
default:user::rwx
default:user:john:rwx
default:group::r-x
default:mask::rwx
default:other::r-x
```

### 15.5. Archiving File Systems With ACLs

By default, the `dump` command now preserves ACLs during a backup operation. When archiving a file or file system with `tar`, use the `--acls` option to preserve ACLs. Similarly, when using `cp` to copy files with ACLs, include the `--preserve=mode` option to ensure that ACLs are copied across too. In addition, the `-a` option (equivalent to `-DR --preserve=all`) of `cp` also preserves ACLs during a backup along with other information such as timestamps, SELinux contexts, and the like. For more information about `dump`, `tar`, or `cp`, refer to their respective `man` pages.

The `star` utility is similar to the `tar` utility in that it can be used to generate archives of files; however, some of its options are different. Refer to `Table 15.1, “Command Line Options for star”` for a listing of more commonly used options. For all available options, refer to `man star`. The `star` package is required to use this utility.

**Table 15.1. Command Line Options for star**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-c</code></td>
<td>Creates an archive file.</td>
</tr>
<tr>
<td><code>-n</code></td>
<td>Do not extract the files; use in conjunction with <code>-x</code> to show what extracting the files does.</td>
</tr>
<tr>
<td><code>-r</code></td>
<td>Replaces files in the archive. The files are written to the end of the archive file, replacing any files with the same path and file name.</td>
</tr>
<tr>
<td><code>-t</code></td>
<td>Displays the contents of the archive file.</td>
</tr>
<tr>
<td><code>-u</code></td>
<td>Updates the archive file. The files are written to the end of the archive if they do not exist in the archive, or if the</td>
</tr>
</tbody>
</table>
Chapter 15. Access Control Lists

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>files are newer than the files of the same name in the archive. This option only works if the archive is a file or an unblocked tape that may backspace.</td>
</tr>
<tr>
<td>-x</td>
<td>Extracts the files from the archive. If used with -U and a file in the archive is older than the corresponding file on the file system, the file is not extracted.</td>
</tr>
<tr>
<td>-help</td>
<td>Displays the most important options.</td>
</tr>
<tr>
<td>-xhelp</td>
<td>Displays the least important options.</td>
</tr>
<tr>
<td>-/</td>
<td>Do not strip leading slashes from file names when extracting the files from an archive. By default, they are stripped when files are extracted.</td>
</tr>
<tr>
<td>-acl</td>
<td>When creating or extracting, archives or restores any ACLs associated with the files and directories.</td>
</tr>
</tbody>
</table>

15.6. Compatibility with Older Systems
If an ACL has been set on any file on a given file system, that file system has the \texttt{ext\_attr} attribute. This attribute can be seen using the following command:

\texttt{tune2fs -l filesystem-device}

A file system that has acquired the \texttt{ext\_attr} attribute can be mounted with older kernels, but those kernels do not enforce any ACLs which have been set.

Versions of the \texttt{e2fsck} utility included in version 1.22 and higher of the \texttt{e2fsprogs} package (including the versions in very early versions of Fedora) can check a file system with the \texttt{ext\_attr} attribute. Older versions refuse to check it.

15.7. References
Refer to the following man pages for more information.

- man \texttt{acl} — Description of ACLs
- man \texttt{getfacl} — Discusses how to get file access control lists
- man \texttt{setfacl} — Explains how to set file access control lists
- man \texttt{star} — Explains more about the \texttt{star} utility and its many options
Write Barriers

A write barrier is a kernel mechanism used to ensure that file system metadata is correctly written and ordered on persistent storage, even when storage devices with volatile write caches lose power. File systems with write barriers enabled also ensure that data transmitted via \texttt{fsync()} is persistent throughout a power loss.

Enabling write barriers incurs a substantial performance penalty for some applications. Specifically, applications that use \texttt{fsync()} heavily or create and delete many small files will likely run much slower.

16.1. Importance of Write Barriers

File systems take great care to safely update metadata, ensuring consistency. Journalled file systems bundle metadata updates into transactions and send them to persistent storage in the following manner:

1. First, the file system sends the body of the transaction to the storage device.
2. Then, the file system sends a commit block.
3. If the transaction and its corresponding commit block are written to disk, the file system assumes that the transaction will survive any power failure.

However, file system integrity during power failure becomes more complex for storage devices with extra caches. Storage target devices like local S-ATA or SAS drives may have write caches ranging from 32MB to 64MB in size (with modern drives). Hardware RAID controllers often contain internal write caches. Further, high end arrays, like those from NetApp, IBM, Hitachi and EMC (among others), also have large caches.

Storage devices with write caches report I/O as “complete” when the data is in cache; if the cache loses power, it loses its data as well. Worse, as the cache de-stages to persistent storage, it may change the original metadata ordering. When this occurs, the commit block may be present on disk without having the complete, associated transaction in place. As a result, the journal may replay these uninitialized transaction blocks into the file system during post-power-loss recovery; this will cause data inconsistency and corruption.

How Write Barriers Work

Write barriers are implemented in the Linux kernel via storage write cache flushes before and after the I/O, which is order-critical. After the transaction is written, the storage cache is flushed, the commit block is written, and the cache is flushed again. This ensures that:

- The disk contains all the data.
- No re-ordering has occurred.

With barriers enabled, an \texttt{fsync()} call will also issue a storage cache flush. This guarantees that file data is persistent on disk even if power loss occurs shortly after \texttt{fsync()} returns.

16.2. Enabling/Disabling Write Barriers

To mitigate the risk of data corruption during power loss, some storage devices use battery-backed write caches. Generally, high-end arrays and some hardware controllers use battery-backed write
cached. However, because the cache's volatility is not visible to the kernel, Fedora 14 enables write barriers by default on all supported journaling file systems.

**Note**

Write caches are designed to increase I/O performance. However, enabling write barriers means constantly flushing these caches, which can significantly reduce performance.

For devices with non-volatile, battery-backed write caches and those with write-caching disabled, you can safely disable write barriers at mount time using the `-o nobarrier` option for `mount`. However, some devices do not support write barriers; such devices will log an error message to `/var/log/messages` (refer to Table 16.1, "Write barrier error messages per file system").

<table>
<thead>
<tr>
<th>File System</th>
<th>Error Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>ext3/ext4</td>
<td>JBD: barrier-based sync failed on device - disabling barriers</td>
</tr>
<tr>
<td>XFS</td>
<td>Filesystem device - Disabling barriers, trial barrier write failed</td>
</tr>
<tr>
<td>btrfs</td>
<td>btrfs: disabling barriers on dev device</td>
</tr>
</tbody>
</table>

### 16.3. Write Barrier Considerations

Some system configurations do not need write barriers to protect data. In most cases, other methods are preferable to write barriers, since enabling write barriers causes a significant performance penalty.

**Disabling Write Caches**

One way to alternatively avoid data integrity issues is to ensure that no write caches lose data on power failures. When possible, the best way to configure this is to simply disable the write cache. On a simple server or desktop with one or more SATA drives (off a local SATA controller Intel AHCI part), you can disable the write cache on the target SATA drives with the `hdparm` command, as in:

```
hdparm -W0 /device/
```

**Battery-Backed Write Caches**

Write barriers are also unnecessary whenever the system uses hardware RAID controllers with battery-backed write cache. If the system is equipped with such controllers and if its component drives have write caches disabled, the controller will advertise itself as a write-through cache; this will inform the kernel that the write cache data will survive a power loss.

Most controllers use vendor-specific tools to query and manipulate target drives. For example, the LSI Megaraid SAS controller uses a battery-backed write cache; this type of controller requires the `MegaCli64` tool to manage target drives. To show the state of all back-end drives for LSI Megaraid SAS, use:

```
MegaCli64 -LDGetProp -DskCache -LAll -aALL
```

To disable the write cache of all back-end drives for LSI Megaraid SAS, use:
MegaCli64 -LDSetProp -DisDskCache -Lall -aALL

**Note**

Hardware RAID cards recharge their batteries while the system is operational. If a system is powered off for an extended period of time, the batteries will lose their charge, leaving stored data vulnerable during a power failure.

**High-End Arrays**

High-end arrays have various ways of protecting data in the event of a power failure. As such, there is no need to verify the state of the internal drives in external RAID storage.

**NFS**

NFS clients do not need to enable write barriers, since data integrity is handled by the NFS server side. As such, NFS servers should be configured to ensure data persistence throughout a power loss (whether through write barriers or other means).
Storage I/O Alignment and Size

Recent enhancements to the SCSI and ATA standards allow storage devices to indicate their preferred (and in some cases, required) I/O alignment and I/O size. This information is particularly useful with newer disk drives that increase the physical sector size from 512 bytes to 4k bytes. This information may also be beneficial for RAID devices, where the chunk size and stripe size may impact performance.

The Linux I/O stack has been enhanced to process vendor-provided I/O alignment and I/O size information, allowing storage management tools (parted, lvm, mkfs.*, and the like) to optimize data placement and access. If a legacy device does not export I/O alignment and size data, then storage management tools in Fedora 14 will conservatively align I/O on a 4k (or larger power of 2) boundary. This will ensure that 4k-sector devices operate correctly even if they do not indicate any required/preferred I/O alignment and size.

Note

Fedora 14 supports 4k-sector devices as data disks, not as boot disks. Boot support for 4k-sector devices is planned for a later release.

Refer to Section 17.2, “Userspace Access” to learn how to determine the information that the operating system obtained from the device. This data is subsequently used by the storage management tools to determine data placement.

17.1. Parameters for Storage Access

The operating system uses the following information to determine I/O alignment and size:

- physical_block_size
  Smallest internal unit on which the device can operate

- logical_block_size
  Used externally to address a location on the device

- alignment_offset
  The number of bytes that the beginning of the Linux block device (partition/MD/LVM device) is offset from the underlying physical alignment

- minimum_io_size
  The device’s preferred minimum unit for random I/O

- optimal_io_size
  The device’s preferred unit for streaming I/O

For example, certain 4K sector devices may use a 4K physical_block_size internally but expose a more granular 512-byte logical_block_size to Linux. This discrepancy introduces potential for misaligned I/O. To address this, the Fedora 14 I/O stack will attempt to start all data areas on a naturally-aligned boundary (physical_block_size) by making sure it accounts for any alignment_offset if the beginning of the block device is offset from the underlying physical alignment.

Storage vendors can also supply I/O hints about the preferred minimum unit for random I/O (minimum_io_size) and streaming I/O (optimal_io_size) of a device. For example, minimum_io_size and optimal_io_size may correspond to a RAID device’s chunk size and stripe size respectively.
Chapter 17. Storage I/O Alignment and Size

17.2. Userspace Access

Always take care to use properly aligned and sized I/O. This is especially important for Direct I/O access. Direct I/O should be aligned on a \texttt{logical_block_size} boundary, and in multiples of the \texttt{logical_block_size}.

With native 4K devices (i.e. \texttt{logical_block_size} is 4K) it is now critical that applications perform direct I/O in multiples of the device’s \texttt{logical_block_size}. This means that applications will fail with native 4k devices that perform 512-byte aligned I/O rather than 4k-aligned I/O.

To avoid this, an application should consult the I/O parameters of a device to ensure it is using the proper I/O alignment and size. As mentioned earlier, I/O parameters are exposed through the both \texttt{sysfs} and block device \texttt{ioctl} interfaces.

For more details, refer to \texttt{man libblkid}. This \texttt{man} page is provided by the \texttt{libblkid-devel} package.

\textbf{sysfs Interface}

- /sys/block/disk/alignment_offset
- /sys/block/disk/partition/alignment_offset
- /sys/block/disk/queue/physical_block_size
- /sys/block/disk/queue/logical_block_size
- /sys/block/disk/queue/minimum_io_size
- /sys/block/disk/queue/optimal_io_size

The kernel will still export these \texttt{sysfs} attributes for “legacy” devices that do not provide I/O parameters information, for example:

```
  alignment_offset:  0  
  physical_block_size: 512
  logical_block_size:  512
  minimum_io_size:     512
  optimal_io_size:     0
```

\textbf{Block Device ioctls}

- \texttt{BLKALIGNOFF: alignment_offset}
- \texttt{BLKPBSZGET: physical_block_size}
- \texttt{BLKSSZGET: logical_block_size}
- \texttt{BLKIOMIN: minimum_io_size}
- \texttt{BLKIOOPT: optimal_io_size}

17.3. Standards

This section describes I/O standards used by ATA and SCSI devices.
ATA

ATA devices must report appropriate information via the IDENTIFY DEVICE command. ATA devices only report I/O parameters for physical_block_size, logical_block_size, and alignment_offset. The additional I/O hints are outside the scope of the ATA Command Set.

SCSI

I/O parameters support in Fedora 14 requires at least version 3 of the SCSI Primary Commands (SPC-3) protocol. The kernel will only send an extended inquiry (which gains access to the BLOCK LIMITS VPD page) and READ CAPACITY(16) command to devices which claim compliance with SPC-3.

The READ CAPACITY(16) command provides the block sizes and alignment offset:

- **LOGICAL BLOCK LENGTH IN BYTES** is used to derive /sys/block/disk/queue/physical_block_size
- **LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT** is used to derive /sys/block/disk/queue/logical_block_size
- **LOWEST ALIGNED LOGICAL BLOCK ADDRESS** is used to derive:
  - /sys/block/disk/alignment_offset
  - /sys/block/disk/partition/alignment_offset

The BLOCK LIMITS VPD page (0xb0) provides the I/O hints. It also uses OPTIMAL TRANSFER LENGTH GRANULARITY and OPTIMAL TRANSFER LENGTH to derive:

- /sys/block/disk/queue/minimum_io_size
- /sys/block/disk/queue/optimal_io_size

The sg3_utils package provides the sg_inq utility, which can be used to access the BLOCK LIMITS VPD page. To do so, run:

```
sg_inq -p 0xb0 disk
```

17.4. Stacking I/O Parameters

All layers of the Linux I/O stack have been engineered to propagate the various I/O parameters up the stack. When a layer consumes an attribute or aggregates many devices, the layer must expose appropriate I/O parameters so that upper-layer devices or tools will have an accurate view of the storage as it transformed. Some practical examples are:

- Only one layer in the I/O stack should adjust for a non-zero alignment_offset; once a layer adjusts accordingly, it will export a device with an alignment_offset of zero.

- A striped Device Mapper (DM) device created with LVM must export a minimum_io_size and optimal_io_size relative to the stripe count (number of disks) and user-provided chunk size.

In Fedora 14, Device Mapper and Software Raid (MD) device drivers can be used to arbitrarily combine devices with different I/O parameters. The kernel's block layer will attempt to reasonably combine the I/O parameters of the individual devices. The kernel will not prevent combining heterogenous devices; however, be aware of the risks associated with doing so.
For instance, a 512-byte device and a 4K device may be combined into a single logical DM device, which would have a logical_block_size of 4K. File systems layered on such a hybrid device assume that 4K will be written atomically, but in reality it will span 8 logical block addresses when issued to the 512-byte device. Using a 4K logical_block_size for the higher-level DM device increases potential for a partial write to the 512-byte device if there is a system crash.

If combining the I/O parameters of multiple devices results in a conflict, the block layer may issue a warning that the device is susceptible to partial writes and/or is misaligned.

17.5. Logical Volume Manager

LVM provides userspace tools that are used to manage the kernel's DM devices. LVM will shift the start of the data area (that a given DM device will use) to account for a non-zero alignment_offset associated with any device managed by LVM. This means logical volumes will be properly aligned (alignment_offset=0).

By default, LVM will adjust for any alignment_offset, but this behavior can be disabled by setting data_alignment_offset_detection to 0 in /etc/lvm/lvm.conf. Disabling this is not recommended.

LVM will also detect the I/O hints for a device. The start of a device's data area will be a multiple of the minimum_io_size or optimal_io_size exposed in sysfs. LVM will use the minimum_io_size if optimal_io_size is undefined (i.e. 0).

By default, LVM will automatically determine these I/O hints, but this behavior can be disabled by setting data_alignment_detection to 0 in /etc/lvm/lvm.conf. Disabling this is not recommended.

17.6. Partition and File System Tools

This section describes how different partition and file system management tools interact with a device's I/O parameters.

util-linux-ng's libblkid and fdisk

The libblkid library provided with the util-linux-ng package includes a programmatic API to access a device's I/O parameters. libblkid allows applications, especially those that use Direct I/O, to properly size their I/O requests. The fdisk utility from util-linux-ng uses libblkid to determine the I/O parameters of a device for optimal placement of all partitions. The fdisk utility will align all partitions on a 1MB boundary.

parted and libparted

The libparted library from parted also uses the I/O parameters API of libblkid. The Fedora installer (Anaconda) uses libparted, which means that all partitions created by either the installer or parted will be properly aligned. For all partitions created on a device that does not appear to provide I/O parameters, the default alignment will be 1MB.

The heuristics parted uses are as follows:

- Always use the reported alignment_offset as the offset for the start of the first primary partition.
- If optimal_io_size is defined (i.e. not 0), align all partitions on an optimal_io_size boundary.
- If optimal_io_size is undefined (i.e. 0), alignment_offset is 0, and minimum_io_size is a power of 2, use a 1MB default alignment.
This is the catch-all for "legacy" devices which don't appear to provide I/O hints. As such, by default all partitions will be aligned on a 1MB boundary.

**Note**

Fedora cannot distinguish between devices that don't provide I/O hints and those that do so with `alignment_offset=0` and `optimal_io_size=0`. Such a device might be a single SAS 4K device; as such, at worst 1MB of space is lost at the start of the disk.

**File System tools**

The different `mkfs.filesystem` utilities have also been enhanced to consume a device's I/O parameters. These utilities will not allow a file system to be formatted to use a block size smaller than the `logical_block_size` of the underlying storage device.

Except for `mkfs.gfs2`, all other `mkfs.filesystem` utilities also use the I/O hints to layout on-disk data structure and data areas relative to the `minimum_io_size` and `optimal_io_size` of the underlying storage device. This allows file systems to be optimally formatted for various RAID (striped) layouts.
Chapter 18.

Setting Up A Remote Diskless System

The Network Booting Service (provided by system-config-netboot) is no longer available in Fedora 14. Deploying diskless systems is now possible in this release without the use of system-config-netboot.

To set up a basic remote diskless system booted over PXE, you need the following packages:

- tftp-server
- xinetd
- dhcp
- syslinux
- dracut-network

Remote diskless system booting requires both a tftp service (provided by tftp-server) and a DHCP service (provided by dhcp). The tftp service is used to retrieve kernel image and initrd over the network via the PXE loader. Both tftp and DHCP services must be provided by the same host machine.

The following sections outline the necessary procedures for deploying remote diskless systems in a network environment.

18.1. Configuring a tftp Service for Diskless Clients

The tftp service is disabled by default. To enable it and allow PXE booting via the network, set the Disabled option in /etc/xinetd.d/tftp to no. To configure tftp, perform the following steps:

1. The tftp root directory (chroot) is located in /var/lib/tftpboot. Copy /usr/share/syslinux/pxelinux.0 to /var/lib/tftpboot/, as in:

   cp /usr/share/syslinux/pxelinux.0 /var/lib/tftpboot/

2. Create a pxelinux.cfg directory inside the tftp root directory:

   mkdir -p /var/lib/tftpboot/pxelinux.cfg/

You will also need to configure firewall rules properly to allow tftp traffic; as tftp supports TCP wrappers, you can configure host access to tftp via /etc/hosts.allow. For more information on configuring TCP wrappers and the /etc/hosts.allow configuration file, refer to the Security Guide for Fedora 14 or Red Hat Enterprise Linux 6; man hosts_access also provides information about /etc/hosts.allow.

After configuring tftp for diskless clients, configure DHCP, NFS, and the exported file system accordingly. Refer to Section 18.2, “Configuring DHCP for Diskless Clients” and Section 18.3, “Configuring an Exported File System for Diskless Clients” for instructions on how to do so.

18.2. Configuring DHCP for Diskless Clients

After configuring a tftp server, you need to set up a DHCP service on the same host machine. Refer to the Fedora 14 Deployment Guide for instructions on how to set up a DHCP server. In addition, you should enable PXE booting on the DHCP server; to do this, add the following configuration to /etc/dhcp/dhcp.conf:
allow booting;
allow bootp;
class "pxeclients" {
    match if substring(option vendor-class-identifier, 0, 9) = "PXEClien";
    next-server server-ip;
    filename "linux-install/pxelinux.0";
}

Replace `server-ip` with the IP address of the host machine on which the `tftp` and DHCP services reside. Now that `tftp` and DHCP are configured, all that remains is to configure NFS and the exported file system; refer to Section 18.3, “Configuring an Exported File System for Diskless Clients” for instructions.

### 18.3. Configuring an Exported File System for Diskless Clients

The root directory of the exported file system (used by diskless clients in the network) is shared via NFS. Configure the NFS service to export the root directory by adding it to `/etc/exports`. For instructions on how to do so, refer to Section 9.6.1, “The `/etc/exports` Configuration File”.

To accommodate completely diskless clients, the root directory should contain a complete Fedora 14 installation. You can synchronize this with a running system via `rsync`, as in:

```
rsync -a -e ssh --exclude='/proc/*' --exclude='/sys/*' hostname.com:/ /exported/root/directory
```

Replace `hostname.com` with the hostname of the running system with which to synchronize via `rsync`. The `/exported/root/directory` is the path to the exported file system.

Alternatively, you can also use `yum` with the `--installroot` option to install Fedora to a specific location. For example:

```
yum groupinstall Base --installroot=/exported/root/directory
```

The file system to be exported still needs to be configured further before it can be used by diskless clients. To do this, perform the following procedure:

1. Configure the exported file system’s `/etc/fstab` to contain (at least) the following configuration:

   ```
   none /tmp tmpfs defaults 0 0
tmpfs /dev/shm tmpfs defaults 0 0
sysfs /sys sysfs defaults 0 0
proc /proc proc defaults 0 0
   ```

2. Select the kernel that diskless clients should use (`vmlinuz-kernel-version`) and copy it to the `tftp` boot directory:

   ```
   cp /boot/vmlinuz-kernel-version /var/lib/tftpboot/
   ```

3. Create the `initrd` (i.e. `initramfs-kernel-version.img`) with network support:

   ```
   dracut initramfs-kernel-version.img vmlinuz-kernel-version
   ```

   Copy the resulting `initramfs-kernel-version.img` into the `tftp` boot directory as well.

4. Edit the default boot configuration to use the `initrd` and kernel inside `/var/lib/tftpboot`. This configuration should instruct the diskless client’s root to mount the exported file system (`/`)
exported/root directory) as read-write. To do this, configure /var/lib/tftpboot/pxelinux.cfg/default with the following:

```
default rhel6
label rhel6
  kernel vmlinuz-kernel-version
  append initrd=initramfs-kernel-version.img root=nfs:server-ip:/exported/root/directory rw
```

Replace server-ip with the IP address of the host machine on which the tftp and DHCP services reside.

The NFS share is now ready for exporting to diskless clients. These clients can boot over the network via PXE.
Solid-State Disk Deployment Guidelines

Solid-state disks (SSD) are storage devices that use NAND flash chips to persistently store data. This sets them apart from previous generations of disks, which store data in rotating, magnetic platters. In an SSD, the access time for data across the full Logical Block Address (LBA) range is constant; whereas with older disks that use rotating media, access patterns that span large address ranges incur seek costs. As such, SSD devices have better latency and throughput.

Not all SSDs show the same performance profiles, however. In fact, many of the first generation devices show little or no advantage over spinning media. Thus, it is important to define classes of solid state storage to frame further discussion in this section.

SSDs can be divided into three classes, based on throughput:

- The first class of SSDs use a PCI-Express connection, which offers the fastest I/O throughput compared to other classes. This class also has a very low latency for random access.

- The second class uses the traditional SATA connection, and features fast random access for read and write operations (though not as fast as SSDs that use PCI-Express connection).

- The third class also uses SATA, but the performance of SSDs in this class do not differ substantially from devices that use 7200rpm rotational disks.

For all three classes, performance degrades as the number of used blocks approaches the disk capacity. The degree of performance impact varies greatly by vendor. However, all devices experience some degradation.

To address the degradation issue, the ATA specification outlines a new command: TRIM. This command allows the file system to communicate to the underlying storage device that a given range of blocks is no longer in use. The SSD can use this information to free up space internally, using the freed blocks for wear-leveling.

Enabling TRIM support is most useful when there is available free space on the file system, but the file system has already written to most logical blocks on the underlying storage device. For more information about TRIM, refer to its Data Set Management T13 Specifications from the following link:


Note

Not all solid-state devices in the market support TRIM.

19.1. Deployment Considerations

Because of the internal layout and operation of SSDs, it is best to partition devices on an internal erase block boundary. Partitioning utilities in Fedora 14 chooses sane defaults if the SSD exports topology information. This is especially true if the exported topology information includes alignment offsets and optimal I/O sizes.

However, if the device does not export topology information, you should that the first partition be created at a 1MB boundary.
In addition, keep in mind that logical volumes, device-mapper targets, and md targets do not support TRIM. As such, the default Fedora 14 installation will not allow the use of the TRIM command, since this install uses DM-linear targets.

Take note as well that software RAID levels 1, 4, 5, and 6 are not recommended for use on SSDs. During the initialization stage of these RAID levels, some RAID management utilities (such as mdadm) write to all of the blocks on the storage device to ensure that checksums operate properly. This will cause the performance of the SSD to degrade quickly.

At present, ext4 is the only fully-supported file system that supports TRIM. To enable TRIM commands on a device, use the mount option discard. For example, to mount /dev/sda2 to /mnt with TRIM enabled, run:

```
mount -t ext4 -o discard /dev/sda2 /mnt
```

By default, ext4 does not issue the TRIM command. This is mostly to avoid problems on devices which may not properly implement the TRIM command. The Linux swap code will issue TRIM commands to TRIM-enabled devices, and there is no option to control this behaviour.

## 19.2. Tuning Considerations

This section describes several factors to consider when configuring settings that may affect SSD performance.

### I/O Scheduler

Any I/O scheduler should perform well with most SSDs. However, as with any other storage type, you should benchmark to determine the optimal configuration for a given workload.

When using SSDs, you should change the I/O scheduler only for benchmarking particular workloads. For more information about the different types of I/O schedulers, refer to the Red Hat Enterprise Linux I/O Tuning Guide. The following kernel document also contains instructions on how to switch between I/O schedulers:

```
/usr/share/doc/kernel-version/Documentation/block/switching-sched.txt
```

### Virtual Memory

Like the I/O scheduler, virtual memory (VM) subsystem requires no special tuning. Given the fast nature of I/O on SSD, it should be possible to turn down the vm_dirty_background_ratio and vm_dirty_ratio settings, as increased write-out activity should not negatively impact the latency of other operations on the disk. However, this can generate more overall I/O and so is not generally recommended without workload-specific testing.

### Swap

An SSD can also be used as a swap device, and is likely to produce good page-out/page-in performance.
Online Storage Management

It is often desirable to add, remove or re-size storage devices while the operating system is running, and without rebooting. This chapter outlines the procedures that may be used to reconfigure storage devices on Fedora 14 host systems while the system is running. It covers iSCSI and Fibre Channel storage interconnects; other interconnect types may be added in the future.

This chapter focuses on adding, removing, modifying, and monitoring storage devices. It does not discuss the Fibre Channel or iSCSI protocols in detail. For more information about these protocols, refer to other documentation.

This chapter makes reference to various sysfs objects. Fedora advises that the sysfs object names and directory structure are subject to change in major Fedora releases. This is because the upstream Linux kernel does not provide a stable internal API. For guidelines on how to reference sysfs objects in a transportable way, refer to the document Documentation/sysfs-rules.txt in the kernel source tree for guidelines.

**Warning**

Online storage reconfiguration must be done carefully. System failures or interruptions during the process can lead to unexpected results. You should reduce system load to the maximum extent possible during the change operations. This will reduce the chance of I/O errors, out-of-memory errors, or similar errors occurring in the midst of a configuration change. The following sections provide more specific guidelines regarding this.

In addition, you should back up all data before reconfiguring online storage.

### 20.1. Fibre Channel

This section discusses the Fibre Channel API, native Fedora 14 Fibre Channel drivers, and the Fibre Channel capabilities of these drivers.

#### 20.1.1. Fibre Channel API

Below is a list of /sys/class/ directories that contain files used to provide the userspace API. In each item, host numbers are designated by $H$, bus numbers are $B$, targets are $T$, logical unit numbers (LUNs) are $L$, and remote port numbers are $R$.

**Important**

If your system is using multipath software, consult your hardware vendor before changing any of the values described in this section.

Transport: /sys/class/fc_transport/target$H:B:T/
- $port_id$ — 24-bit port ID/address
- $node_name$ — 64-bit node name
- $port_name$ — 64-bit port name

Remote Port: /sys/class/fc_remote_ports/rport-$H:B-R/
- $port_id$
Chapter 20. Online Storage Management

- *node_name*
- *port_name*
- *dev_loss_tmo* — number of seconds to wait before marking a link as “bad”. Once a link is marked bad, I/O running on its corresponding path (along with any new I/O on that path) will be failed.

The default *dev_loss_tmo* value varies, depending on which driver/device is used. If a Qlogic adapter is used, the default is 35 seconds, while if an Emulex adapter is used, it is 30 seconds. The *dev_loss_tmo* value can be changed via the `scsi_transport_fc` module parameter `dev_loss_tmo`, although the driver can override this timeout value.

The maximum *dev_loss_tmo* value is 600 seconds. If *dev_loss_tmo* is set to zero or any value greater than 600, the driver's internal timeouts will be used instead.

- *fast_io_fail_tmo* — length of time to wait before failing I/O executed when a link problem is detected. I/O that reaches the driver will fail. If I/O is in a blocked queue, it will not be failed until *dev_loss_tmo* expires and the queue is unblocked.

Host: `/sys/class/fc_host/hostH`
- *port_id*
- *issue_lip* — instructs the driver to rediscover remote ports.

### 20.1.2. Native Fibre Channel Drivers and Capabilities

Fedora 14 ships with the following native fibre channel drivers:

- *lpfc*
- *qla2xxx*
- *zfcp*
- *mptfc*

*Table 20.1, “Fibre-Channel API Capabilities”* describes the different fibre-channel API capabilities of each native Fedora 14 driver. X denotes support for the capability.

<table>
<thead>
<tr>
<th></th>
<th>lpfc</th>
<th>qla2xxx</th>
<th>zfcp</th>
<th>mptfc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transport</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>port_id</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Transport</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>node_name</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Transport</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>port_name</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Remote Port</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dev_loss_tmo</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Remote Port</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fast_io_fail_tmo</td>
<td>X</td>
<td>X(^1)</td>
<td>X(^2)</td>
<td></td>
</tr>
<tr>
<td>Host</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>port_id</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
20.2. iSCSI

This section describes the iSCSI API and the `iscsiadm` utility. Before using the `iscsiadm` utility, install the `iscsi-initiator-utils` package first; to do so, run `yum install iscsi-initiator-utils`.

In addition, the iSCSI service must be running in order to discover or log in to targets. To start the iSCSI service, run `service iscsi start`.

20.2.1. iSCSI API

To get information about running sessions, run:

```bash
iscsiadm -m session -P 3
```

This command displays the session/device state, session ID (sid), some negotiated parameters, and the SCSI devices accessible through the session.

For shorter output (for example, to display only the sid-to-node mapping), run:

```bash
iscsiadm -m session -P 0
```

or

```bash
iscsiadm -m session
```

These commands print the list of running sessions with the format:

```
driver [sid] target_ip:port,target_portal_group_tag proper_target_name
```

For example:

```bash
iscsiadm -m session
```

```
```

For more information about the iSCSI API, refer to `/usr/share/doc/iscsi-initiator-utils-version/README`.

20.3. Persistent Naming

The operating system issues I/O to a storage device by referencing the path that is used to reach it. For SCSI devices, the path consists of the following:

- PCI identifier of the host bus adapter (HBA)
- channel number on that HBA
- the remote SCSI target address
Chapter 20. Online Storage Management

- the Logical Unit Number (LUN)

This path-based address is not persistent. It may change any time the system is reconfigured (either by on-line reconfiguration, as described in this manual, or when the system is shutdown, reconfigured, and rebooted). It is even possible for the path identifiers to change when no physical reconfiguration has been done, as a result of timing variations during the discovery process when the system boots, or when a bus is re-scanned.

The operating system provides several non-persistent names to represent these access paths to storage devices. One is the /dev/sd name; another is the major:minor number. A third is a symlink maintained in the /dev/disk/by-path/ directory. This symlink maps from the path identifier to the current /dev/sd name. For example, for a Fibre Channel device, the PCI info and Host:BusTarget:LUN info may appear as follows:

```
pci-0000:02:0e.0-scsi-0:0:0:0 -> ../../sda
```

For iSCSI devices, by-path/ names map from the target name and portal information to the sd name.

It is generally not appropriate for applications to use these path-based names. This is because the storage device these paths reference may change, potentially causing incorrect data to be written to the device. Path-based names are also not appropriate for multipath devices, because the path-based names may be mistaken for separate storage devices, leading to uncoordinated access and unintended modifications of the data.

In addition, path-based names are system-specific. This can cause unintended data changes when the device is accessed by multiple systems, such as in a cluster.

For these reasons, several persistent, system-independent, methods for identifying devices have been developed. The following sections discuss these in detail.

### 20.3.1. WWID

The World Wide Identifier (WWID) can be used in reliably identifying devices. It is a persistent, system-independent ID that the SCSI Standard requires from all SCSI devices. The WWID identifier is guaranteed to be unique for every storage device, and independent of the path that is used to access the device.

This identifier can be obtained by issuing a SCSI Inquiry to retrieve the Device Identification Vital Product Data (page 0x83) or Unit Serial Number (page 0x80). The mappings from these WWIDs to the current /dev/sd names can be seen in the symlinks maintained in the /dev/disk/by-id/ directory.

For example, a device with a page 0x83 identifier would have:

```
scsi-3600508b400185e210000900000490000 -> ../../sda
```

Or, a device with a page 0x80 identifier would have:

```
scsi-SESEAGATE_ST373453LW_3HW1RHM6 -> ../../sda
```

Fedora automatically maintains the proper mapping from the WWID-based device name to a current /dev/sd name on that system. Applications can use the /dev/disk/by-id/ name to reference the data on the disk, even if the path to the device changes, and even when accessing the device from different systems.
If there are multiple paths from a system to a device, device-mapper-multipath uses the WWID to detect this. Device-mapper-multipath then presents a single “pseudo-device” in /dev/mapper/wwid, such as /dev/mapper/3600508b400105df70000e0000ac0000.

The command multipath -l shows the mapping to the non-persistent identifiers: Host:Channel:Target:LUN, /dev/sd name, and the major:minor number.

<table>
<thead>
<tr>
<th>WWID</th>
<th>Vendor, Product</th>
<th>Size</th>
<th>Features</th>
<th>HwHandler</th>
<th>RW</th>
</tr>
</thead>
<tbody>
<tr>
<td>dm-2</td>
<td></td>
<td>20G</td>
<td>1 queue_if_no_path</td>
<td>0</td>
<td>rw</td>
</tr>
<tr>
<td>5:0:1:1 sdc 8:32</td>
<td>active</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6:0:1:1 sdg 8:96</td>
<td>active</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5:0:0:1 sdb 8:16</td>
<td>active</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6:0:0:1 sdf 8:80</td>
<td>active</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Device-mapper-multipath automatically maintains the proper mapping of each WWID-based device name to its corresponding /dev/sd name on the system. These names are persistent across path changes, and they are consistent when accessing the device from different systems.

When the user_friendly_names feature (of device-mapper-multipath) is used, the WWID is mapped to a name of the form /dev/mapper/mpathn. By default, this mapping is maintained in the file /var/lib/multipath/bindings. These mpathn names are persistent as long as that file is maintained.

**Warning**

The multipath bindings file (by default, /var/lib/multipath/bindings) must be available at boot time. If /var is a separate file system from /, then you must change the default location of the file. For more information, refer to [http://kbase.redhat.com/faq/docs/DOC-17650](http://kbase.redhat.com/faq/docs/DOC-17650).

**Important**

If you use user_friendly_names, then additional steps are required to obtain consistent names in a cluster. Refer to the Consistent Multipath Device Names section in the Using Device-Mapper Multipath book.

In addition to these persistent names provided by the system, you can also use udev rules to implement persistent names of your own, mapped to the WWID of the storage. For more information about this, refer to [http://kbase.redhat.com/faq/docs/DOC-7319](http://kbase.redhat.com/faq/docs/DOC-7319).

### 20.3.2. UUID and Other Persistent Identifiers

If a storage device contains a file system, then that file system may provide one or both of the following:

- **Universally Unique Identifier (UUID)**

Chapter 20. Online Storage Management

- File system label

These identifiers are persistent, and based on metadata written on the device by certain applications. They may also be used to access the device using the symlinks maintained by the operating system in the /dev/disk/by-label/ (e.g. boot -> ../../sda1) and /dev/disk/by-uuid/ (e.g. f8bf09e3-4c16-4d91-bd5e-6f62da165c08 -> ../../sda1) directories.

md and LVM write metadata on the storage device, and read that data when they scan devices. In each case, the metadata contains a UUID, so that the device can be identified regardless of the path (or system) used to access it. As a result, the device names presented by these facilities are persistent, as long as the metadata remains unchanged.

20.4. Removing a Storage Device

Before removing access to the storage device itself, it is advisable to back up data from the device first. Afterwards, flush I/O and remove all operating system references to the device (as described below). If the device uses multipathing, then do this for the multipath “pseudo device” (Section 20.3.1, “WWID”) and each of the identifiers that represent a path to the device. If you are only removing a path to a multipath device, and other paths will remain, then the procedure is simpler, as described in Section 20.6, “Adding a Storage Device or Path”.

Removal of a storage device is not recommended when the system is under memory pressure, since the I/O flush will add to the load. To determine the level of memory pressure, run the command vmstat 1 100; device removal is not recommended if:

- Free memory is less than 5% of the total memory in more than 10 samples per 100 (the command free can also be used to display the total memory).
- Swapping is active (non-zero si and so columns in the vmstat output).

The general procedure for removing all access to a device is as follows:

Procedure 20.1. Ensuring a Clean Device Removal
1. Close all users of the device and backup device data as needed.
2. Use umount to unmount any file systems that mounted the device.
3. Remove the device from any md and LVM volume using it. If the device is a member of an LVM Volume group, then it may be necessary to move data off the device using the pvmove command, then use the vgreduce command to remove the physical volume, and (optionally) pvremove to remove the LVM metadata from the disk.
4. If the device uses multipathing, run multipath -l and note all the paths to the device. Afterwards, remove the multipathed device using multipath -f device.
5. Run blockdev -flushbufs device to flush any outstanding I/O to all paths to the device. This is particularly important for raw devices, where there is no umount or vgreduce operation to cause an I/O flush.
6. Remove any reference to the device’s path-based name, like /dev/sd, /dev/disk/by-path or the major:minor number, in applications, scripts, or utilities on the system. This is important in ensuring that different devices added in the future will not be mistaken for the current device.
7. Finally, remove each path to the device from the SCSI subsystem. To do so, use the command echo 1 > /sys/block/device-name/device/delete where device-name may be sde, for example.
Another variation of this operation is `echo 1 > /sys/class/scsi_device/h:c:t:l/device/delete`, where `h` is the HBA number, `c` is the channel on the HBA, `t` is the SCSI target ID, and `l` is the LUN.

**Note**

The older form of these commands, `echo "scsi remove-single-device 0 0 0 0" > /proc/scsi/scsi`, is deprecated.

You can determine the `device-name`, HBA number, HBA channel, SCSI target ID and LUN for a device from various commands, such as `lsscsi`, `scsi_id`, `multipath -l`, and `ls -l /dev/disk/by-*`.

After performing Procedure 20.1, "Ensuring a Clean Device Removal", a device can be physically removed safely from a running system. It is not necessary to stop I/O to other devices while doing so.

Other procedures, such as the physical removal of the device, followed by a rescan of the SCSI bus (as described in Section 20.8, "Scanning Storage Interconnects") to cause the operating system state to be updated to reflect the change, are not recommended. This will cause delays due to I/O timeouts, and devices may be removed unexpectedly. If it is necessary to perform a rescan of an interconnect, it must be done while I/O is paused, as described in Section 20.8, "Scanning Storage Interconnects".

### 20.5. Removing a Path to a Storage Device

If you are removing a path to a device that uses multipathing (without affecting other paths to the device), then the general procedure is as follows:

**Procedure 20.2. Removing a Path to a Storage Device**

1. Remove any reference to the device's path-based name, like `/dev/sd` or `/dev/disk/by-path` or the `major:minor` number, in applications, scripts, or utilities on the system. This is important in ensuring that different devices added in the future will not be mistaken for the current device.

2. Take the path offline using `echo offline > /sys/block/sda/device/state`. This will cause any subsequent I/O sent to the device on this path to be failed immediately. `Device-mapper-multipath` will continue to use the remaining paths to the device.

3. Remove the path from the SCSI subsystem. To do so, use the command `echo 1 > /sys/block/device-name/device/delete` where `device-name` may be `sde`, for example (as described in Procedure 20.1, "Ensuring a Clean Device Removal").

After performing Procedure 20.2, "Removing a Path to a Storage Device", the path can be safely removed from the running system. It is not necessary to stop I/O while this is done, as `device-mapper-multipath` will re-route I/O to remaining paths according to the configured path grouping and failover policies.

Other procedures, such as the physical removal of the cable, followed by a rescan of the SCSI bus to cause the operating system state to be updated to reflect the change, are not recommended. This will cause delays due to I/O timeouts, and devices may be removed unexpectedly. If it is necessary to perform a rescan of an interconnect, it must be done while I/O is paused, as described in Section 20.8, "Scanning Storage Interconnects".
20.6. Adding a Storage Device or Path

When adding a device, be aware that the path-based device name (/dev/sd name, major:minor number, and /dev/disk/by-path name, for example) the system assigns to the new device may have been previously in use by a device that has since been removed. As such, ensure that all old references to the path-based device name have been removed. Otherwise, the new device may be mistaken for the old device.

The first step in adding a storage device or path is to physically enable access to the new storage device, or a new path to an existing device. This is done using vendor-specific commands at the Fibre Channel or iSCSI storage server. When doing so, note the LUN value for the new storage that will be presented to your host. If the storage server is Fibre Channel, also take note of the World Wide Node Name (WWNN) of the storage server, and determine whether there is a single WWNN for all ports on the storage server. If this is not the case, note the World Wide Port Name (WWPN) for each port that will be used to access the new LUN.

Next, make the operating system aware of the new storage device, or path to an existing device. The recommended command to use is:

```
echo "c t l" > /sys/class/scsi_host/host/h/scan
```

In the previous command, \( h \) is the HBA number, \( c \) is the channel on the HBA, \( t \) is the SCSI target ID, and \( l \) is the LUN.

**Note**

The older form of this command, `echo "scsi add-single-device 0 0 0 0" > /proc/scsi/scsi`, is deprecated.

For Fibre Channel storage servers that implement a single WWNN for all ports, you can determine the correct \( h, c, \)and \( t \) values (i.e. HBA number, HBA channel, and SCSI target ID) by searching for the WWNN in `sysfs`. For example, if the WWNN of the storage server is `0x5006016090203181`, use:

```
grep 5006016090203181 /sys/class/fc_transport/*/node_name
```

This should display output similar to the following:

```
/sys/class/fc_transport/target5:0:2/node_name:0x5006016090203181
/sys/class/fc_transport/target5:0:3/node_name:0x5006016090203181
/sys/class/fc_transport/target6:0:2/node_name:0x5006016090203181
/sys/class/fc_transport/target6:0:3/node_name:0x5006016090203181
```

This indicates there are four Fibre Channel routes to this target (two single-channel HBAs, each leading to two storage ports). Assuming a LUN value is \( 56 \), then the following command will configure the first path:

```
echo "0 2 56" > /sys/class/scsi_host/host5/scan
```

This must be done for each path to the new device.

For Fibre Channel storage servers that do not implement a single WWNN for all ports, you can determine the correct HBA number, HBA channel, and SCSI target ID by searching for each of the WWPNs in `sysfs`.

Another way to determine the HBA number, HBA channel, and SCSI target ID is to refer to another device that is already configured on the same path as the new device. This can be done with various commands, such as `lsscsi`, `scsi_id`, `multipath -l`, and `ls -l /dev/disk/by-*`. This
information, plus the LUN number of the new device, can be used as shown above to probe and configure that path to the new device.

After adding all the SCSI paths to the device, execute the `multipath` command, and check to see that the device has been properly configured. At this point, the device can be added to `md`, `LVM`, `mkfs`, or `mount`, for example.

If the steps above are followed, then a device can safely be added to a running system. It is not necessary to stop I/O to other devices while this is done. Other procedures involving a rescan (or a reset) of the SCSI bus, which cause the operating system to update its state to reflect the current device connectivity, are not recommended while storage I/O is in progress.

### 20.7. Configuring a Fibre-Channel Over Ethernet Interface

Setting up and deploying a Fibre-channel over ethernet (FCoE) interface requires two packages:

- `fcoe-utils`
- `dcbd`

Once these packages are installed, perform the following procedure to enable FCoE over a virtual LAN (VLAN):

**Procedure 20.3. Configuring an ethernet interface to use FCoE**

1. Configure a new VLAN (101) by creating a new network script for it. The easiest way to do so is to copy the network script of an ethernet interface (`eth3`) to a new one with the `101` file name suffix, as in:

   ```
   cp /etc/sysconfig/network-scripts/ifcfg-eth3 /etc/sysconfig/network-scripts/ifcfg-eth3.101
   ```

2. Open `/etc/sysconfig/network-scripts/ifcfg-eth3.101`. Edit it to ensure that the following settings are configured accordingly:

   ```
   DEVICE=eth3.101
   VLAN=yes
   ONBOOT=yes
   ```

3. Start the data center bridging daemon (`dcbd`) using the following command:

   ```
   /etc/init.d/dcbd start
   ```

4. Use the `dcbtool` utility to enable data center bridging and FCoE on the ethernet interface using the following commands:

   ```
   dcbtool sc eth3 dcb on
   dcbtool sc eth3 app:fcoe e:1
   ```

   These commands will only work if no other changes have been made to the `dcbd` settings for the ethernet interface.

5. Start FCoE using the command `/etc/init.d/fcoe start`. The fibre-channel device should appear shortly, assuming all other settings on the fabric are correct.

   After correctly configuring the ethernet interface to use FCoE, you should set FCoE and `dcbd` to run at startup. To do so, use `chkconfig`, as in:
chkconfig dcbd on
chkconfig fcoe on

**Warning**

Do not run software-based DCB or LLDP on CNAs that implement DCB.

Some Combined Network Adapters (CNAs) implement the Data Center Bridging (DCB) protocol in firmware. The DCB protocol assumes that there is just one originator of DCB on a particular network link. This means that any higher-level software implementation of DCB, or Link Layer Discovery Protocol (LLDP), must be disabled on CNAs that implement DCB.

### 20.8. Scanning Storage Interconnects

There are several commands available that allow you to reset and/or scan one or more interconnects, potentially adding and removing multiple devices in one operation. This type of scan can be disruptive, as it can cause delays while I/O operations timeout, and remove devices unexpectedly. As such, you should only use this type of scan *only when necessary*. In addition, the following restrictions must be observed when scanning storage interconnects:

1. All I/O on the effected interconnects must be paused and flushed before executing the procedure, and the results of the scan checked before I/O is resumed.

2. As with removing a device, interconnect scanning is not recommended when the system is under memory pressure. To determine the level of memory pressure, run the command `vmstat 1 100`; interconnect scanning is not recommended if free memory is less than 5% of the total memory in more than 10 samples per 100. It is also not recommended if swapping is active (non-zero `si` and `so` columns in the `vmstat` output). The command `free` can also display the total memory.

The following commands can be used to scan storage interconnects.

**echo "1" > /sys/class/fc_host/host/issue_lip**

This operation performs a Loop Initialization Protocol (LIP) and then scans the interconnect and causes the SCSI layer to be updated to reflect the devices currently on the bus. A LIP is, essentially, a bus reset, and will cause device addition and removal. This procedure is necessary to configure a new SCSI target on a Fibre Channel interconnect.

Bear in mind that `issue_lip` is an asynchronous operation. The command may complete before the entire scan has completed. You must monitor `/var/log/messages` to determine when it is done.

The `lpfc` and `qla2xxx` drivers support `issue_lip`. For more information about the API capabilities supported by each driver in Fedora 14, refer to Table 20.1, “Fibre-Channel API Capabilities”.

**/usr/bin/rescan-scsi-bus.sh**

By default, this script scans all the SCSI buses on the system, updating the SCSI layer to reflect new devices on the bus. The script provides additional options to allow device removal and the issuing of LIPs. For more information about this script (including known issues), refer to Section 20.14, “Adding/Removing a Logical Unit Through rescan-scsi-bus.sh”.

**echo "- - -" > /sys/class/scsi_host/host/*scan**

This is the same command described in Section 20.6, “Adding a Storage Device or Path” to add a storage device or path. In this case, however, the channel number, SCSI target ID, and LUN...
values are replaced by wildcards. Any combination of identifiers and wildcards is allowed, allowing you to make the command as specific or broad as needed. This procedure will add LUNs, but not remove them.

\texttt{rmmod} \texttt{driver-name} or \texttt{modprobe} \texttt{driver-name}

These commands completely re-initialize the state of all interconnects controlled by the driver. Although this is extreme, it may be appropriate in some situations. This may be used, for example, to re-start the driver with a different module parameter value.

20.9. iSCSI Discovery Configuration

The default iSCSI configuration file is \texttt{/etc/iscsi/iscsid.conf}. This file contains iSCSI settings used by \texttt{iscsid} and \texttt{iscsiadm}.

During target discovery, the \texttt{iscsiadm} tool uses the settings in \texttt{/etc/iscsi/iscsid.conf} to create two types of records:

Node records in \texttt{/var/lib/iscsi/nodes}

When logging into a target, \texttt{iscsiadm} uses the settings in this file.

Discovery records in \texttt{/var/lib/iscsi/discovery\_type}

When performing discovery to the same destination, \texttt{iscsiadm} uses the settings in this file.

Before using different settings for discovery, delete the current discovery records (i.e. \texttt{/var/lib/iscsi/discovery\_type}) first. To do this, use the following command:

\texttt{iscsiadm} -m discovery -t \texttt{discovery\_type} -p target\_IP:port -o delete

Here, \texttt{discovery\_type} can be either \texttt{sendtargets}, \texttt{isns}, or \texttt{fw}.

There are two ways to reconfigure discovery record settings:

- Edit the \texttt{/etc/iscsi/iscsid.conf} file directly prior to performing a discovery. Discovery settings use the prefix \texttt{discovery}; to view them, run:

\texttt{iscsiadm} -m discovery -t \texttt{discovery\_type} -p target\_IP:port

- Alternatively, \texttt{iscsiadm} can also be used to directly change discovery record settings, as in:

\texttt{iscsiadm} -m discovery -t \texttt{discovery\_type} -p target\_IP:port -o update -n setting -v %value

Refer to \texttt{man iscsiadm} for more information on available \texttt{settings} and valid \texttt{values} for each.

After configuring discovery settings, any subsequent attempts to discover new targets will use the new settings. Refer to Section 20.11, "Scanning iSCSI Interconnects" for details on how to scan for new iSCSI targets.

For more information on configuring iSCSI target discovery, refer to the \texttt{man} pages of \texttt{iscsiadm} and \texttt{iscsid}. The \texttt{/etc/iscsi/iscsid.conf} file also contains examples on proper configuration syntax.

\textsuperscript{3} The \texttt{target\_IP} and \texttt{port} variables refer to the IP address and port combination of a target/portal, respectively. For more information, refer to Section 20.2.1, "iSCSI API" and Section 20.11, "Scanning iSCSI Interconnects".

\textsuperscript{4} For details on different types of discovery, refer to the DISCOVERY TYPES section of \texttt{man iscsiadm}.
20.10. Configuring iSCSI Offload and Interface Binding

This chapter describes how to set up iSCSI interfaces in order to bind a session to a NIC port when using software iSCSI. It also describes how to set up interfaces for use with network devices that support offloading; namely, devices from Chelsio, Broadcom and ServerEngines.

The network subsystem can be configured to determine the path/NIC that iSCSI interfaces should use for binding. For example, if portals and NICs are set up on different subnets, then it is not necessary to manually configure iSCSI interfaces for binding.

Before attempting to configure an iSCSI interface for binding, run the following command first:

```
ping -I ethX target_IP
```

If `ping` fails, then you will not be able to bind a session to a NIC. If this is the case, check the network settings first.

### 20.10.1. Viewing Available iface Configurations

Fedora 11 (and later) supports iSCSI offload and interface binding for the following iSCSI initiator implementations:

- **Software iSCSI** — like the `scsi_tcp` and `ib_iser` modules, this stack allocates an iSCSI host instance (i.e. `scsi_host`) per session, with a single connection per session. As a result, `/sys/class/scsi_host` and `/proc/scsi` will report a `scsi_host` for each connection/session you are logged into.

- **Offload iSCSI** — like the Chelsio `cxgb3i`, Broadcom `bnx2i` and ServerEngines `be2iscsi` modules, this stack allocates a `scsi_host` for each PCI device. As such, each port on a host bus adapter will show up as a different PCI device, with a different `scsi_host` per HBA port.

To manage both types of initiator implementations, `iscsiadm` uses the `iface` structure. With this structure, an `iface` configuration must be entered in `/var/lib/iscsi/ifaces` for each HBA port, software iSCSI, or network device (ethX) used to bind sessions.

To view available `iface` configurations, run `iscsiadm -m iface`. This will display `iface` information in the following format:

```
iface_name transport_name,hardware_address,ip_address,net_ifacename,initiator_name
```

Refer to the following table for an explanation of each value/setting.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>iface_name</code></td>
<td>iFace configuration name.</td>
</tr>
<tr>
<td><code>transport_name</code></td>
<td>Name of driver</td>
</tr>
<tr>
<td><code>hardware_address</code></td>
<td>MAC address</td>
</tr>
<tr>
<td><code>ip_address</code></td>
<td>IP address to use for this port</td>
</tr>
<tr>
<td><code>net_ifacename</code></td>
<td>Name used for the <code>vlan</code> or alias binding of a software iSCSI session. For iSCSI offloads, <code>net_ifacename</code> will be <code>&lt;empty&gt;</code> because this value is not persistent across reboots.</td>
</tr>
</tbody>
</table>
### Configuring an iface for Software iSCSI

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>initiator_name</strong></td>
<td>This setting is used to override a default name for the initiator, which is defined in <code>/etc/iscsi/initiatorname.iscsi</code></td>
</tr>
</tbody>
</table>

The following is a sample output of the `iscsiadm -m iface` command:

```
iface0 qla4xxx,00:c0:dd:08:63:e8,20.15.0.7,default,iqn.2005-06.com.redhat:madmax
iface1 qla4xxx,00:c0:dd:08:63:ea,20.15.0.9,default,iqn.2005-06.com.redhat:madmax
```

For software iSCSI, each **iface** configuration must have a unique name (with less than 65 characters). The **iface_name** for network devices that support offloading appears in the format **transport_name.hardware_name**.

For example, the sample output of `iscsiadm -m iface` on a system using a Chelsio network card might appear as:

```
default tcp,<empty>,<empty>,<empty>,<empty>
iser iser,<empty>,<empty>,<empty>,<empty>
cxgb3i.00:07:43:05:97:07 cxgb3i,00:07:43:05:97:07,<empty>,<empty>,<empty>
```

It is also possible to display the settings of a specific **iface** configuration in a more friendly way. To do so, use the option `-I iface_name`. This will display the settings in the following format:

```
iface.setting = value
```

Using the previous example, the **iface** settings of the same Chelsio video card (i.e. `iscsiadm -m iface -I cxgb3i.00:07:43:05:97:07`) would appear as:

```
# BEGIN RECORD 2.0-871
iface.iscsi_ifacename = cxgb3i.00:07:43:05:97:07
iface.net_ifacename = <empty>
iface.ipaddress = <empty>
iface.hwaddress = 00:07:43:05:97:07
iface.transport_name = cxgb3i
iface.initiatorname = <empty>
# END RECORD
```

#### 20.10.2. Configuring an iface for Software iSCSI

As mentioned earlier, an **iface** configuration is required for each network object that will be used to bind a session.

**Before**

To create a new **iface** configuration for software iSCSI, run the following command:

```
iscsiadm -m iface -I iface_name --op=new
```

This will create a new empty **iface** configuration with a specified **iface_name**. If an existing **iface** configuration already has the same **iface_name**, then it will be overwritten with a new, empty one.

To configure a specific setting of an **iface** configuration, use the following command:

```
iscsiadm -m iface -I iface_name --op=update -n iface.setting -v hw_address
```
For example, to set the MAC address (hardware_address) of iface0 to 00:0F:1F:92:6B:BF, run:

```
iscsiadm -m iface -I iface0 - -op=update -n iface.hwaddress -v 00:0F:1F:92:6B:BF
```

**Warning**
Do not use default or iser as iface names. Both strings are special values used by iscsiadm for backward compatibility. Any manually-created iface configurations named default or iser will disable backwards compatibility.

### 20.10.3. Configuring an iface for iSCSI Offload

By default, iscsiadm will create an iface configuration for each Chelsio, Broadcom, and ServerEngines port. To view available iface configurations, use the same command for doing so in software iSCSI, i.e. iscsiadm -m iface.

Before using the iface of a network card for iSCSI offload, first set the IP address (target_IP) that the device should use. For ServerEngines devices that use the be2iscsi driver (i.e. ServerEngines iSCSI HBAs), the IP address is configured in the ServerEngines BIOS setup screen.

For Chelsio and Broadcom devices, the procedure for configuring the IP address is the same as for any other iface setting. So to configure the IP address of the iface, use:

```
iscsiadm -m iface -I iface_name -o update -n iface.ipaddress -v target_IP
```

For example, to set the iface IP address of a Chelsio card (with iface name cxgb3i.00:07:43:05:97:07) to 20.15.0.66, use:

```
iscsiadm -m iface -I cxgb3i.00:07:43:05:97:07 -o update -n iface.ipaddress -v 20.15.0.66
```

### 20.10.4. Binding/Unbinding an iface to a Portal

Whenever iscsiadm is used to scan for interconnects, it will first check the iface.transport settings of each iface configuration in /var/lib/iscsi/ifaces. The iscsiadm utility will then bind discovered portals to any iface whose iface.transport is tcp.

This behavior was implemented for compatibility reasons. To override this, use the -I iface_name to specify which portal to bind to an iface, as in:

```
iscsiadm -m discovery -t st -p target_IP:port -I iface_name -P 1
```

By default, the iscsiadm utility will not automatically bind any portals to iface configurations that use offloading. This is because such iface configurations will not have iface.transport set to tcp. As such, the iface configurations of Chelsio, Broadcom, and ServerEngines ports need to be manually bound to discovered portals.

It is also possible to prevent a portal from binding to any existing iface. To do so, use default as the iface_name, as in:

```
iscsiadm -m discovery -t st -p IP:port -I default -P 1
```

To remove the binding between a target and iface, use:
Scanning iSCSI Interconnects

iscsiadm -m node -targetname proper_target_name -I iface0 --op=delete

To delete all bindings for a specific iface, use:

iscsiadm -m node -I iface_name --op=delete

To delete bindings for a specific portal (e.g. for Equalogic targets), use:

iscsiadm -m node -p IP:port -I iface_name --op=delete

Note

If there are no iface configurations defined in /var/lib/iscsi iface and the -I option is not used, iscsiadm will allow the network subsystem to decide which device a specific portal should use.

20.11. Scanning iSCSI Interconnects

For iSCSI, if the targets send an iSCSI async event indicating new storage is added, then the scan is done automatically. Cisco MDS™ and EMC Celerra™ support this feature.

However, if the targets do not send an iSCSI async event, you need to manually scan them using the iscsiadm utility. Before doing so, however, you need to first retrieve the proper --targetname and the --portal values. If your device model supports only a single logical unit and portal per target, use iscsiadm to issue a sendtargets command to the host, as in:

iscsiadm -m discovery -t sendtargets -p target_IP:port

The output will appear in the following format:

target_IP:port,target_portal_group_tag proper_target_name

For example, on a target with a proper_target_name of iqn.1992-08.com.netapp:sn.33615311 and a target_IP:port of 10.15.85.19:3260, the output may appear as:

10.15.84.19:3260,2 iqn.1992-08.com.netapp:sn.33615311

In this example, the target has two portals, each using target_ip:ports of 10.15.84.19:3260 and 10.15.85.19:3260.

To see which iface configuration will be used for each session, add the -P 1 option. This option will print also session information in tree format, as in:

Target: proper_target_name
Portal: target_IP:port,target_portal_group_tag
  Iface Name: iface_name

5 Refer to Section 20.11, "Scanning iSCSI Interconnects" for information on proper_target_name.
Chapter 20. Online Storage Management

For example, with `iscsiadm -m discovery -t sendtargets -p 10.15.85.19:3260 -P 1`, the output may appear as:

```
  Portal: 10.15.84.19:3260,2
    Iface Name: iface2
  Portal: 10.15.85.19:3260,3
    Iface Name: iface2
```

This means that the target `iqn.1992-08.com.netapp:sn.33615311` will use `iface2` as its `iFace` configuration.

With some device models (e.g. from EMC and Netapp), however, a single target may have multiple logical units and/or portals. In this case, issue a `sendtargets` command to the host first to find new portals on the target. Then, rescan the existing sessions using:

```
iscsiadm -m session --rescan
```

You can also rescan a specific session by specifying the session’s `SID` value, as in:

```
iscsiadm -m session -r SID --rescan
```

If your device supports multiple targets, you will need to issue a `sendtargets` command to the hosts to find new portals for each target. Then, rescan existing sessions to discover new logical units on existing sessions (i.e. using the `--rescan` option).

**Important**

The `sendtargets` command used to retrieve `--targetname` and `--portal` values overwrites the contents of the `/var/lib/iscsi/nodes` database. This database will then be repopulated using the settings in `/etc/iscsi/iscsid.conf`. However, this will not occur if a session is currently logged in and in use.

To safely add new targets/portals or delete old ones, use the `-o new` or `-o delete` options, respectively. For example, to add new targets/portals without overwriting `/var/lib/iscsi/nodes`, use the following command:

```
iscsiadm -m discovery -t st -p target_IP -o new
```

To delete `/var/lib/iscsi/nodes` entries that the target did not display during discovery, use:

```
iscsiadm -m discovery -t st -p target_IP -o delete
```

You can also perform both tasks simultaneously, as in:

```
iscsiadm -m discovery -t st -p target_IP -o delete -o new
```

---

6 For information on how to retrieve a session’s SID value, refer to Section 20.2.1, "iSCSI API".
The `sendtargets` command will yield the following output:

```
ip:port,target_portal_group_tag proper_target_name
```

For example, given a device with a single target, logical unit, and portal, with `equallogic-iscsi1` as your `target_name`, the output should appear similar to the following:

```
10.16.41.155:3260,0 iqn.2001-05.com.equallogic:6-8a0900-ac3fe0101-63aff113e344a4a2-dl585-03-1
```

Note that `proper_target_name` and `ip:port,target_portal_group_tag` are identical to the values of the same name in Section 20.2.1, “iSCSI API”.

At this point, you now have the proper `--targetname` and `--portal` values needed to manually scan for iSCSI devices. To do so, run the following command:

```
iscsiadm --mode node --targetname proper_target_name --portal ip:port,target_portal_group_tag --login
```

Using our previous example (where `proper_target_name` is `equallogic-iscsi1`), the full command would be:

```
iscsiadm --mode node --targetname \ iqన.2001-05.com.equallogic:6-8a0900-ac3fe0101-63aff113e344a4a2-dl585-03-1 \ --portal 10.16.41.155:3260,0 --login
```

### 20.12. Logging In to an iSCSI Target

As mentioned in Section 20.2, “iSCSI”, the iSCSI service must be running in order to discover or log into targets. To start the iSCSI service, run:

```
service iscsi start
```

When this command is executed, the iSCSI init scripts will automatically log into targets where the `node.startup` setting is configured as `automatic`. This is the default value of `node.startup` for all targets.

To prevent automatic login to a target, set `node.startup` to `manual`. To do this, run the following command:

```
iscsiadm -m node --targetname proper_target_name -p target_IP:port -o update -n node.startup -v manual
```

Deleting the entire record will also prevent automatic login. To do this, run:

```
iscsiadm -m node --targetname proper_target_name -p target_IP:port -o delete
```

To automatically mount a file system from an iSCSI device on the network, add a partition entry for the mount in `/etc/fstab` with the `_netdev` option. For example, to automatically mount the iSCSI device `sdb` to `/mount/iscsi` during startup, add the following line to `/etc/fstab`:

```
```

---

7 This is a single command split into multiple lines, to accommodate printed and PDF versions of this document. All concatenated lines — preceded by the backslash (\) — should be treated as one command, sans backslashes.
To manually log in to an iSCSI target, use the following command:

```
iscsiadm -m node --targetname {proper_target_name} -p {target_IP:port} -l
```

Note

The `proper_target_name` and `target_IP:port` refer to the full name and IP address/port combination of a target. For more information, refer to Section 20.2.1, "iSCSI API" and Section 20.11, "Scanning iSCSI Interconnects".

### 20.13. Resizing an Online Logical Unit

In most cases, fully resizing an online logical unit involves two things: resizing the logical unit itself and reflecting the size change in the corresponding multipath device (if multipathing is enabled on the system).

To resize the online logical unit, start by modifying the logical unit size through the array management interface of your storage device. This procedure differs with each array; as such, consult your storage array vendor documentation for more information on this.

Note

In order to resize an online file system, the file system must not reside on a partitioned device.

#### 20.13.1. Resizing Fibre Channel Logical Units

After modifying the online logical unit size, re-scan the logical unit to ensure that the system detects the updated size. To do this for Fibre Channel logical units, use the following command:

```
echo 1 > /sys/block/sdX/device/rescan
```

Important

To re-scan fibre channel logical units on a system that uses multipathing, execute the aforementioned command for each sd device (i.e. `sd1`, `sd2`, and so on) that represents a path for the multipathed logical unit. To determine which devices are paths for a multipath logical unit, use `multipath -ll`; then, find the entry that matches the logical unit being resized. It is advisable that you refer to the WWID of each entry to make it easier to find which one matches the logical unit being resized.

#### 20.13.2. Resizing an iSCSI Logical Unit

After modifying the online logical unit size, re-scan the logical unit to ensure that the system detects the updated size. To do this for iSCSI devices, use the following command:

```
iscsiadm -m node --targetname {target_name} -R
```

Replace `target_name` with the name of the target where the device is located.
Note

You can also re-scan iSCSI logical units using the following command:

```bash
iscsiadm -m node -R -I interface
```

Replace `interface` with the corresponding interface name of the resized logical unit (for example, `iface0`). This command performs two operations:

- It scans for new devices in the same way that the command `echo "- - -" > /sys/class/scsi_host/host/scan` does (refer to Section 20.11, "Scanning iSCSI Interconnects").
- It re-scans for new/modified logical units the same way that the command `echo 1 > /sys/block/sdX/device/rescan` does. Note that this command is the same one used for re-scanning fibre-channel logical units.

20.13.3. Updating the Size of Your Multipath Device

If multipathing is enabled on your system, you will also need to reflect the change in logical unit size to the logical unit's corresponding multipath device (after resizing the logical unit). For Fedora 12 (and later), you can do this through `multipathd`. To do so, first ensure that `multipathd` is running using `service multipathd status`. Once you've verified that `multipathd` is operational, run the following command:

```bash
multipathd -k"resize map multipath_device"
```

The `multipath_device` variable is the corresponding multipath entry of your device in `/dev/mapper`. Depending on how multipathing is set up on your system, `multipath_device` can be either of two formats:

- `mpathX`, where `X` is the corresponding entry of your device (for example, `mpath0`)
- a WWID; for example, `3600508b400105e210000900000490000`

To determine which multipath entry corresponds to your resized logical unit, run `multipath -ll`. This displays a list of all existing multipath entries in the system, along with the major and minor numbers of their corresponding devices.

Important

Do not use `multipathd -k"resize map multipath_device"` if there are any commands queued to `multipath_device`. That is, do not use this command when the `no_path_retry` parameter (in `/etc/multipath.conf`) is set to "queue", and there are no active paths to the device.

If your system is using an earlier version of Fedora, you will need to perform the following procedure to instruct the `multipathd` daemon to recognize (and adjust to) the changes you made to the resized logical unit:

Procedure 20.4. Resizing the Corresponding Multipath Device (Required for Fedora 12 and earlier)

1. Dump the device mapper table for the multipathed device using:
Chapter 20. Online Storage Management

dmsetup table multipath_device

2. Save the dumped device mapper table as table_name. This table will be re-loaded and edited later.

3. Examine the device mapper table. Note that the first two numbers in each line correspond to the start and end sectors of the disk, respectively.

4. Suspend the device mapper target:
   
   dmsetup suspend multipath_device

5. Open the device mapper table you saved earlier (i.e. table_name). Change the second number (i.e. the disk end sector) to reflect the new number of 512 byte sectors in the disk. For example, if the new disk size is 2GB, change the second number to 4194304.

6. Reload the modified device mapper table:
   
   dmsetup reload multipath_device table_name

7. Resume the device mapper target:
   
   dmsetup resume multipath_device

For more information about multipathing, refer to the Using Device-Mapper Multipath guide (in http://www.redhat.com/docs/manuals/enterprise/).


The sg3_utils package provides the rescan-scsi-bus.sh script, which can automatically update the logical unit configuration of the host as needed (after a device has been added to the system). The rescan-scsi-bus.sh script can also perform an issue_lip on supported devices. For more information about how to use this script, refer to rescan-scsi-bus.sh --help.

To install the sg3_utils package, run yum install sg3_utils.

Known Issues With rescan-scsi-bus.sh

When using the rescan-scsi-bus.sh script, take note of the following known issues:

- In order for rescan-scsi-bus.sh to work properly, LUN0 must be the first mapped logical unit. The rescan-scsi-bus.sh can only detect the first mapped logical unit if it is LUN0. The rescan-scsi-bus.sh will not be able to scan any other logical unit unless it detects the first mapped logical unit even if you use the --nootscan option.

- A race condition requires that rescan-scsi-bus.sh be run twice if logical units are mapped for the first time. During the first scan, rescan-scsi-bus.sh only adds LUN0; all other logical units are added in the second scan.

- A bug in the rescan-scsi-bus.sh script incorrectly executes the functionality for recognizing a change in logical unit size when the --remove option is used.

---

• The `rescan-scsi-bus.sh` script does not recognize iSCSI logical unit removals.

### 20.15. Modifying Link Loss Behavior

This section describes how to modify the link loss behavior of devices that use either fibre channel or iSCSI protocols.

#### 20.15.1. Fibre Channel

If a driver implements the Transport `dev_loss_tmo` callback, access attempts to a device through a link will be blocked when a transport problem is detected. To verify if a device is blocked, run the following command:

```bash
cat /sys/block/device/device/state
```

This command will return `blocked` if the device is blocked. If the device is operating normally, this command will return `running`.

**Procedure 20.5. Determining The State of a Remote Port**

1. To determine the state of a remote port, run the following command:

   ```bash
cat /sys/class/fc_remote_port/rport-H:B:R/port_state
```

2. This command will return `Blocked` when the remote port (along with devices accessed through it) are blocked. If the remote port is operating normally, the command will return `Online`.

3. If the problem is not resolved within `dev_loss_tmo` seconds, the rport and devices will be unblocked and all I/O running on that device (along with any new I/O sent to that device) will be failed.

**Procedure 20.6. Changing `dev_loss_tmo`**

- To change the `dev_loss_tmo` value, `echo` in the desired value to the file. For example, to set `dev_loss_tmo` to 30 seconds, run:

  ```bash
echo 30 > /sys/class/fc_remote_port/rport-H:B:R/dev_loss_tmo
```

For more information about `dev_loss_tmo`, refer to Section 20.1.1, “Fibre Channel API”.

When a device is blocked, the fibre channel class will leave the device as is; i.e. `/dev/sdx` will remain `/dev/sdx`. This is because the `dev_loss_tmo` expired. If the link problem is fixed at a later time, operations will continue using the same SCSI device and device node name.

**Fibre Channel: remove_on_dev_loss**

If you prefer that devices are removed at the SCSI layer when links are marked bad (i.e. expired after `dev_loss_tmo` seconds), you can use the `scsi_transport_fc` module parameter `remove_on_dev_loss`. When a device is removed at the SCSI layer while `remove_on_dev_loss` is in effect, the device will be added back once all transport problems are corrected.
Warning

The use of `remove_on_dev_loss` is not recommended, as removing a device at the SCSI layer does not automatically unmount any file systems from that device. When file systems from a removed device are left mounted, the device may not be properly removed from multipath or RAID devices.

Further problems may arise from this if the upper layers are not hotplug-aware. This is because the upper layers may still be holding references to the state of the device before it was originally removed. This can cause unexpected behavior when the device is added again.

20.15.2. iSCSI Settings With `dm-multipath`

If `dm-multipath` is implemented, it is advisable to set iSCSI timers to immediately defer commands to the multipath layer. To configure this, nest the following line under `device { in /etc/multipath.conf`:

```
features  "1 queue_if_no_path"
```

This ensures that I/O errors are retried and queued if all paths are failed in the `dm-multipath` layer.

You may need to adjust iSCSI timers further to better monitor your SAN for problems. Available iSCSI timers you can configure are `NOP-Out Interval/Timeouts` and `replacement_timeout`, which are discussed in the following sections.

20.15.2.1. NOP-Out Interval/Timeout

To help monitor problems the SAN, the iSCSI layer sends a NOP-Out request to each target. If a NOP-Out request times out, the iSCSI layer responds by failing any running commands and instructing the SCSI layer to requeue those commands when possible.

When `dm-multipath` is being used, the SCSI layer will fail those running commands and defer them to the multipath layer. The multipath layer then retries those commands on another path. If `dm-multipath` is not being used, those commands are retried five times before failing altogether.

Intervals between NOP-Out requests are 10 seconds by default. To adjust this, open `/etc/iscsi/iscsid.conf` and edit the following line:

```
node.conn[0].timeo.noop_out_interval = [interval value]
```

Once set, the iSCSI layer will send a NOP-Out request to each target every `[interval value]` seconds.

By default, NOP-Out requests time out in 10 seconds. To adjust this, open `/etc/iscsi/iscsid.conf` and edit the following line:

```
node.conn[0].timeo.noop_out_timeout = [timeout value]
```

9 In previous versions of Fedora, the default NOP-Out requests time out was 15 seconds.
This sets the iSCSI layer to timeout a NOP-Out request after \([timeout \ value]\) seconds.

**SCSI Error Handler**

If the SCSI Error Handler is running, running commands on a path will not be failed immediately when a NOP-Out request times out on that path. Instead, those commands will be failed after \(replacement\_timeout\) seconds. For more information about \(replacement\_timeout\), refer to Section 20.15.2.2, “\(replacement\_timeout\)”.

To verify if the SCSI Error Handler is running, run:

```bash
iscsiadm -m session -P 3
```

**20.15.2.2. replacement\_timeout**

\(replacement\_timeout\) controls how long the iSCSI layer should wait for a timed-out path/session to reestablish itself before failing any commands on it. The default \(replacement\_timeout\) value is 120 seconds.

To adjust \(replacement\_timeout\), open `/etc/iscsi/iscsid.conf` and edit the following line:

```bash
node.session.timeo.replacement_timeout = [replacement\_timeout]
```

The `queue_if_no_path` option in `/etc/multipath.conf` sets iSCSI timers to immediately defer commands to the multipath layer (refer to Section 20.15.2, “iSCSI Settings With dm-multipath”). This setting prevents I/O errors from propagating to the application; because of this, you can set \(replacement\_timeout\) to 15-20 seconds.

By configuring a lower \(replacement\_timeout\), I/O is quickly sent to a new path and executed (in the event of a NOP-Out timeout) while the iSCSI layer attempts to re-establish the failed path/session. If all paths time out, then the multipath and device mapper layer will internally queue I/O based on the settings in `/etc/multipath.conf` instead of `/etc/iscsi/iscsid.conf`.

**Important**

Whether your considerations are failover speed or security, the recommended value for \(replacement\_timeout\) will depend on other factors. These factors include the network, target, and system workload. As such, it is recommended that you thoroughly test any new configurations to \(replacements\_timeout\) before applying it to a mission-critical system.

**20.15.3. iSCSI Root**

When accessing the root partition directly through an iSCSI disk, the iSCSI timers should be set so that iSCSI layer has several chances to try to reestablish a path/session. In addition, commands should not be quickly re-queued to the SCSI layer. This is the opposite of what should be done when `dm-multipath` is implemented.

To start with, NOP-Outs should be disabled. You can do this by setting both NOP-Out interval and timeout to zero. To set this, open `/etc/iscsi/iscsid.conf` and edit as follows:

```bash
node.conn[0].timeo.noop_out_interval = 0
```
node.conn[0].timeo.noop_out_timeout = 0

In line with this, replacement_timeout should be set to a high number. This will instruct the system to wait a long time for a path/session to reestablish itself. To adjust replacement_timeout, open /etc/iscsi/iscsid.conf and edit the following line:

node.session.timeo.replacement_timeout = replacement_timeout

After configuring /etc/iscsi/iscsid.conf, you must perform a re-discovery of the affected storage. This will allow the system to load and use any new values in /etc/iscsi/iscsid.conf. For more information on how to discover iSCSI devices, refer to Section 20.11, "Scanning iSCSI Interconnects".

**Configuring Timeouts for a Specific Session**

You can also configure timeouts for a specific session and make them non-persistent (instead of using /etc/iscsi/iscsid.conf). To do so, run the following command (replace the variables accordingly):

```
iscsiadm -m node -T target_name -p target_IP:port -o update -n node.session.timeo.replacement_timeout -v $timeout_value
```

**Important**

The configuration described here is recommended for iSCSI sessions involving root partition access. For iSCSI sessions involving access to other types of storage (namely, in systems that use dm-multipath), refer to Section 20.15.2, "iSCSI Settings With dm-multipath".

### 20.16. Controlling the SCSI Command Timer and Device Status

The Linux SCSI layer sets a timer on each command. When this timer expires, the SCSI layer will quiesce the host bus adapter (HBA) and wait for all outstanding commands to either time out or complete. Afterwards, the SCSI layer will activate the driver's error handler.

When the error handler is triggered, it attempts the following operations in order (until one successfully executes):

1. Abort the command.
2. Reset the device.
3. Reset the bus.
4. Reset the host.

If all of these operations fail, the device will be set to the offline state. When this occurs, all I/O to that device will be failed, until the problem is corrected and the user sets the device to running.

The process is different, however, if a device uses the fibre channel protocol and the rport is blocked. In such cases, the drivers wait for several seconds for the rport to become online again before activating the error handler. This prevents devices from becoming offline due to temporary transport problems.
Device States
To display the state of a device, use:

```
cat /sys/block/device-name/device/state
```
To set a device to running state, use:

```
echo running > /sys/block/device-name/device/state
```

Command Timer
To control the command timer, you can write to `/sys/block/device-name/device/timeout`. To do so, run:

```
echo value /sys/block/device-name/device/timeout
```
Here, `value` is the timeout value (in seconds) you want to implement.

20.17. Troubleshooting
This section provides solution to common problems users experience during online storage reconfiguration.

Logical unit removal status is not reflected on the host.
When a logical unit is deleted on a configured filer, the change is not reflected on the host. In such cases, `lvm` commands will hang indefinitely when `dm-multipath` is used, as the logical unit has now become stale.

To work around this, perform the following procedure:

**Procedure 20.7. Working Around Stale Logical Units**

1. Determine which `mpath` link entries in `/etc/lvm/cache/.cache` are specific to the stale logical unit. To do this, run the following command:

   ```
   ls -l /dev/mpath | grep stale-logical-unit
   ```

   For example, if `stale-logical-unit` is `3600d0230003414f30000203a7bc41a00`, the following results may appear:

   ```
   lrwxrwxrwx 1 root root 7 Aug  2 10:33 /3600d0230003414f30000203a7bc41a00 -> ../dm-4
   lrwxrwxrwx 1 root root 7 Aug  2 10:33 /3600d0230003414f30000203a7bc41a00p1 -> ../dm-5
   ```

   This means that `3600d0230003414f30000203a7bc41a00` is mapped to two `mpath` links: `dm-4` and `dm-5`.

2. Next, open `/etc/lvm/cache/.cache`. Delete all lines containing `stale-logical-unit` and the `mpath` links that `stale-logical-unit` maps to.

   Using the same example in the previous step, the lines you need to delete are:

   ```
   /dev/dm-4
   /dev/dm-5
   /dev/mapper/3600d0230003414f30000203a7bc41a00
   /dev/mapper/3600d0230003414f30000203a7bc41a00p1
   ```
Chapter 20. Online Storage Management

/dev/mpath/3600d0230003414f30000203a7bc41a00
/dev/mpath/3600d0230003414f30000203a7bc41a00p1
Device Mapper Multipathing and Virtual Storage

Fedora 14 also supports DM-Multipath and virtual storage. Both features are documented in detail in other stand-alone books also provided by Red Hat.

21.1. Virtual Storage

Fedora 14 supports the following file systems/online storage methods for virtual storage:

- Fibre Channel
- iSCSI
- NFS
- GFS2

Virtualization in Fedora 14 uses libvirt to manage virtual instances. The libvirt utility uses the concept of storage pools to manage storage for virtualized guests. A storage pool is storage that can be divided up into smaller volumes or allocated directly to a guest. Volumes of a storage pool can be allocated to virtualized guests. There are two categories of storage pools available:

Local storage pools

Local storage covers storage devices, files or directories directly attached to a host. Local storage includes local directories, directly attached disks, and LVM Volume Groups.

Networked (shared) storage pools

Networked storage covers storage devices shared over a network using standard protocols. Networked storage includes shared storage devices using Fibre Channel, iSCSI, NFS, GFS2, and SCSI RDMA protocols. Networked storage is a requirement for migrating guest virtualized guests between hosts.

Important

For comprehensive information on the deployment and configuration of virtual storage instances in your environment, please refer to the Virtualization Storage section of the Virtualization guide provided by Red Hat.

21.2. DM-Multipath

Device Mapper Multipathing (DM-Multipath) is a feature that allows you to configure multiple I/O paths between server nodes and storage arrays into a single device. These I/O paths are physical SAN connections that can include separate cables, switches, and controllers. Multipathing aggregates the I/O paths, creating a new device that consists of the aggregated paths.

DM-Multipath are used primarily for the following reasons:

Redundancy

DM-Multipath can provide failover in an active/passive configuration. In an active/passive configuration, only half the paths are used at any time for I/O. If any element of an I/O path (the cable, switch, or controller) fails, DM-Multipath switches to an alternate path.
Improved Performance

DM-Multipath can be configured in active/active mode, where I/O is spread over the paths in a round-robin fashion. In some configurations, DM-Multipath can detect loading on the I/O paths and dynamically re-balance the load.

Important

For comprehensive information on the deployment and configuration of DM-Multipath in your environment, please refer to the *Using DM-Multipath* guide provided by Red Hat.
Appendix A. Revision History

Revision 1.0  Thu Jul 09 2009  Don Domingo ddomingo@redhat.com
initial build
Glossary

This glossary defines common terms relating to file systems and storage used throughout the Storage Administration Guide.

Defragmentation  The act of reorganizing a file's data blocks so that they are more physically contiguous on disk.

Delayed Allocation  An allocator behavior in which disk locations are chosen when data is flushed to disk, rather than when the write occurs. This can generally lead to more efficient allocation because the allocator is called less often and with larger requests.

Extended Attributes  Name/Value metadata pairs which may be associated with a file.

Extent  A unit of file allocation, stored in the file's metadata as an offset, length pair. A single extent record can describe many contiguous blocks in a file.

File System Repair (fsck)  A method of verifying and repairing consistency of a file system's metadata. May be needed post-crash for non-journalling file systems, or after a hardware failure or kernel bug.

Fragmentation  The condition in which a file's data blocks are not allocated in contiguous physical (disk) locations for contiguous logical offsets within the file. File fragmentation can lead to poor performance in some situations, due to disk seek time.

Metadata Journaling  A method used to ensure that a file system's metadata is consistent even after a system crash. Metadata journaling can take different forms, but in each case a journal or log can be replayed after a crash, writing only consistent transactional changes to the disk.

Persistent Preallocation  A type of file allocation which chooses locations on disk, and marks these blocks as used regardless of when or if they are written. Until data is written into these blocks, reads will return 0s. Preallocation is performed with the fallocate() glibc function.

POSIX Access Control Lists (ACLs)  Metadata attached to a file which permits more fine-grained access controls. ACLs are often implemented as a special type of extended attribute.

Quota  A limit on block or inode usage of individual users and groups in a file system, set by the administrator.

Stripe Unit  Also sometimes referred to as stride or chunk-size. The stripe unit is the amount of data written to one component of striped storage before moving on to the next. Specified in byte or file system block units.

Stripe Width  The number of individual data stripe units in striped storage (excluding parity). Depending on the administrative tool used, may be specified in byte or file system block units, or in multiples of the stripe unit.

Stripe-aware allocation  An allocator behavior in which allocations and I/O are well-aligned to underlying striped storage. This depends on stripe information being
available at mkfs time as well. Doing well-aligned allocation I/O can avoid inefficient read-modify-write cycles on the underlying storage.

Write Barriers

A method to enforce consistent I/O ordering on storage devices which have volatile write caches. Barriers must be used to ensure that after a power loss, the ordering guarantees required by metadata journaling are not violated due to the storage hardware writing out blocks from its volatile write cache in a different order than the operating system requested.

See Also Metadata Journaling.
Index

Symbols
'software iSCSI
   offload and interface binding
   iSCSI, 133
/boot/ directory, 31
/dev/disk
   persistent naming, 124
/dev/shm, 30
/etc/fstab, 40, 60
/etc/fstab file
   enabling disk quotas with, 95
/local/directory (client configuration, mounting)
   NFS, 59
/proc
   /proc/devices, 37
   /proc/filesystems, 37
   /proc/mdstat, 37
   /proc/mounts, 37
   /proc/mounts/, 37
   /proc/partitions, 37
   /proc/devices
      virtual file system (/proc), 37
   /proc/filesystems
      virtual file system (/proc), 37
   /proc/mdstat
      virtual file system (/proc), 37
   /proc/mounts
      virtual file system (/proc), 37
   /proc/mounts/
      virtual file system (/proc), 37
   /proc/partitions
      virtual file system (/proc), 37
   /remote/export (client configuration, mounting)
      NFS, 59

A
Access Control Lists (see ACLs)
ACLs
   access ACLs, 101
   additional resources, 104
   archiving with, 103
   default ACLs, 102
   getfacl, 102
   mounting file systems with, 101
   mounting NFS shares with, 101
   on ext3 file systems, 101
   retrieving, 102
   setfacl, 102
   setting
      access ACLs, 101
      with Samba, 101
adding a new volume group
   LVM, 16
adding paths to a storage device, 128
adding/removing
   LUN (logical unit number), 140
advanced path
   storage considerations during installation, 3
advanced RAID device creation
   RAID, 90
allocation features
   ext4, 43
   XFS, 49
Anaconda support
   RAID, 89
API, fibre channel, 121
API, iSCSI, 123
ATA standards
   I/O alignment and size, 111
   auto-partitioning and /home
      storage considerations during installation, 3
autofs, 60, 62
   (see also NFS)
autofs version 5
   NFS, 61

B
backup/restoration
   XFS, 54
basic path
   storage considerations during installation, 3
battery-backed write caches
   write barriers, 106
bcull (cache cull limits settings)
   FS-Cache, 81
binding/unbinding an iface to a portal
   offload and interface binding
   iSCSI, 134
block device ioctls (userspace access)
   I/O alignment and size, 110
blocked device, verifying
   fibre channel
      modifying link loss behavior, 141
brun (cache cull limits settings)
   FS-Cache, 81
bstop (cache cull limits settings)
   FS-Cache, 81

C
cache back-end
   FS-Cache, 77
cache cull limits
   FS-Cache, 80
cache limitations with NFS
Index

FS-Cache, 80
  cache setup
    FS-Cache, 78
  cache sharing
    FS-Cache, 79
  cachefiles
    FS-Cache, 78
  cachefilesd
    FS-Cache, 78
  caching, file system
    overview, 1
  CCW, channel command word
    storage considerations during installation, 4
  changing dev_loss_tmo
    fibre channel
      modifying link loss behavior, 141
    channel command word (CCW)
      storage considerations during installation, 4
  coherency data
    FS-Cache, 78
  command timer (SCSI)
    Linux SCSI layer, 145
  configuration
    discovery
      iSCSI, 131
  configuring a tftp service for diskless clients
    diskless systems, 115
  configuring an ethernet interface to use FCoE
    FCoE, 129
  configuring DHCP for diskless clients
    diskless systems, 115
  configuring RAID sets
    RAID, 89
  controlling SCSI command timer and device status
    Linux SCSI layer, 144
  converting ext3 to ext4
    ext4, 44
  creating
    ext4, 44
    XFS, 49
  cumulative mode (xfsrestore)
    XFS, 55

D

DASD and zFCP devices on IBM System z
  storage considerations during installation, 4
debugfs (other ext4 file system utilities)
  ext4, 46

deployment
  solid state disks, 119

deployment guidelines
  solid state disks, 119
determining remote port states

fibre channel
  modifying link loss behavior, 141
dev directory, 31
device status
  Linux SCSI layer, 145
device-mapper multipathing, 147
devices, removing, 126
dev_loss_tmo
  fibre channel
    modifying link loss behavior, 141
    fibre channel API, 122
dev_loss_tmo, changing
  fibre channel
    modifying link loss behavior, 141
df , 29

DHCP, configuring
  diskless systems, 115
DIF/DIX-enabled block devices
  storage considerations during installation, 5
direct map support (autofs version 5)
  NFS, 61
directories
  /boot/ , 31
  /dev/ , 31
  /etc/ , 32
  /lib/ , 32
  /media/ , 32
  /mnt/ , 32
  /opt/ , 32
  /proc/ , 33
  /sbin/ , 33
  /srv/ , 33
  /sys/ , 33
  /usr/ , 34
  /var/ , 35
dirty logs (repairing XFS file systems)
  XFS, 53
disabling NOP-Outs
  iSCSI configuration, 143
disabling write caches
  write barriers, 106
discovery
  iSCSI, 131
disk quotas, 95
  additional resources, 100
  assigning per file system, 98
  assigning per group, 97
  assigning per user, 97
  disabling, 98
  enabling, 95, 98
  /etc/fstab, modifying, 95
creating quota files, 96
  quotacheck, running, 96
grace period, 97
hard limit, 97
management of, 98
quotacheck command, using to check, 99
reporting, 99
soft limit, 97
disk storage (see disk quotas)
parted (see parted)
diskless systems
DHCP, configuring, 115
exported file systems, 116
network booting service, 115
remote diskless systems, 115
required packages, 115
tftp service, configuring, 115
dm-multipath
iSCSI configuration, 142
dmraid
RAID, 89
dmraid (configuring RAID sets)
RAID, 89
documentation
LVM, 21
drivers (native), fibre channel, 122
du, 30
dump levels
XFS, 54

E
e2fsck, 40
e2fsck (converting ext3 to ext4)
ext4, 45
e2image (other ext4 file system utilities)
ext4, 46
e2label
ext4, 46
e2label (other ext4 file system utilities)
ext4, 46
cryptfs
overview, 1
cryptfs
file system types, 83
mount settings for encrypted file systems, 83
mounting, 83
mounting a file system as encrypted, 83
editing a logical volume
LVM, 18
enabling/disabling
write barriers, 105
encrypted file system
mount settings for encrypted file systems, 83
mounting, 83
mounting a file system as encrypted, 83
encryption, file system
overview, 1
Enhanced LDAP support (autofs version 5)
NFS, 61
entries, device mapper table, 140
error messages
write barriers, 106
etc directory, 32
expert mode (xfs_quot)
XFS, 51
exported file systems
diskless systems, 116
ext2
reverting from ext3, 40
ext3
converting from ext2, 40
creating, 40
features, 39
ext3 (mounting as ext4)
ext4, 46
ext4
allocation features, 43
converting ext3 to ext4, 44
creating, 44
debugfs (other ext4 file system utilities), 46
e2fsck (converting ext3 to ext4), 45
e2image (other ext4 file system utilities), 46
e2label, 46
e2label (other ext4 file system utilities), 46
ext3 (mounting as ext4), 46
ext3 to ext4, converting, 44
file system types, 43
fsync(), 43
main features, 43
mkfs.ext4, 44
mounting, 45
mounting ext3 as ext4, 46
nobarrier mount option, 46
other file system utilities, 46
quota (other ext4 file system utilities), 46
resize2fs (resizing ext4), 46
resizing, 46
stride (specifying stripe geometry), 44
stripe geometry, 44
stripe-width (specifying stripe geometry), 44
tune2fs (converting ext3 to ext4), 44
tune2fs (mounting), 45
write barriers, 46
extending a volume group
LVM, 17
extents, migration of
LVM, 13

F
fast_io_fail_tmo
fibre channel API, 122
FCoE
configuring an ethernet interface to use FCoE, 129
fibre channel over ethernet, 129
required packages, 129
storage considerations during installation, 3
fcul (cache cull limits settings)
FS-Cache, 81
Fedora-specific file locations
/etc/sysconfig/, 36
(see also sysconfig directory)
/var/cache/yum, 36
/var/lib/rpm/, 36
feedback
contact information for this manual, xi
FHS, 29, 29
(see also file system)
fibre channel
online storage, 121
fibre channel API, 121
fibre channel drivers (native), 122
fibre channel over ethernet
FCoE, 129
fibre-channel over ethernet
storage considerations during installation, 3
file system
FHS standard, 29
hierarchy, 29
organization, 29
structure, 29
file system caching
overview, 1
file system encryption
overview, 1
file system types
encrypted file system, 83
ext4, 43
XFS, 49
file systems, 29
ext2 (see ext2), 43
ext3 (see ext3), 43
file systems, overview of supported types
storage considerations during installation, 3
frun (cache cull limits settings)
FS-Cache, 81
fs-cache
overview, 1
FS-Cache
bcull (cache cull limits settings), 81
brun (cache cull limits settings), 81
bstop (cache cull limits settings), 81
cache back-end, 77
cache cull limits, 80
cache sharing, 79
cachefiles, 78
cachefilesd, 78
coherency data, 78
fcull (cache cull limits settings), 81
frun (cache cull limits settings), 81
fstop (cache cull limits settings), 81
indexing keys, 78
NFS (cache limitations with), 80
NFS (using with), 79
performance guarantee, 78
setting up a cache, 78
statistical information (tracking), 81
tune2fs (setting up a cache), 78
fstop (cache cull limits settings)
FS-Cache, 81
fsync()
ext4, 43
XFS, 49
G
getfacl, 102
gquota/gqnoenforce
XFS, 51
H
Hardware RAID (see RAID)
hardware RAID controller drivers
RAID, 88
hierarchy, file system, 29
high-end arrays
write barriers, 107
host
fibre channel API, 122
how write barriers work
write barriers, 105
I
I/O alignment and size, 109
ATA standards, 111
block device ioctls (userspace access), 110
Linux I/O stack, 109
logical_block_size, 110
LVM, 112
READ CAPACITY(16), 111
SCSI standards, 111
stacking I/O parameters, 111
storage access parameters, 109
sysfs interface (userspace access), 110
tools (for partitioning and other file system functions), 112
userspace access, 110
I/O limit processing
overview, 1
I/O parameters stacking
I/O alignment and size, 111
I/O scheduler (tuning)
solid state disks, 120
iface (configuring for iSCSI offload)
offload and interface binding
iSCSI, 134
iface binding/unbinding
offload and interface binding
iSCSI, 134
iface configurations, viewing
offload and interface binding
iSCSI, 132
iface for software iSCSI
offload and interface binding
iSCSI, 133
iface settings
offload and interface binding
iSCSI, 132
importance of write barriers
write barriers, 105
increasing file system size
XFS, 52
indexing keys
FS-Cache, 78
initiator implementations
offload and interface binding
iSCSI, 132
inode64 mount option
XFS, 50
installation storage configurations
advanced path, 3
auto-partitioning and /home, 3
basic path, 3
channel command word (CCW), 4
DASD and zFCP devices on IBM System z, 4
DIF/DIX-enabled block devices, 5
fibre-channel over ethernet, 3
file systems, overview of supported types, 3
iSCSI detection and configuration, 5
LUKS/dm-crypt, encrypting block devices using, 4
separate partitions (for /home, /opt, /usr/local), 4
stale BIOS RAID metadata, 4
storage device filter interface, 3
updates, 3
what's new, 3
installer support
RAID, 89
interactive operation (xfsrestore)
XFS, 55
interconnects (scanning)
iSCSI, 135
introduction, 1
iSCSI
discovery, 131
configuration, 131
record types, 131
offload and interface binding, 132
binding/unbinding an iface to a portal, 134
iface (configuring for iSCSI offload), 134
iface configurations, viewing, 132
iface for software iSCSI, 133
iface settings, 132
initiator implementations, 132
offload iSCSI (initiator implementations), 132
software iSCSI, 133
software iSCSI (initiator implementations), 132
viewing available iface configurations, 132
scanning interconnects, 135
software iSCSI, 133
targets, 137
logging in, 137
iSCSI API, 123
iSCSI detection and configuration
storage considerations during installation, 5
iSCSI logical unit, resizing, 138
iSCSI root
iSCSI configuration, 143
issue_lip
fibre channel API, 122
K
known issues
adding/removing
LUN (logical unit number), 140
L
lazy mount/unmount support (autofs version 5)
NFS, 61
levels
RAID, 86
lib directory, 32
limit (xfs_quota expert mode)
XFS, 51
linear RAID
RAID, 86
Linux I/O stack
I/O alignment and size, 109
logging in
iSCSI targets, 137
logical volume, 7
logical volume, editing a
LVM, 18
logical volumes, 7
logical_block_size
  I/O alignment and size, 110
LUKS/dm-crypt, encrypting block devices using
storage considerations during installation, 4
LUN (logical unit number)
  adding/removing, 140
    known issues, 140
    required packages, 140
    rescan-scsi-bus.sh, 140
LVM, 7
  adding a new volume group, 16
  additional resources, 21
  documentation, 21
  editing a logical volume, 18
    explanation of, 7
  extending a volume group, 17
  extents, migration of, 13
  I/O alignment and size, 112
  logical volume, 7
  logical volume, editing a, 18
  logical volumes, 7
  migrating extents, 13
  new hard disk, adding a, 15
  physical volume, 7
  system-config-lvm, 8
  unallocated volumes, 11
  uninitialized entries, 11
LVM2
  explanation of, 8

M
main features
  ext4, 43
  XFS, 49
mdadm (configuring RAID sets)
  RAID, 89
mdraid
  RAID, 88
media directory, 32
migrating extents
  LVM, 13
mirroring
  RAID, 86
mkfs, 26
mkfs.ext4
  ext4, 44
mkfs.xfs
  XFS, 49
mkpart, 25
mnt directory, 32
modifying link loss behavior, 141
  fibre channel, 141
mount (client configuration)
  NFS, 59
    mount settings for encrypted file systems
      encrypted file system, 83
    mounting
      encrypted file system, 83
      ext4, 45
      XFS, 50
    mounting a file system as encrypted
      encrypted file system, 83
    mounting ext3 as ext4
      ext4, 46
    multiple master map entries per autofs mount
      point (autofs version 5)
        NFS, 61

N
  native fibre channel drivers, 122
  network booting service
    diskless systems, 115
  Network File System (see NFS)
  new hard disk, adding a
    LVM, 15
  NFS
    /etc/fstab, 60
    /local/directory (client configuration, mounting), 59
    /remote/export (client configuration, mounting), 59
  additional resources, 76
    installed documentation, 76
    related books, 76
    useful websites, 76
  autofs
    augmenting, 63
    configuration, 62
    LDAP, 64
  autofs version 5, 61
  client
    autofs, 60
    configuration, 59
    mount options, 65
  condrestart, 67
  configuration with firewall, 71
  direct map support (autofs version 5), 61
  enhanced LDAP support (autofs version 5), 61
  FS-Cache, 79
  hostname formats, 72
  how it works, 57
  introducing, 57
  lazy mount/unmount support (autofs version 5), 61
  mount (client configuration), 59
  multiple master map entries per autofs mount
    point (autofs version 5), 61
Index

XFS, 51
proc directory, 33
processing, I/O limit
  overview, 1
project limits (setting)
  XFS, 52
proper nsswitch configuration (autofs version 5), use of
    NFS, 61

queue_if_no_path
  iSCSI configuration, 142
  modifying link loss
    iSCSI configuration, 143
quota (other ext4 file system utilities)
  ext4, 46
quota management
  XFS, 51
quotacheck, 96
quotacheck command
  checking quota accuracy with, 99
quotaoff, 98
quotamon, 98

RAID
  advanced RAID device creation, 90
  Anaconda support, 89
  configuring RAID sets, 89
  dmraid, 89
  dmraid (configuring RAID sets), 89
  explanation of, 85
  Hardware RAID, 85
  hardware RAID controller drivers, 88
  installer support, 89
  level 0, 86
  level 1, 86
  level 4, 86
  level 5, 86
  levels, 86, 86
  linear RAID, 86
  mdadm (configuring RAID sets), 89
  mdraid, 88
  mirroring, 86
  parity, 86
  reasons to use, 85
  Software RAID, 85
  striping, 86
  subsystems of RAID, 88
READ CAPACITY(16)
  I/O alignment and size, 111
required packages
  adding/removing
    LUN (logical unit number), 140
  diskless systems, 115
  FCoE, 129
  rescan-scsi-bus.sh
    adding/removing
      LUN (logical unit number), 140
  resize2fs, 40
  resize2fs (resizing ext4)
    ext4, 46
  resized logical units, resizing, 138
  resizing
    ext4, 46
  resizing an iSCSI logical unit, 138
  resizing multipath device
    resizing online resized logical units, 139
  restoring a backup
    XFS, 54
  rfc2307bis (autofs)
    NFS, 64
  rpcbind, 74
    (see also NFS)
    NFS, 74
    rpcinfo, 74
    status, 67
    rpcinfo, 74
  running sessions, retrieving information about
iSCSI API, 123
running status
   Linux SCSI layer, 144

S
sbin directory, 33
scanning interconnects
   iSCSI, 135
scanning storage interconnects, 130
SCSI command timer
   Linux SCSI layer, 145
SCSI Error Handler
   modifying link loss
   iSCSI configuration, 143
SCSI standards
   I/O alignment and size, 111
   separate partitions (for /home, /opt, /usr/local)
   storage considerations during installation, 4
   server (client configuration, mounting)
      NFS, 59
   setfacl, 102
   setting up a cache
      FS-Cache, 78
   simple mode (xfsrestore)
      XFS, 55
software iSCSI
   iSCSI, 133
software iSCSI (initiator implementations)
   offload and interface binding
   iSCSI, 132
Software RAID (see RAID)
solid state disks
   deployment, 119
   deployment guidelines, 119
   I/O scheduler (tuning), 120
   SSD, 119
   swap (tuning), 120
   throughput classes, 119
   TRIM command, 119
   tuning, 120
   virtual memory (tuning), 120
specific session timeouts, configuring
   iSCSI configuration, 144
srv directory, 33
SSD
   solid state disks, 119
stacking I/O parameters
   I/O alignment and size, 111
stale BIOS RAID metadata
   storage considerations during installation, 4
star, 103
statistical information (tracking)
   FS-Cache, 81
storage access parameters
I/O alignment and size, 109
storage considerations during installation
   advanced path, 3
   auto-partitioning and /home, 3
   basic path, 3
   channel command word (CCW), 4
   DASD and zFCP devices on IBM System z, 4
   DIF/DIX-enabled block devices, 5
   fibre-channel over ethernet (FCoE), 3
   file systems, overview of supported types, 3
   iSCSI detection and configuration, 5
   LUKS/dm-crypt, encrypting block devices
   using, 4
   separate partitions (for /home, /opt, /usr/local), 4
   stale BIOS RAID metadata, 4
   storage device filter interface, 3
   updates, 3
   what's new, 3
storage device filter interface
   storage considerations during installation, 3
   storage interconnects, scanning, 130
   storing automounter maps, using LDAP to store
   (autofs)
      NFS, 63
   stride (specifying stripe geometry)
      ext4, 44
   stripe geometry
      ext4, 44
   stripe-width (specifying stripe geometry)
      ext4, 44
stripping
   RAID, 86
   RAID fundamentals, 85
   su (mkfs.xfs sub-options)
      XFS, 50
subsystenms of RAID
   RAID, 88
suspending
   XFS, 53
   sw (mkfs.xfs sub-options)
      XFS, 50
swap (tuning)
   solid state disks, 120
swap space, 91
   creating, 91
   expanding, 91
   explanation of, 91
   file
      creating, 92, 94
   LVM2
      creating, 92
      extending, 92
      reducing, 93
Index

removing, 94
moving, 94
recommended size, 91
removing, 93
symbolic links in /dev/disk
persistent naming, 124
sys directory, 33
sysconfig directory, 36
sysfs
overview
online storage, 121
sysfs interface (userspace access)
I/O alignment and size, 110
system information
file systems, 29
/dev/shm, 30

T
targets
iSCSI, 137
TCP, using NFS over
NFS, 75
tftp service, configuring
diskless systems, 115
throughput classes
solid state disks, 119
timeouts for a specific session, configuring
iSCSI configuration, 144
tools (for partitioning and other file system functions)
I/O alignment and size, 112
tracking statistical information
FS-Cache, 81
transport
fibre channel API, 121
TRIM command
solid state disks, 119
troubleshooting
online storage, 145
troubleshooting NFS and rpcbind
NFS, 74
tune2fs
converting to ext3 with, 40
reverting to ext2 with, 40
tune2fs (mounting)
ext4, 45
tune2fs (setting up a cache)
FS-Cache, 78
tuning
solid state disks, 120

U
udev
persistent naming, 125
udev rule (timeout)
command timer (SCSI), 145
unallocated volumes
LVM, 11
uninitialized entries
LVM, 11
Universally Unique Identifier (UUID)
persistent naming, 125
updates
storage considerations during installation, 3
quota/quotonoreforce
XFS, 51
userspace access
I/O alignment and size, 110
userspace API files
fibre channel API, 121
usr directory, 34
UUID
persistent naming, 125

V
var directory, 35
var/lib/rpm/ directory, 36
var/spool/up2date/ directory, 36
viewing available iface configurations
offload and interface binding
iSCSI, 132
virtual file system (/proc)
/proc/devices, 37
/proc/filesystems, 37
/proc/mdstat, 37
/proc/mounts, 37
/proc/mounts/, 37
/proc/partitions, 37
virtual memory (tuning)
solid state disks, 120
virtual storage, 147
volume group, 7

W
what’s new
storage considerations during installation, 3
World Wide Identifier (WWID)
persistent naming, 124
write barriers
battery-backed write caches, 106
definition, 105
disabling write caches, 106
enabling/disabling, 105
error messages, 106
ext4, 46
high-end arrays, 107
how write barriers work, 105
importance of write barriers, 105
NFS, 107
XFS, 50
write caches, disabling
write barriers, 106
WWID
  persistent naming, 124

X
XFS
  allocation features, 49
  backup/restoration, 54
  creating, 49
  cumulative mode (xfsrestore), 55
dump levels, 54
  expert mode (xfs_quota), 51
  file system types, 49
  fsync(), 49
  gquota/gqnoenforce, 51
  increasing file system size, 52
  inod64 mount option, 50
  interactive operation (xfsrestore), 55
  limit (xfs_quota expert mode), 51
  main features, 49
  mkfs.xfs, 49
  mounting, 50
  nobarrier mount option, 50
  pquota/pqnoenforce, 51
  project limits (setting), 52
  quota management, 51
  repairing file system, 53
  repairing XFS file systems with dirty logs, 53
  report (xfs_quota expert mode), 51
  simple mode (xfsrestore), 55
  su (mkfs.xfs sub-options), 50
  suspending, 53
  sw (mkfs.xfs sub-options), 50
  uquota/uqnoenforce, 51
  write barriers, 50
  xsfdump, 54
  xsfsprogs, 53
  xfsrestore, 54
  xfs_admin, 55
  xfs_bmap, 55
  xfs_copy, 55
  xfs_db, 55
  xfs.freeze, 53
  xfsdump
  XFS, 54
  xsfsprogs
  XFS, 53
  xfsrestore
  XFS, 54
  xfs_admin
  XFS, 55
  xfs_bmap
  XFS, 55
  xfs_copy
  XFS, 55
  xfs.db
  XFS, 55
  xfs_freeze
  XFS, 53
  xfs.fs
  XFS, 53
  xfs_growfs
  XFS, 52
  xfs.info
  XFS, 55
  xfs_mdrestore
  XFS, 55
  xfs_metadump
  XFS, 55
  xfs_quota
  XFS, 51
  xfs.repair
  XFS, 53