

Oracle Linux 7

Managing File Systems



F32760-08
October 2022



Oracle Linux 7 Managing File Systems,

F32760-08

Copyright © 2022, Oracle and/or its affiliates.

Contents

Preface

Conventions	vii
Documentation Accessibility	vii
Access to Oracle Support for Accessibility	vii
Diversity and Inclusion	vii

1 About File System Management in Oracle Linux

2 Creating and Managing File Systems

Making File Systems	2-1
Mounting File Systems	2-1
About Mount Options	2-3
About the File System Mount Table	2-4
Configuring the Automounter	2-5
Mounting a File Containing a File System Image	2-7
Creating a File System on a File	2-7
Checking and Repairing a File System	2-8
Changing the Frequency of File System Checking	2-9
About Access Control Lists	2-9
Configuring ACL Support	2-10
Setting and Displaying ACLs	2-10
About Disk Quotas	2-12
Enabling Disk Quotas on File Systems	2-12
Assigning Disk Quotas to Users and Groups	2-12
Setting the Grace Period	2-13
Displaying Disk Quotas	2-13
Enabling and Disabling Disk Quotas	2-14
Reporting on Disk Quota Usage	2-14
Maintaining the Accuracy of Disk Quota Reporting	2-14

3 Managing the Btrfs File System

About the Btrfs File System	3-1
Modifying a Btrfs File System	3-1
Compressing and Defragmenting a Btrfs File System	3-2
Resizing a Btrfs File System	3-3
Creating Subvolumes and Snapshots	3-3
Using snapper with Btrfs Subvolumes	3-5
Cloning Virtual Machine Images and Linux Containers	3-7
Using the Send/Receive Feature	3-7
How to Use Send/Receive to Implement Incremental Backups	3-7
Using Quota Groups	3-8
Replacing Devices on a Live File System	3-9
Creating Snapshots of Files	3-9
About the Btrfs root File System	3-9
Creating Snapshots of the root File System	3-11
Mounting Alternate Snapshots as the root File System	3-11
Deleting Snapshots of the root File System	3-12

4 Managing the Ext File System

Converting an Ext2, Ext3, or Ext4 File System to a Btrfs File System	4-1
How to Convert a Non-root File System	4-1
Converting a Non-root Ext2 File System to Ext3	4-2
Converting a root Ext2 File System to Ext3	4-2

5 Managing the XFS File System

About the XFS File System	5-1
About External XFS Journals	5-2
About XFS Write Barriers	5-2
About Lazy Counters	5-3
Installing the XFS Packages	5-3
Creating an XFS File System	5-3
Modifying an XFS File System	5-4
Growing an XFS File System	5-5
Freezing and Unfreezing an XFS File System	5-5
Setting Quotas on an XFS File System	5-6
Setting Project Quotas	5-7
Backing up and Restoring XFS File Systems	5-8
Defragmenting an XFS File System	5-10

Checking and Repairing an XFS File System	5-10
---	------

6 Shared File System Administration

About Shared File Systems	6-1
About NFS	6-1
Configuring an NFS Server	6-1
Mounting an NFS File System	6-4
About Samba	6-5
Configuring a Samba Server	6-5
About Samba Configuration for Windows Workgroups and Domains	6-7
Configuring Samba as a Standalone Server	6-7
Configuring Samba as a Member of an ADS Domain	6-8
Configuring Samba as a Member of a Windows NT4 Security Domain	6-9
Accessing Samba Shares from a Windows Client	6-10
Accessing Samba Shares from an Oracle Linux Client	6-10

7 Managing Oracle Cluster File System Version 2

About OCFS2	7-1
Creating a Local OCFS2 File System	7-2
Installing and Configuring OCFS2	7-2
Preparing a Cluster for OCFS2	7-2
Configuring the Firewall	7-4
Configuring the Cluster Software	7-4
Creating the Configuration File for the Cluster Stack	7-4
Configuring the Cluster Stack	7-7
Configuring the Kernel for Cluster Operation	7-9
Starting and Stopping the Cluster Stack	7-10
Creating OCFS2 volumes	7-10
Mounting OCFS2 Volumes	7-12
Querying and Changing Volume Parameters	7-13
Troubleshooting OCFS2	7-13
Recommended Tools for Debugging	7-13
Mounting the debugfs File System	7-14
Configuring OCFS2 Tracing	7-14
Debugging File System Locks	7-15
Configuring the Behavior of Fenced Nodes	7-16
Use Cases for OCFS2	7-17
Load Balancing	7-17
Oracle Real Application Cluster (RAC)	7-17

Preface

Oracle® Linux 7: Managing File Systems provides information and instructions about administering supported file systems in Oracle Linux 7 systems, including different configuration tasks.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
<i>italic</i>	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
monospace	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at <https://www.oracle.com/corporate/accessibility/>.

For information about the accessibility of the Oracle Help Center, see the Oracle Accessibility Conformance Report at <https://www.oracle.com/corporate/accessibility/templates/t2-11535.html>.

Access to Oracle Support for Accessibility

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit <https://www.oracle.com/corporate/accessibility/learning-support.html#support-tab>.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry

standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

1

About File System Management in Oracle Linux

This chapter briefly describes local file system management in Oracle Linux. It also lists supported file systems in the operating system.

Oracle Linux supports a large number of local file system types that you can configure on block devices, including:

btrfs

Btrfs is a copy-on-write file system that is designed to address the expanding scalability requirements of large storage subsystems. It supports snapshots, a roll-back capability, checksum functionality for data integrity, transparent compression, and integrated logical volume management.

The maximum supported file or file system size is 50 TB.

For more information, see [About the Btrfs File System](#).

ext3

The ext3 file system includes journaling capabilities to improve reliability and availability. Consistency checks after a power failure or an uncontrolled system shutdown are unnecessary. ext2 file systems are upgradeable to ext3 without reformatting.

See [Converting a Non-root Ext2 File System to Ext3](#) and [Converting a root Ext2 File System to Ext3](#).

The maximum supported file and file system sizes are 2 TB and 16 TB.

ext4

In addition to the features of ext3, the ext4 file system supports extents (contiguous physical blocks), pre-allocation, delayed allocation, faster file system checking, more robust journaling, and other enhancements.

The maximum supported file or file system size is 50 TB.

ocfs2

Although intended as a general-purpose, high-performance, high-availability, shared-disk file system intended for use in clusters, it is possible to use Oracle Cluster File System version 2 (OCFS2) as a standalone, non-clustered file system.

Although it might seem that there is no benefit in mounting OCFS2 locally as compared to alternative file systems such as ext4 or btrfs, you can use the `reflink` command with OCFS2 to create copy-on-write clones of individual files in a similar way to using the `cp --reflink` command with the btrfs file system. Typically, such clones allow you to save disk space when storing multiple copies of very similar files, such as VM images or Linux Containers. In addition, mounting a local OCFS2 file system allows you to subsequently migrate it to a cluster file system without requiring any conversion.

See [Creating a Local OCFS2 File System](#).

The maximum supported file or file system size is 16 TB.

vfat

The vfat file system (also known as FAT32) was originally developed for MS-DOS. It does not support journaling and lacks many of the features that are available with other file system

types. It is mainly used to exchange data between Microsoft Windows and Oracle Linux systems.

The maximum supported file size or file system size is 2 GB.

xfs

XFS is a high-performance journaling file system, which provides high scalability for I/O threads, file system bandwidth, file and file system size, even when the file system spans many storage devices.

The maximum supported file and file system sizes are 16 TB and 500 TB respectively. For more information, see [About the XFS File System](#).

To see what file system types your system supports, use the following command:

```
ls /sbin/mkfs.*
/sbin/mkfs.btrfs   /sbin/mkfs.ext3   /sbin/mkfs.msdos
/sbin/mkfs.cramfs /sbin/mkfs.ext4   /sbin/mkfs.vfat
/sbin/mkfs.ext2   /sbin/mkfs.ext4dev /sbin/mkfs.xfs
```

These executables are used to make the file system type specified by their extension. `mkfs.msdos` and `mkfs.vfat` are alternate names for `mkdosfs`. `mkfs.cramfs` creates a compressed ROM, read-only cramfs file system for use by embedded or small-footprint systems.

2

Creating and Managing File Systems

This chapter describes how to create, mount, check, and repair file systems, how to configure Access Control Lists, how to configure and manage disk quotas.

Making File Systems

The `mkfs` command build a file system on a block device:

```
sudo mkfs [options] device
```

`mkfs` is a front end for builder utilities in `/sbin` such as `mkfs.ext4`. You can use either the `mkfs` command with the `-t fstype` option or the builder utility to specify the type of file system to build. For example, the following commands are equivalent ways of creating an `ext4` file system with the label `Projects` on the device `/dev/sdb1`:

```
sudo mkfs -t ext4 -L Projects /dev/sdb1
sudo mkfs.ext4 -L Projects /dev/sdb1
```

If you do not specify the file system type to `makefs`, it creates an `ext2` file system.

To display the type of a file system, use the `blkid` command:

```
sudo blkid /dev/sdb1

/dev/sdb1: UUID="ad8113d7-b279-4da8-b6e4-cfba045f66ff" TYPE="ext4" LABEL="Projects"
```

The `blkid` command also display information about the device such as its UUID and label.

Each file system type supports a number of features that you can enable or disable by specifying additional options to `mkfs` or the build utility. For example, you can use the `-J` option to specify the size and location of the journal used by the `ext3` and `ext4` file system types.

For more information, see the `blkid(8)`, `mkfs(8)`, and `mkfs.fstype(8)` manual pages.

Mounting File Systems

To access a file system's contents, you must attach its block device to a mount point in the directory hierarchy. You can use the `mkdir` command to create a directory for use as a mount point, for example:

```
mkdir /var/projects
```

You can use an existing directory as a mount point, but its contents are hidden until you unmount the overlying file system.

The `mount` command attaches the device containing the file system to the mount point:

```
sudo mount [options] device mount_point
```

You can specify the device by its name, UUID, or label. For example, the following commands are equivalent ways of mounting the file system on the block device `/dev/sdb1`:

```
sudo mount /dev/sdb1 /var/projects
sudo mount UUID="ad8113d7-b279-4da8-b6e4-cfba045f66ff" /var/projects
sudo mount LABEL="Projects" /var/projects
```

If you do not specify any arguments, `mount` displays all file systems that the system currently has mounted, for example:

```
sudo mount

/dev/mapper/vg_host01-lv_root on / type ext4 (rw)
...
```

In this example, the LVM logical volume `/dev/mapper/vg_host01-lv_root` is mounted on `/`. The file system type is `ext4` and is mounted for both reading and writing. (You can also use the command `cat /proc/mounts` to display information about mounted file systems.)

The `df` command displays information about how much space remains on mounted file systems, for example:

```
sudo df -h

Filesystem                Size  Used Avail Use% Mounted on
/dev/mapper/vg_host01-lv_root 36G   12G   22G   36% /
...
```

You can use the `-B` (bind) option to the `mount` command to attach a block device at multiple mount points. You can also remount part of a directory hierarchy, which need not be a complete file system, somewhere else. For example, the following command mounts `/var/projects/project1` on `/mnt`:

```
sudo mount -B /var/projects/project1 /mnt
```

Each directory hierarchy acts as a mirror of the other. The same files are accessible in either location, although any submounts are not replicated. These mirrors do not provide data redundancy.

You can also mount a file over another file, for example:

```
touch /mnt/foo
mount -B /etc/hosts /mnt/foo
```

In this example, `/etc/hosts` and `/mnt/foo` represent the same file. The existing file that acts as a mount point is not accessible until you unmount the overlying file.

The `-B` option does not recursively attach any submounts below a directory hierarchy. To include submounts in the mirror, use the `-R` (recursive bind) option instead.

When you use `-B` or `-R`, the file system `mount` options remain the same as those for the original mount point. To modify, the mount options, use a separate `remount` command, for example:

```
sudo mount -o remount,ro /mnt/foo
```

You can mark the submounts below a mount point as being shared, private, or secondary (slave):

mount --make-shared *mount_point*

Any mounts or unmounts below the specified mount point propagate to any mirrors that you create, and this mount hierarchy reflects mounts or unmount changes that you make to other mirrors.

mount --make-private *mount_point*

Any mounts or unmounts below the specified mount point do not propagate to other mirrors, nor does this mount hierarchy reflect mounts or unmount changes that you make to other mirrors.

mount --make-slave *mount_point*

Any mounts or unmounts below the specified mount point do not propagate to other mirrors, but this mount hierarchy does reflect mounts or unmount changes that you make to other mirrors.

To prevent a mount from being mirrored by using the `-B` or `-R` options, mark its mount point as being unbindable:

```
sudo mount --make-unbindable mount_point
```

To move a mounted file system, directory hierarchy, or file between mount points, use the `-M` option, for example:

```
touch /mnt/foo  
mount -M /mnt/foo /mnt/bar
```

To unmount a file system, use the `umount` command, for example:

```
sudo umount /var/projects
```

Alternatively, you can specify the block device provided that it is mounted on only one mount point.

For more information, see the `mount(8)` and `umount(8)` manual pages.

About Mount Options

To modify the behavior of `mount`, use the `-o` flag followed by a comma-separated list of options or specify the options in the `/etc/fstab` file. The following are some of the options that are available:

auto

Allows the file system to be mounted automatically by using the `mount -a` command.

exec

Allows the execution of any binary files located in the file system.

loop

Uses a loop device (`/dev/loop*`) to mount a file that contains a file system image. See [Mounting a File Containing a File System Image](#), [Creating a File System on a File](#), and the `losetup(8)` manual page.

 **Note:**

The default number of available loop devices is 8. You can use the kernel boot parameter `max_loop=N` to configure up to 255 devices. Alternatively, add the following entry to `/etc/modprobe.conf`:

```
options loop max_loop=N
```

In the previous example, *N* is the number of loop devices that you require (from 0 to 255), and reboot the system.

noauto

Disallows the file system from being mounted automatically by using `mount -a`.

noexec

Disallows the execution of any binary files located in the file system.

nouser

Disallows any user other than `root` from mounting or unmounting the file system.

remount

Remounts the file system if it is already mounted. You would usually combine this option with another option such as `ro` or `rw` to change the behavior of a mounted file system.

ro

Mounts a file system as read-only.

rw

Mounts a file system for reading and writing.

user

Allows any user to mount or unmount the file system.

For example, mount `/dev/sdd1` as `/test` with read-only access and only `root` permitted to mount or unmount the file system:

```
sudo mount -o nouser,ro /dev/sdd1 /test
```

Mount an ISO image file on `/mount/cdrom` with read-only access by using the loop device:

```
sudo mount -o ro,loop ./OracleLinux-R6-U1-Server-x86_64-dvd.iso /media/cdrom
```

Remount the `/test` file system with both read and write access, but do not permit the execution of any binary files that are located in the file system:

```
sudo mount -o remount,rw,noexec /test
```

About the File System Mount Table

The `/etc/fstab` file contains the file system mount table, and provides all the information that the `mount` command needs to mount block devices or to implement binding of mounts. If you add a file system, create the appropriate entry in `/etc/fstab`

to ensure that the file system is mounted at boot time. The following are sample entries from `/etc/fstab`:

```
/dev/sda1      /boot  ext4    defaults 1 2
/dev/sda2      /       ext4    defaults 1 1
/dev/sda3      swap   swap    defaults 0 0
```

The first field is the device to mount specified by the device name, UUID, or device label, or the specification of a remote file system. A UUID or device label is preferable to a device name if the device name could change, for example:

```
LABEL=Projects /var/projects ext4 defaults 1 2
```

The second field is either the mount point for a file system or `swap` to indicate a swap partition.

The third field is the file system type, for example, `ext4` or `swap`.

The fourth field specifies any mount options.

The fifth column is used by the `dump` command. A value of 1 means dump the file system; 0 means the file system does not need to be dumped.

The sixth column is used by the file system checker, `fsck`, to determine in which order to perform file system checks at boot time. The value should be 1 for the root file system, 2 for other file systems. A value of 0 skips checking, as is appropriate for swap, file systems that are not mounted at boot time, or for binding of existing mounts.

For bind mounts, only the first four fields are specified, for example:

```
pathmount_point none bind
```

The first field specifies the path of the file system, directory hierarchy, or file that is to be mounted on the mount point specified by the second field. The mount point must be a file if the path specifies a file; otherwise, it must be a directory. The third and fourth fields are specified as `none` and `bind`.

For more information, see the `fstab(5)` manual page.

Configuring the Automounter

The automounter mounts file systems when they are accessed, rather than maintaining connections for those mounts at all times. When a file system becomes inactive for more than a certain period of time, the automounter unmounts it. Using automounting frees up system resources and improves system performance.

The automounter consists of two components: the `autofs` kernel module and the `automount` user-space daemon.

To configure a system to use automounting:

1. Install the `autofs` package and any other packages that are required to support remote file systems:

```
sudo yum install autofs
```

2. Edit the `/etc/auto.master` configuration file to define map entries. Each map entry specifies a mount point and a map file that contains definitions of the remote file systems that can be mounted, for example:

```

/-          /etc/auto.direct
/misc       /etc/auto.misc
/net        -hosts

```

Here, the `/-`, `/misc`, and `/net` entries are examples of a direct map, an indirect map, and a host map respectively. Direct map entries always specify `/-` as the mount point. Host maps always specify the keyword `-hosts` instead of a map file.

A direct map contains definitions of directories that are automounted at the specified absolute path. In the example, the `auto.direct` map file might contain an entry such as:

```

/usr/man    -fstype=nfs,ro,soft          host01:/usr/man

```

This entry mounts the file system `/usr/man` exported by `host01` using the options `ro` and `soft`, and creates the `/usr/man` mount point if it does not already exist. If the mount point already exists, the mounted file system hides any existing files that it contains.

As the default file system type is NFS, the previous example can be shortened to read:

```

/usr/man    -ro,soft                    host01:/usr/man

```

An indirect map contains definitions of directories (*keys*) that are automounted relative to the mount point (`/misc`) specified in `/etc/auto.master`. In the example, the `/etc/auto.misc` map file might contain entries such as the following:

```

xyz        -ro,soft                    host01:/xyz
cd         -fstype=iso9600,ro,nosuid,nodev  :/dev/cdrom
abc       -fstype=ext3                  :/dev/hda1
fenetres   -fstype=cifs,credentials=credfile  ://fenetres/c

```

The `/misc` directory must already exist, but the automounter creates a mount point for the keys `xyz`, `cd`, and so on if they do not already exist, and removes them when it unmounts the file system. For example, entering a command such as `ls /misc/xyz` causes the automounter to mount the `/xyz` directory exported by `host01` as `/misc/xyz`.

The `cd` and `abc` entries mount local file systems: an ISO image from the CD-ROM drive on `/misc/cd` and an `ext3` file system from `/dev/hda1` on `/misc/abc`. The `fenetres` entry mounts a Samba share as `/misc/fenetres`.

If a host map entry exists and a command references an NFS server by name relative to the mount point (`/net`), the automounter mounts all directories that the server exports below a subdirectory of the mount point named for the server. For example, the command `cd /net/host03` causes the automounter to mount all exports from `host03` below the `/net/host03` directory. By default, the automounter uses the mount options `nosuid,nodev,intr` unless you override the options in the host map entry, for example:

```

/net        -hosts    -suid,dev,nointr

```


 **Note:**

The name of the NFS server must be resolvable to an IP address in DNS or in the `/etc/hosts` file.

For more information, including details of using maps with NIS, NIS+, and LDAP, see the `hosts.master(5)` manual page.

3. Start the `autofs` service, and configure the service to start following a system reboot:

```
sudo systemctl stat autofs
sudo systemctl enable autofs
```

You can configure various settings for `autofs` in `/etc/sysconfig/autofs`, such as the idle timeout value after which a file system is automatically unmounted.

If you modify `/etc/auto.master` or `/etc/sysconfig/autofs`, restart the `autofs` service to make it re-read these files:

```
sudo systemctl restart autofs
```

For more information, see the `automount(8)`, `autofs(5)`, and `auto.master(5)` manual pages.

Mounting a File Containing a File System Image

A loop device allows you to access a file as a block device. For example, to mount a file that contains a DVD ISO image on the directory mount point `/ISO`:

```
sudo mount -t iso9660 -o ro,loop /var/ISO_files/V33411-01.iso /ISO
```

If required, create a permanent entry for the file system in `/etc/fstab`:

```
/var/ISO_files/V33411-01.iso          /ISO          iso9660      ro,loop      0 0
```

Creating a File System on a File

To create a file system on a file within another file system:

1. Create an empty file of the required size, for example:

```
sudo dd if=/dev/zero of=/fsfile bs=1024 count=1000000

1000000+0 records in
1000000+0 records out
1024000000 bytes (1.0 GB) copied, 8.44173 s, 121 MB/s
```

2. Create a file system on the file:

```
sudo mkfs.ext4 -F /fsfile

mke2fs 1.41.12 (17-May-2010)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
62592 inodes, 250000 blocks
```

```

12500 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=260046848
8 block groups
32768 blocks per group, 32768 fragments per group
7824 inodes per group
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376

Writing inode tables: done
Creating journal (4096 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 33 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.

```

3. Mount the file as a file system by using a loop device:

```
sudo mount -o loop /fsfile /mnt
```

The file appears as a normal file system:

```

sudo mount

...
/fsfile on /mnt type ext4 (rw,loop=/dev/loop0)
# df -h
Filesystem      Size  Used Avail Use% Mounted on
...
/fsfile          962M  18M  896M   2% /mnt

```

If required, create a permanent entry for the file system in `/etc/fstab`:

```
/fsfile          /mnt          ext4          rw,loop       0 0
```

Checking and Repairing a File System

The `fsck` utility checks and repairs file systems. For file systems other than `/` (root) and `/boot`, `mount` invokes file system checking if more than a certain number of mounts have occurred or more than 180 days have elapsed without checking having been performed. You might want to run `fsck` manually if a file system has not been checked for several months.

NOT_SUPPORTED:

Running `fsck` on a mounted file system can corrupt the file system and cause data loss.

To check and repair a file system:

1. Unmount the file system:

```
sudo umount filesystem
```

2. Use the `fsck` command to check the file system:

```
sudo fsck [-y] filesystem
```

filesystem be a device name, a mount point, or a label or UUID specifier, for example:

```
sudo fsck UUID=ad8113d7-b279-4da8-b6e4-cfba045f66ff
```

By default, `fsck` prompts you to choose whether it should apply a suggested repair to the file system. If you specify the `-y` option, `fsck` assumes a `yes` response to all such questions.

For the `ext2`, `ext3`, and `ext4` file system types, other commands that are used to perform file system maintenance include `dumpe2fs` and `debugfs`. `dumpe2fs` prints super block and block group information for the file system on a specified device. `debugfs` is an interactive file system debugger that requires expert knowledge of the file system architecture. Similar commands exist for most file system types and also require expert knowledge.

For more information, see the `fsck(8)` manual page.

Changing the Frequency of File System Checking

To change the number of mounts before the system automatically checks the file system for consistency:

```
sudo tune2fs -c mount_count device
```

In the previous example, *device* specifies the block device that corresponds to the file system.

A *mount_count* of 0 or -1 disables automatic checking based on the number of mounts.

Tip:

Specifying a different value of *mount_count* for each file system reduces the probability that the system checks all the file systems at the same time.

To specify the maximum interval between file system checks:

```
sudo tune2fs -i interval[unit] device
```

The *unit* can be `d`, `w`, or `m` for days, weeks, or months. The default unit is `d` for days. An *interval* of 0 disables checking that is based on the time that has elapsed since the last check. Even if the interval is exceeded, the file system is not checked until it is next mounted.

For more information, see the `tune2fs(8)` manual page.

About Access Control Lists

POSIX Access Control Lists (ACLs) provide a richer access control model than traditional UNIX Discretionary Access Control (DAC) that sets read, write, and execute permissions for the owner, group, and all other system users. You can configure ACLs that define access rights for more than just a single user or group, and specify rights for programs, processes, files, and directories. If you set a default ACL on a directory, its descendents inherit the same rights automatically. You can use ACLs with `btrfs`, `ext3`, `ext4`, `OCFS2`, and `XFS` file systems and with mounted NFS file systems.

An ACL consists of a set of rules that specify how a specific user or group can access the file or directory with which the ACL is associated. A regular ACL entry specifies access

information for a single file or directory. A default ACL entry is set on directories only, and specifies default access information for any file within the directory that does not have an access ACL.

Configuring ACL Support

To enable ACL support:

1. Install the `acl` package:

```
sudo yum install acl
```

2. Edit `/etc/fstab` and change the entries for the file systems with which you want to use ACLs so that they include the appropriate option that supports ACLs, for example:

```
LABEL=/work      /work      ext4      acl      0 0
```

For mounted Samba shares, use the `cifsacl` option instead of `acl`.

3. Remount the file systems, for example:

```
sudo mount -o remount /work
```

Setting and Displaying ACLs

To add or modify the ACL rules for file, use the `setfacl` command:

```
sudo setfacl -m rules file ...
```

The rules take the following forms:

[d:]u: *user*[: *permissions*]

Sets the access ACL for the user specified by name or user ID. The permissions apply to the owner if a user is not specified.

[d:]g: *group*[: *permissions*]

Sets the access ACL for a group specified by name or group ID. The permissions apply to the owning group if a group is not specified.

[d:]m[:][: *permissions*]

Sets the effective rights mask, which is the union of all permissions of the owning group and all of the user and group entries.

[d:]o[:][: *permissions*]

Sets the access ACL for other (everyone else to whom no other rule applies).

The permissions are `r`, `w`, and `x` for read, write, and execute as used with `chmod`.

The `d:` prefix is used to apply the rule to the default ACL for a directory.

To display a file's ACL, use the `getfacl` command, for example:

```
sudo getfacl foofile

# file: foofile
# owner: bob
# group: bob
user::rw-
```

```
user::fiona:r--
user::jack:rw-
user::jill:rw-
group::r--
mask::r--
other::r--
```

If extended ACLs are active on a file, the `-l` option to `ls` displays a plus sign (+) after the permissions, for example:

```
ls -l foofile

-rw-r--r--+ 1 bob bob 105322 Apr 11 11:02 foofile
```

The following are examples of how to set and display ACLs for directories and files.

Grant read access to a file or directory by a user.

```
sudo setfacl -m u:user:r file
```

Display the name, owner, group, and ACL for a file or directory.

```
sudo getfacl file
```

Remove write access to a file for all groups and users by modifying the effective rights mask rather than the ACL.

```
sudo setfacl -m m::rx file
```

The `-x` option removes rules for a user or group.

Remove the rules for a user from the ACL of a file.

```
sudo setfacl -x u:user file
```

Remove the rules for a group from the ACL of a file.

```
sudo setfacl -x g:group file
```

The `-b` option removes all extended ACL entries from a file or directory.

```
sudo setfacl -b file
```

Copy the ACL of file *f1* to file *f2*.

```
sudo getfacl f1 | setfacl --set-file=- f2
```

Set a default ACL of read and execute access for other on a directory:

```
sudo setfacl -m d:o:rx directory
```

Promote the ACL settings of a directory to default ACL settings that can be inherited.

```
sudo getfacl --access directory | setfacl -d -M- directory
```

The `-k` option removes the default ACL from a directory.

```
sudo setfacl -k directory
```

For more information, see the `acl(5)`, `setfacl(1)`, and `getfacl(1)` manual pages.

About Disk Quotas

**Note:**

For information about how to configure quotas for the XFS file system, see [Setting Quotas on an XFS File System](#).

You can set disk quotas to restrict the amount of disk space (*blocks*) that users or groups can use, to limit the number of files (*inodes*) that users or groups can create, and to notify you when usage is reaching a specified limit. A hard limit specifies the maximum number of blocks or inodes available to a user or group on the file system. Users or groups can exceed a soft limit for a period of time known as a *grace period*.

Enabling Disk Quotas on File Systems

To enable user or group disk quotas on a file system:

1. Install or update the quota package:

```
sudo yum install quota
```

2. Include the `usrquota` or `grpquota` options in the file system's `/etc/fstab` entry, for example:

```
/dev/sdb1      /home          ext4    usrquota,grpquota  0 0
```

3. Remount the file system:

```
sudo mount -o remount /home
```

4. Create the quota database files:

```
sudo quotacheck -cug /home
```

This command creates the files `aquota.user` and `aquota.group` in the root of the file system (`/home` in this example).

For more information, see the `quotacheck(8)` manual page.

Assigning Disk Quotas to Users and Groups

To configure the disk quota for a user:

1. Enter the following command for a user:

```
sudo edquota username
```

or for a group:

```
sudo edquota -g group
```

The command opens a text file in the default editor defined by the `EDITOR` environment variable where you can specify the limits for the user or group, for example:

```
Disk quotas for user guest (uid 501)
Filesystem blocks soft hard inodes soft hard
/dev/sdb1 10325 0 0 1054 0 0
```

The `blocks` and `inodes` entries show the user's currently usage on a file system.

 **Tip:**

Setting a limit to 0 disables quota checking and enforcement for the corresponding `blocks` or `inodes` category.

2. Edit the soft and hard block limits for number of blocks and inodes, and save and close the file.

Alternatively, you can use the `setquota` command to configure quota limits from the command-line. The `-p` option allows you to apply quota settings from one user or group to another user or group.

For more information, see the `edquota(8)` and `setquota(8)` manual pages.

Setting the Grace Period

To configure the grace period for soft limits:

1. Enter the following command:

```
sudo edquota -t
```

The command opens a text file opens in the default editor defined by the `EDITOR` environment variable, where you can specify the grace period, for example:

```
Grace period before enforcing soft limits for users:
Time units may be: days, hours, minutes, or seconds
Filesystem      Block grace period   Inode grace period
/dev/sdb1       7days                7days
```

2. Edit the grace periods for the soft limits on the number of blocks and inodes, and save and close the file.

For more information, see the `edquota(8)` manual page.

Displaying Disk Quotas

To display a user's disk usage:

```
sudo quota username
```

To display a group's disk usage:

```
sudo quota -g group
```

To display information about file systems where usage is over the quota limits:

```
sudo quota -q
```

Users can also use the `quota` command to display their own and their group's usage.

For more information, see the `quota(1)` manual page.

Enabling and Disabling Disk Quotas

To disable disk quotas for all users, groups on a specific file system:

```
sudo quotaoff -guv filesystem
```

To disable disk quotas for all users, groups, and file systems:

```
sudo quotaoff -aguv
```

To re-enable disk quotas for all users, groups, and file systems:

```
sudo quotaon -aguv
```

For more information, see the `quotaon(1)` manual page.

Reporting on Disk Quota Usage

To display the disk quota usage for a file system:

```
sudo repquota filesystem
```

To display the disk quota usage for all file systems:

```
sudo repquota -a
```

For more information, see the `repquota(8)` manual page.

Maintaining the Accuracy of Disk Quota Reporting

Uncontrolled system shutdowns can lead to inaccuracies in disk quota reports.

To rebuild the quota database for a file system:

1. Disable disk quotas for the file system:

```
sudo quotaoff -guv filesystem
```

2. Unmount the file system:

```
sudo umount filesystem
```

3. Enter the following command to rebuild the quota databases:

```
sudo quotacheck -guv filesystem
```

4. Mount the file system:

```
sudo mount filesystem
```

5. Enable disk quotas for the file system:

```
sudo quotaon -guv filesystem
```

For more information, see the `quotacheck(8)` manual page.

3

Managing the Btrfs File System

This chapter describes tasks for administering the Btrfs file system in Oracle Linux.

About the Btrfs File System

The btrfs file system is designed to meet the expanding scalability requirements of large storage subsystems. As the btrfs file system uses B-trees in its implementation, its name derives from the name of those data structures, although it is not a true acronym. A B-tree is a tree-like data structure that enables file systems and databases to efficiently access and update large blocks of data no matter how large the tree grows.

The btrfs file system provides the following important features:

- Copy-on-write functionality allows you to create both readable and writable snapshots, and to roll back a file system to a previous state, even after you have converted it from an `ext3` or `ext4` file system.
- Checksum functionality ensures data integrity.
- Transparent compression saves disk space.
- Transparent defragmentation improves performance.
- Integrated logical volume management allows you to implement RAID 0, RAID 1, or RAID 10 configurations, and to dynamically add and remove storage capacity.



Note:

Configuring a swap file on a btrfs file system is not supported.

You can find more information about the btrfs file system at https://btrfs.wiki.kernel.org/index.php/Main_Page.

Modifying a Btrfs File System

The following table shows how you can use the `btrfs` command to add or remove devices, and to rebalance the layout of the file system data and metadata across the devices.

Command	Description
<code>btrfs device adddevicemountpoint</code>	Add a device to the file system that is mounted on the specified mount point. For example: <code>btrfs device add /dev/sdd /myfs</code>

Command	Description
<code>btrfs device delete <i>devicemountpoint</i></code>	Remove a device from a mounted file system. For example: <code>btrfs device delete /dev/sde /myfs</code>
<code>btrfs device delete <i>missingmountpoint</i></code>	Remove a failed device from the file system that is mounted in degraded mode. For example: <code>btrfs device remove missing /myfs</code> To mount a file system in degraded mode, specify the <code>-o degraded</code> option to the <code>mount</code> command. For a RAID configuration, if the number of devices would fall below the minimum number that are required, you must add the replacement device before removing the failed device.
<code>btrfs filesystem balance <i>mountpoint</i></code>	After adding or removing devices, redistribute the file system data and metadata across the available devices.

Compressing and Defragmenting a Btrfs File System

You can compress a btrfs file system to increase its effective capacity, and you can defragment it to increase I/O performance.

To enable compression of a btrfs file system, specify one of the following `mount` options:

Mount Option	Description
<code>compress=lzo</code>	Use LZO compression.
<code>compress=zlib</code>	Use zlib compression.

LZO offers a better compression ratio, while zlib offers faster compression.

You can also compress a btrfs file system at the same time that you defragment it.

To defragment a btrfs file system, use the following command:

```
sudo btrfs filesystem defragment filesystem_name
```

To defragment a btrfs file system and compress it at the same time:

```
sudo btrfs filesystem defragment -c filesystem_name
```

You can also defragment, and optionally compress, individual file system objects, such as directories and files, within a btrfs file system.

```
# btrfs filesystem defragment [-c] file_name ...
```

 **Note:**

You can set up automatic defragmentation by specifying the `autodefrag` option when you mount the file system. However, automatic defragmentation is not recommended for large databases or for images of virtual machines.

Defragmenting a file or a subvolume that has a copy-on-write copy results breaks the link between the file and its copy. For example, if you defragment a subvolume that has a snapshot, the disk usage by the subvolume and its snapshot will increase because the snapshot is no longer a copy-on-write image of the subvolume.

Resizing a Btrfs File System

You can use the `btrfs` command to increase the size of a mounted btrfs file system if there is space on the underlying devices to accommodate the change, or to decrease its size if the file system has sufficient available free space. The command does not have any effect on the layout or size of the underlying devices.

For example, to increase the size of `/mybtrfs1` by 2 GB:

```
sudo btrfs filesystem resize +2g /mybtrfs1
```

Decrease the size of `/mybtrfs2` by 4 GB:

```
sudo btrfs filesystem resize -4g /mybtrfs2
```

Set the size of `/mybtrfs3` to 20 GB:

```
sudo btrfs filesystem resize 20g /mybtrfs3
```

Creating Subvolumes and Snapshots

The top level of a btrfs file system is a subvolume consisting of a named b-tree structure that contains directories, files, and possibly further btrfs subvolumes that are themselves named b-trees that contain directories and files, and so on. To create a subvolume, change directory to the position in the btrfs file system where you want to create the subvolume and enter the following command:

```
sudo btrfs subvolume create subvolume_name
```

Snapshots are a type of subvolume that records the contents of their parent subvolumes at the time that you took the snapshot. If you take a snapshot of a btrfs file system and do not write to it, the snapshot records the state of the original file system and forms a stable image from which you can make a backup. If you make a snapshot writable, you can treat it as an alternate version of the original file system. The copy-on-write functionality of btrfs file system means that snapshots are quick to create, and consume very little disk space initially.

 **Note:**

Taking snapshots of a subvolume is not a recursive process. If you create a snapshot of a subvolume, every subvolume or snapshot that the subvolume contains is mapped to an empty directory of the same name inside the snapshot.

The following table shows how to perform some common snapshot operations:


Command	Description
<code>btrfs subvolume snapshot <i>pathname</i> <i>pathname</i> / <i>snapshot_path</i></code>	Create a snapshot <i>snapshot_path</i> of a parent subvolume or snapshot specified by <i>pathname</i> . For example: <pre>btrfs subvolume snapshot /mybtrfs /mybtrfs/snapshot1</pre>
<code>btrfs subvolume list <i>pathname</i></code>	List the subvolumes or snapshots of a subvolume or snapshot specified by <i>pathname</i> . For example: <pre>btrfs subvolume list /mybtrfs</pre>
<code>btrfs subvolume set-default <i>ID</i> <i>pathname</i></code>	By default, mount the snapshot or subvolume specified by its ID instead of the parent subvolume. For example: <pre>btrfs subvolume set-default 4 /mybtrfs</pre>
<code>btrfs subvolume get-default <i>pathname</i></code>	Displays the ID of the default subvolume that is mounted for the specified subvolume. For example: <pre>btrfs subvolume get-default /mybtrfs</pre>

 **Note:**

You can use this command to determine the ID of a subvolume or snapshot.

You can mount a btrfs subvolume as though it were a disk device. If you mount a snapshot instead of its parent subvolume, you effectively roll back the state of the file system to the time that the snapshot was taken. By default, the operating system mounts the parent btrfs volume, which has an ID of 0, unless you use `set-default` to change the default subvolume. If you set a new default subvolume, the system will mount that subvolume instead in future. You can override the default setting by specifying either of the following `mount` options:

Mount Option	Description
<code>subvalid=<i>snapshot-ID</i></code>	Mount the subvolume or snapshot specified by its subvolume ID instead of the default subvolume.
<code>subvol=<i>pathname/snapshot_path</i></code>	Mount the subvolume or snapshot specified by its pathname instead of the default subvolume.

 **Note:**

The subvolume or snapshot must be located in the root of the btrfs file system.

When you have rolled back a file system by mounting a snapshot, you can take snapshots of the snapshot itself to record its state.

When you no longer require a subvolume or snapshot, use the following command to delete it:

```
sudo btrfs subvolume delete subvolume_path
```



Note:

Deleting a subvolume deletes all subvolumes that are below it in the b-tree hierarchy. For this reason, you cannot remove the topmost subvolume of a btrfs file system, which has an ID of 0.

For details of how to use the `snapper` command to create and manage btrfs snapshots, see [Using snapper with Btrfs Subvolumes](#).

Using snapper with Btrfs Subvolumes

You can use the `snapper` utility to create and manage snapshots of btrfs subvolumes.

To set up the `snapper` configuration for an existing mounted btrfs subvolume:

```
sudo snapper -c config_name create-config -f btrfs fs_name
```

Here `config_name` is the name of the configuration and `fs_name` is the path of the mounted btrfs subvolume. The command adds an entry for `config_name` to `/etc/sysconfig/snapper`, creates the configuration file `/etc/snapper/configs/config_name`, and sets up a `.snapshots` subvolume for the snapshots.

For example, the following command sets up the `snapper` configuration for a btrfs root file system:

```
sudo snapper -c root create-config -f btrfs /
```

By default, `snapper` sets up a `cron.hourly` job to create snapshots in the `.snapshot` subdirectory of the subvolume and a `cron.daily` job to clean up old snapshots. You can edit the configuration file to disable or change this behavior. For more information, see the `snapper-configs(5)` manual page.

There are three types of snapshot that you can create using `snapper`:

post

You use a *post snapshot* to record the state of a subvolume after a modification. A post snapshot should always be paired with a *pre snapshot* that you take immediately before you make the modification.

pre

You use a *pre snapshot* to record the state of a subvolume before a modification. A pre snapshot should always be paired with a *post snapshot* that you take immediately after you have completed the modification.

single

You can use a single snapshot to record the state of a subvolume but it does not have any association with other snapshots of the subvolume.

For example, the following commands create pre and post snapshots of a subvolume:

```
sudo snapper -c config_name create -t pre -p N
... Modify the subvolume's contents...
sudo snapper -c config_name create -t post --pre-num N -p N'
```

The `-p` option causes `snapper` to display the number of the snapshot so that you can reference it when you create the post snapshot or when you compare the contents of the pre and post snapshots.

To display the files and directories that have been added, removed, or modified between the pre and post snapshots, use the `status` subcommand:

```
sudo snapper -c config_name status N .. N'
```

To display the differences between the contents of the files in the pre and post snapshots, use the `diff` subcommand:

```
sudo snapper -c config_name diff N .. N'
```

To list the snapshots that exist for a subvolume:

```
sudo snapper -c config_name list
```

To delete a snapshot, specify its number to the `delete` subcommand:

```
sudo snapper -c config_name delete N''
```

To undo the changes in the subvolume from post snapshot `N'` to pre snapshot `N`:

```
sudo snapper -c config_name undochange N..N'
```

For more information, see the `snapper(8)` manual page.

Cloning Virtual Machine Images and Linux Containers

You can use a btrfs file system to provide storage space for virtual machine images and Linux Containers. The ability to quickly clone files and create snapshots of directory structures makes btrfs an ideal candidate for this purpose. For details of how to use the snapshot feature of btrfs to implement Linux Containers, see Linux Containers in [Oracle Linux 7: Working With LXC](#).

Using the Send/Receive Feature

Note:

The send/receive feature requires that you boot the system using UEK R3.

The send operation compares two subvolumes and writes a description of how to convert one subvolume (the *parent* subvolume) into the other (the *sent* subvolume). You would usually direct the output to a file for later use or pipe it to a receive operation for immediate use.

The simplest form of the send operation writes a complete description of a subvolume:

```
sudo btrfs send [-v] [-f] [sent_file] ... subvol
```

You can specify multiple instances of the `-v` option to display increasing amounts of debugging output. The `-f` option allows you to save the output to a file. Both of these options are implicit in the following usage examples.

The following form of the send operation writes a complete description of how to convert one subvolume into another:

```
sudo btrfs send -p parent_subvol sent_subvol
```

If a subvolume such as a snapshot of the parent volume, known as a *clone source*, will be available during the receive operation from which some of the data can be recovered, you can specify the clone source to reduce the size of the output file:

```
sudo btrfs send [-p parent_subvol] -c clone_src [-c clone_src] ... subvol
```

You can specify the `-c` option multiple times if there is more than one clone source. If you do not specify the parent subvolume, btrfs chooses a suitable parent from the clone sources.

You use the receive operation to regenerate the sent subvolume at a specified path:

```
sudo btrfs receive [-f sent_file] mountpoint
```

How to Use Send/Receive to Implement Incremental Backups

The following procedure is a suggestion for setting up an incremental backup and restore process for a subvolume.

1. Create a read-only snapshot of the subvolume to serve as an initial reference point for the backup:

```
sudo btrfs subvolume snapshot -r /vol /vol/backup_0
```

2. Run `sync` to ensure that the snapshot has been written to disk:

```
sudo sync
```

3. Create a subvolume or directory on a `btrfs` file system as a backup area to receive the snapshot, for example, `/backupvol`.
4. Send the snapshot to `/backupvol`:

```
sudo btrfs send /vol/backup_0 | btrfs receive /backupvol
```

This command creates the subvolume `/backupvol/backup_0`.

Having created the reference backup, you can then create incremental backups as required.

5. To create an incremental backup:
 - a. Create a new snapshot of the subvolume:

```
sudo btrfs subvolume snapshot -r /vol /vol/backup_1
```

- b. Run `sync` to ensure that the snapshot has been written to disk:

```
sudo sync
```

- c. Send only the differences between the reference backup and the new backup to the backup area:

```
sudo btrfs send -p /vol/backup_0 /vol/backup_1 | btrfs receive /backupvol
```

This command creates the subvolume `/backupvol/backup_1`.

Using Quota Groups



Note:

The quota groups feature requires that you boot the system using UEK R3.

To enable quotas, use the following command on a newly created `btrfs` file system before any creating any subvolumes:

```
sudo btrfs quota enable volume
```

To assign a quota-group limit to a subvolume, use the following command:

```
sudo btrfs qgroup limit size /volume/subvolume
```

For example:

```
sudo btrfs qgroup limit 1g /myvol/subvol1  
sudo btrfs qgroup limit 512m /myvol/subvol2
```

To find out the quota usage for a subvolume, use the `btrfs qgroup show path` command:

Replacing Devices on a Live File System



Note:

The device replacement feature requires that you boot the system using UEK R3.

You can replace devices on a live file system. You do not need to unmount the file system or stop any tasks that are using it. If the system crashes or loses power while the replacement is taking place, the operation resumes when the system next mounts the file system.

Use the following command to replace a device on a mounted btrfs file system:

```
sudo btrfs replace start source_dev target_dev [-r] mountpoint
```

source_dev and *target_dev* specify the device to be replaced (*source device*) and the replacement device (*target device*). *mountpoint* specifies the file system that is using the source device. The target device must be the same size as or larger than the source device. If the source device is no longer available or you specify the `-r` option, the data is reconstructed by using redundant data obtained from other devices (such as another available mirror). The source device is removed from the file system when the operation is complete.

You can use the `btrfs replace status mountpoint` and `btrfs replace cancel mountpoint` commands to check the progress of the replacement operation or to cancel the operation.

Creating Snapshots of Files

You can use the `--reflink` option to the `cp` command to create lightweight copies of a file within the same subvolume of a btrfs file system. The copy-on-write mechanism saves disk space and allows copy operations to be almost instantaneous. The btrfs file system creates a new inode that shares the same disk blocks as the existing file, rather than creating a complete copy of the file's data or creating a link that points to the file's inode. The resulting file appears to be a copy of the original file, but the original data blocks are not duplicated. If you subsequently write to one of the files, the btrfs file system makes copies of the blocks before they are written to, preserving the other file's content.

For example, the following command creates the snapshot `bar` of the file `foo`:

```
sudo cp --reflink foo bar
```

About the Btrfs root File System

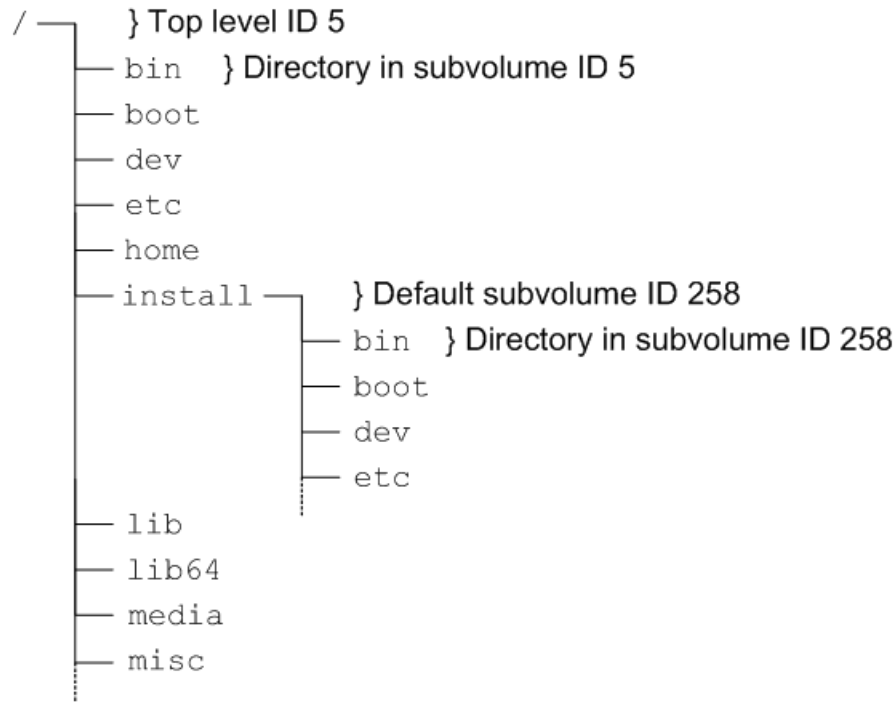
Oracle Linux 7 installation allows you to create a btrfs root file system. The mounted root file system is a snapshot (named `install`) of the root file system taken at the end of installation. To find out the ID of the parent of the root file system subvolume, use the following command:

```
sudo btrfs subvolume list /
```

```
ID 258 top level 5 path install
```

In this example, the installation root file system subvolume has an ID of 5. The subvolume with ID 258 (`install`) is currently mounted as `/`. [Figure 3-1](#) illustrates the layout of the file system:

Figure 3-1 Layout of the root File System Following Installation



The top-level subvolume with ID 5 records the contents of the root file system file system at the end of installation. The default subvolume (`install`) with ID 258 is currently mounted as the active root file system.

The `mount` command shows the device that is currently mounted as the root file system:

```

sudo mount

/dev/mapper/vg_btrfs-lv_root on / type btrfs (rw)
...

```

To mount the installation root file system volume, you can use the following commands:

```

sudo mkdir /instroot
sudo mount -o subvolid=5 /dev/mapper/vg-btrfs-lv-root /instroot

```

If you list the contents of `/instroot`, you can see both the contents of the installation root file system volume and the `install` snapshot, for example:

```

ls /instroot

bin  cgroup  etc  install  lib64  misc  net  proc  sbin  srv  tmp  var
boot dev    home lib      media  mnt  opt  root  selinux  sys  usr

```

The contents of `/` and `/instroot/install` are identical as demonstrated in the following example where a file (`foo`) created in `/instroot/install` is also visible in `/`:

```
touch /instroot/install/foo
ls /

bin  cgroup  etc  home  lib  media  mnt  opt  root  selinux  sys  usr
boot dev    foo  instroot  lib64  misc  net  proc  sbin  srv    tmp  var

ls /instroot/install

bin  cgroup  etc  home  lib  media  mnt  opt  root  selinux  sys  usr
boot dev    foo  instroot  lib64  misc  net  proc  sbin  srv    tmp  var

rm -f /foo
ls /

bin  cgroup  etc  instroot  lib64  misc  net  proc  sbin  srv  tmp  var
boot dev    home  lib    media  mnt  opt  root  selinux  sys  usr

# ls /instroot/install

bin  cgroup  etc  instroot  lib64  misc  net  proc  sbin  srv  tmp  var
boot dev    home  lib    media  mnt  opt  root  selinux  sys  usr
```

Creating Snapshots of the root File System

To take a snapshot of the current root file system:

1. Mount the top level of the root file system on a suitable mount point.

```
sudo mount -o subvolid=5 /dev/mapper/vg-btrfs-lv-root /mnt
```

2. Change directory to the mount point and take the snapshot. In this example, the `install` subvolume is currently mounted as the root file system system.

```
cd /mnt
sudo btrfs subvolume snapshot install root_snapshot_1

Create a snapshot of 'install' in './root_snapshot_1'
```

3. Change directory to `/` and unmount the top level of the file system.

```
# cd /
# umount /mnt
```

The list of subvolumes now includes the newly created snapshot.

```
sudo btrfs subvolume list /

ID 258 top level 5 path install
ID 260 top level 5 path root_snapshot_1
```

Mounting Alternate Snapshots as the root File System

If you want to roll back changes to your system, you can mount a snapshot as the root file system by specifying its ID as the default subvolume, for example:

```
sudo btrfs subvolume set-default 260 /
```

Reboot the system for the change to take effect.

Deleting Snapshots of the root File System

To delete a snapshot:

1. Mount the top level of the file system, for example:

```
sudo mount -o subvolid=5 /dev/mapper/vg-btrfs-lv-root /mnt
```

2. Change directory to the mount point and delete the snapshot.

```
cd /mnt  
sudo btrfs subvolume delete install
```

```
Delete subvolume '/mnt/install'
```

3. Change directory to / and unmount the top level of the file system.

```
cd /  
sudo umount /mnt
```

The list of subvolumes now does not include `install`.

```
sudo btrfs subvolume list /  
  
=ID 260 top level 5 path root_snapshot_1
```

4

Managing the Ext File System

This chapter describes tasks for administering the Ext file system in Oracle Linux.

Converting an Ext2, Ext3, or Ext4 File System to a Btrfs File System

You can use the `btrfs-convert` utility to convert an `ext2`, `ext3`, or `ext4` file system to `btrfs`. The utility preserves an image of the original file system in a snapshot named `ext2_saved`. This snapshot allows you to roll back the conversion, even if you have made changes to the `btrfs` file system.



Note:

You cannot convert the root file system or a bootable partition, such as `/boot`, to `btrfs`.

How to Convert a Non-root File System



Caution:

Before performing a file system conversion, make a backup of the file system from which you can restore its state.

To convert an `ext2`, `ext3`, or `ext4` file system other than the root file system to `btrfs`:

1. Unmount the file system.

```
sudo umount mountpoint
```

2. Run the correct version of `fsck` (for example, `fsck.ext4`) on the underlying device to check and correct the integrity of file system.

```
sudo fsck.extN -f device
```

3. Convert the file system to a `btrfs` file system.

```
sudo btrfs-convert device
```

4. Edit the file `/etc/fstab`, and change the file system type of the file system to `btrfs`, for example:

```
/dev/sdb          /myfs          btrfs    defaults 0 0
```

5. Mount the converted file system on the old mount point.

```
sudo mount device mountpoint
```

Converting a Non-root Ext2 File System to Ext3

▲ Caution:

Before performing a file system conversion, make a backup of the file system from which you can restore its state.

To convert a non-root ext2 file system to ext3:

1. Unmount the ext2 file system:

```
sudo umount filesystem
```

2. Use `fsck.ext2` to check the file system.

```
bash-4.1# fsck.ext2 -f device
```

3. Use the following command with the block device corresponding to the ext2 file system:

```
sudo tune2fs -j device
```

The command adds an ext3 journal inode to the file system.

4. Use `fsck.ext3` to check the file system.

```
bash-4.1# fsck.ext3 -f device
```

5. Correct any entry for the file system in `/etc/fstab` so that its type is defined as ext3 instead of ext2.

6. You can now remount the file system whenever convenient:

```
sudo mount filesystem
```

For more information, see the `tune2fs(8)` manual page.

Converting a root Ext2 File System to Ext3

▲ Caution:

Before performing a root file system conversion, make a full system backup from which you can restore its state.

To convert a root ext2 file system to ext3:

1. Use the following command with the block device corresponding to the root file system:

```
sudo tune2fs -j device
```

The command adds an ext3 journal to the file system as the file `/.journal`.

2. Run the `mount` command to determine the device that is currently mounted as the root file system.

In the following example, the root file system corresponds to the disk partition `/dev/sda2`:

```
sudo mount  
  
/dev/sda2 on / type ext2 (rw)
```

3. Shut down the system.
4. Boot the system from an Oracle Linux boot CD, DVD or ISO. You can download the ISO from <https://edelivery.oracle.com/linux>.
5. From the installation menu, select **Rescue Installed System**. When prompted, choose a language and keyboard, select **Local CD/DVD** as the installation media, select **No** to bypass starting the network interface, and select **Skip** to bypass selecting a rescue environment.
6. Select **Start shell** to obtain a `bash` shell prompt (`bash-4.1#`) at the bottom of the screen.
7. If the existing root file system is configured as an LVM volume, use the following command to start the volume group (for example, `vg_host01`):

```
bash-4.1# lvchange -ay vg_host01
```

8. Use `fsck.ext3` to check the file system.

```
bash-4.1# fsck.ext3 -f device
```

In the previous example, `device` is the root file system device (for example, `/dev/sda2`).

The command moves the `.journal` file to the journal inode.

9. Create a mount point (`/mnt1`) and mount the converted root file system on that mount point, for example:

```
bash-4.1# mkdir /mnt1  
bash-4.1# mount -t ext3 device /mnt1
```

10. Using a text editor, edit the `/mnt1/etc/fstab` file and change the file system type of the root file system to `ext3`, for example:

```
/dev/sda2      /      ext3      defaults 1 1
```

11. Create the file `.autorelabel` in the root of the mounted file system.

```
bash-4.1# touch /mnt1/.autorelabel
```

The presence of the `.autorelabel` file in `/` instructs SELinux to recreate the security attributes of all files on the file system.

 **Note:**

If you do not create the `.autorelabel` file, you might not be able to boot the system successfully. If you forget to create the file and the reboot fails, either disable SELinux temporarily by specifying `selinux=0` to the kernel boot parameters, or run SELinux in permissive mode by specifying `enforcing=0`.

12. Unmount the converted root file system.

```
bash-4.1# umount /mnt1
```

13. Remove the boot CD, DVD, or ISO, and reboot the system.
For more information, see the `tune2fs(8)` manual page.

5

Managing the XFS File System

This chapter describes tasks for administering the XFS file system in Oracle Linux.

About the XFS File System

XFS is a high-performance journaling file system that was initially created by Silicon Graphics, Inc. for the IRIX operating system and later ported to Linux. The parallel I/O performance of XFS provides high scalability for I/O threads, file system bandwidth, file and file system size, even when the file system spans many storage devices.

A typical use case for XFS is to implement a several-hundred terabyte file system across multiple storage servers, each server consisting of multiple FC-connected disk arrays.

XFS is supported for use with the root (`/`) or `boot` file systems on Oracle Linux 7.

XFS has a large number of features that make it suitable for deployment in an enterprise-level computing environment that requires the implementation of very large file systems:

- XFS implements journaling for metadata operations, which guarantees the consistency of the file system following loss of power or a system crash. XFS records file system updates asynchronously to a circular buffer (the *journal*) before it can commit the actual data updates to disk. The journal can be located either internally in the data section of the file system, or externally on a separate device to reduce contention for disk access. If the system crashes or loses power, it reads the journal when the file system is remounted, and replays any pending metadata operations to ensure the consistency of the file system. The speed of this recovery does not depend on the size of the file system.
- XFS is internally partitioned into allocation groups, which are virtual storage regions of fixed size. Any files and directories that you create can span multiple allocation groups. Each allocation group manages its own set of inodes and free space independently of other allocation groups to provide both scalability and parallelism of I/O operations. If the file system spans many physical devices, allocation groups can optimize throughput by taking advantage of the underlying separation of channels to the storage components.
- XFS is an extent-based file system. To reduce file fragmentation and file scattering, each file's blocks can have variable length extents, where each extent consists of one or more contiguous blocks. XFS's space allocation scheme is designed to efficiently locate free extents that it can use for file system operations. XFS does not allocate storage to the holes in sparse files. If possible, the extent allocation map for a file is stored in its inode. Large allocation maps are stored in a data structure maintained by the allocation group.
- To maximize throughput for XFS file systems that you create on an underlying striped, software or hardware-based array, you can use the `su` and `sw` arguments to the `-d` option of the `mkfs.xfs` command to specify the size of each stripe unit and the number of units per stripe. XFS uses the information to align data, inodes, and journal appropriately for the storage. On `lvm` and `md` volumes and some hardware RAID configurations, XFS can automatically select the optimal stripe parameters for you.
- To reduce fragmentation and increase performance, XFS implements *delayed allocation*, reserving file system blocks for data in the buffer cache, and allocating the block when the operating system flushes that data to disk.

- XFS supports extended attributes for files, where the size of each attribute's value can be up to 64 KB, and each attribute can be allocated to either a `root` or a `user` name space.
- Direct I/O in XFS implements high throughput, non-cached I/O by performing DMA directly between an application and a storage device, utilising the full I/O bandwidth of the device.
- To support the snapshot facilities that volume managers, hardware subsystems, and databases provide, you can use the `xfs_freeze` command to suspend and resume I/O for an XFS file system. See [Freezing and Unfreezing an XFS File System](#).
- To defragment individual files in an active XFS file system, you can use the `xfs-fsr` command. See [Defragmenting an XFS File System](#).
- To grow an XFS file system, you can use the `xfs_growfs` command. See [Growing an XFS File System](#).
- To back up and restore a live XFS file system, you can use the `xfsdump` and `xfsrestore` commands. See [Backing up and Restoring XFS File Systems](#).
- XFS supports user, group, and project disk quotas on block and inode usage that are initialized when the file system is mounted. Project disk quotas allow you to set limits for individual directory hierarchies within an XFS file system without regard to which user or group has write access to that directory hierarchy.

You can find more information about XFS at <https://xfs.wiki.kernel.org/>.

About External XFS Journals

The default location for an XFS journal is on the same block device as the data. As synchronous metadata writes to the journal must complete successfully before any associated data writes can start, such a layout can lead to disk contention for the typical workload pattern on a database server. To overcome this problem, you can place the journal on a separate physical device with a low-latency I/O path. As the journal typically requires very little storage space, such an arrangement can significantly improve the file system's I/O throughput. A suitable host device for the journal is a solid-state drive (SSD) device or a RAID device with a battery-backed write-back cache.

To reserve an external journal with a specified size when you create an XFS file system, specify the `-l logdev=device,size=size` option to the `mkfs.xfs` command. If you omit the `size` parameter, `mkfs.xfs` selects a journal size based on the size of the file system. To mount the XFS file system so that it uses the external journal, specify the `-o logdev=device` option to the `mount` command.

About XFS Write Barriers

A write barrier assures file system consistency on storage hardware that supports flushing of in-memory data to the underlying device. This ability is particularly important for write operations to an XFS journal that is held on a device with a volatile write-back cache.

By default, an XFS file system is mounted with a write barrier. If you create an XFS file system on a LUN that has a battery-backed, non-volatile cache, using a write barrier degrades I/O performance by requiring data to be flushed more often than necessary.

In such cases, you can remove the write barrier by mounting the file system with the `-o nobarrier` option to the `mount` command.

About Lazy Counters

With lazy-counters enabled on an XFS file system, the free-space and inode counters are maintained in parts of the file system other than the superblock. This arrangement can significantly improve I/O performance for application workloads that are metadata intensive.

Lazy counters are enabled by default, but if required, you can disable them by specifying the `-l lazy-count=0` option to the `mkfs.xfs` command.

Installing the XFS Packages



Note:

You can also obtain the XFS packages from the Oracle Linux Yum Server.

To install the XFS packages on a system:

1. Log in to ULN, and subscribe your system to the `ol7_x86_64_latest` channel.
2. On your system, use `yum` to install the `xfsprogs` and `xfsdump` packages:

```
sudo yum install xfsprogs xfsdump
```

3. If you require the XFS development and QA packages, additionally subscribe your system to the `ol7_x86_64_optional` channel and use `yum` to install them:

```
sudo yum install xfsprogs-devel xfsprogs-qa-devel
```

Creating an XFS File System

You can use the `mkfs.xfs` command to create an XFS file system, for example.

```
sudo mkfs.xfs /dev/vg0/lv0
```

```
meta-data=/dev/vg0/lv0      isize=256    agcount=32, agsize=8473312 blks
      =                   sectsz=512    attr=2, projid32bit=0
data      =                   bsize=4096  blocks=271145984, imaxpct=25
      =                   sunit=0        swidth=0 blks
naming    =version 2        bsize=4096  ascii-ci=0
log       =internal log    bsize=4096  blocks=32768, version=2
      =                   sectsz=512    sunit=0 blks, lazy-count=1
realtime  =none            extsz=4096  blocks=0, rtextents=0
```

To create an XFS file system with a stripe-unit size of 32 KB and 6 units per stripe, you would specify the `su` and `sw` arguments to the `-d` option, for example:

```
sudo mkfs.xfs -d su=32k,sw=6 /dev/vg0/lv1
```

For more information, see the `mkfs.xfs(8)` manual page.

Modifying an XFS File System

**Note:**

You cannot modify a mounted XFS file system.

You can use the `xfs_admin` command to modify an unmounted XFS file system. For example, you can enable or disable lazy counters, change the file system UUID, or change the file system label.

To display the existing label for an unmounted XFS file system and then apply a new label:

```
sudo xfs_admin -l /dev/sdb

label = ""

sudo xfs_admin -L "VideoRecords" /dev/sdb

writing all SBs
new label = "VideoRecords"
```

**Note:**

The label can be a maximum of 12 characters in length.

To display the existing UUID and then generate a new UUID:

```
sudo xfs_admin -u /dev/sdb

UUID = cd4f1cc4-15d8-45f7-afa4-2ae87d1db2ed

sudo xfs_admin -U generate /dev/sdb

writing all SBs
new UUID = c1b9d5a2-f162-11cf-9ece-0020afc76f16
```

To clear the UUID altogether:

```
sudo xfs_admin -U nil /dev/sdb

Clearing log and setting UUID
writing all SBs
new UUID = 00000000-0000-0000-0000-000000000000
```

To disable and then re-enable lazy counters:

```
sudo xfs_admin -c 0 /dev/sdb

Disabling lazy-counters

sudo xfs_admin -c 1 /dev/sdb
```

Enabling lazy-counters

For more information, see the `mkfs_admin(8)` manual page.

Growing an XFS File System



Note:

You cannot grow an XFS file system that is currently unmounted.

There is currently no command to shrink an XFS file system.

You can use the `xfs_growfs` command to increase the size of a mounted XFS file system if there is space on the underlying devices to accommodate the change. The command does not have any effect on the layout or size of the underlying devices. If necessary, use the underlying volume manager to increase the physical storage that is available. For example, you can use the `vgextend` command to increase the storage that is available to an LVM volume group and `lvextend` to increase the size of the logical volume that contains the file system.

You cannot use the `parted` command to resize a partition that contains an XFS file system. You must instead recreate the partition with a larger size and restore its contents from a backup if you deleted the original partition or from the contents of the original partition if you did not delete it to free up disk space.

For example, to increase the size of `/myxfs1` to 4 TB, assuming a block size of 4 KB:

```
sudo xfs_growfs -D 1073741824 /myxfs1
```

To increase the size of the file system to the maximum size that the underlying device supports, specify the `-d` option:

```
sudo xfs_growfs -d /myxfs1
```

For more information, see the `xfs_growfs(8)` manual page.

Freezing and Unfreezing an XFS File System

If you need to take a hardware-based snapshot of an XFS file system, you can temporarily stop write operations to it.



Note:

You do not need to explicitly suspend write operations if you use the `lvcreate` command to take an LVM snapshot.

To freeze and unfreeze an XFS file system, use the `-f` and `-u` options with the `xfs_freeze` command, for example:

```
sudo xfs_freeze -f /myxfs
```

You would then take a snapshot of file systems, after which you would type:

```
sudo xfs_freeze -u /myxfs
```



Note:

You can also use the `xfs_freeze` command with `btrfs`, `ext3`, and `ext4` file systems.

For more information, see the `xfs_freeze(8)` manual page.

Setting Quotas on an XFS File System

The following list shows the `mount` options that you can specify to enable quotas on an XFS file system:

- `gqnoenforce`
Enable group quotas. Report usage, but do not enforce usage limits.
- `gquota`
Enable group quotas and enforce usage limits.
- `pqnoenforce`
Enable project quotas. Report usage, but do not enforce usage limits.
- `pquota`
Enable project quotas and enforce usage limits.
- `uqnoenforce`
Enable user quotas. Report usage, but do not enforce usage limits.
- `uquota`
Enable user quotas and enforce usage limits.

To show the block usage limits and the current usage in the `myxfs` file system for all users, use the `xfs_quota` command:

```
sudo xfs_quota -x -c 'report -h' /myxfs
```

```
User quota on /myxfs (/dev/vg0/lv0)
      Blocks
User ID      Used   Soft   Hard Warn/Grace
-----
root         0      0      0  00 [-----]
guest        0    200M   250M  00 [-----]
```

The following forms of the command display the free and used counts for blocks and inodes respectively in the manner of the `df -h` command:

```
sudo xfs_quota -c 'df -h' /myxfs
```

```
Filesystem      Size   Used  Avail Use% Pathname
/dev/vg0/lv0 200.0G  32.2M  20.0G   1% /myxfs
```

```
sudo xfs_quota -c 'df -ih' /myxfs
```

```
Filesystem  Inodes   Used   Free Use% Pathname
/dev/vg0/lv0 21.0m     4 21.0m  1% /myxfs
```

If you specify the `-x` option to enter expert mode, you can use subcommands such as `limit` to set soft and hard limits for block and inode usage by an individual user, for example:

```
sudo xfs_quota -x -c 'limit bsoft=200m bhard=250m isoft=200 ihard=250 guest' /myxfs
```

Of course, this command requires that you mounted the file system with user quotas enabled.

To set limits for a group on an XFS file system that you have mounted with group quotas enabled, specify the `-g` option to `limit`, for example:

```
sudo xfs_quota -x -c 'limit -g bsoft=5g bhard=6g devgrp' /myxfs
```

For more information, see the `xfs_quota(8)` manual page.

Setting Project Quotas

User and group quotas are supported by other file systems, such as `ext4`. The XFS file system additionally allows you to set quotas on individual directory hierarchies in the file system that are known as *managed trees*. Each managed tree is uniquely identified by a *project ID* and an optional *project name*. Being able to control the disk usage of a directory hierarchy is useful if you do not otherwise want to set quota limits for a privileged user (for example, `/var/log`) or if many users or groups have write access to a directory (for example, `/var/tmp`).

To define a project and set quota limits on it:

1. Mount the XFS file system with project quotas enabled:

```
sudo mount -o pquota device mountpoint
```

For example, to enable project quotas for the `/myxfs` file system:

```
sudo mount -o pquota /dev/vg0/lv0 /myxfs
```

2. Define a unique project ID for the directory hierarchy in the `/etc/projects` file:

```
echo project_ID:mountpoint/directory >> /etc/projects
```

For example, to set a project ID of 51 for the directory hierarchy `/myxfs/testdir`:

```
echo 51:/myxfs/testdir >> /etc/projects
```

3. Create an entry in the `/etc/projid` file that maps a project name to the project ID:

```
echo project_name:project_ID >> /etc/projid
```

For example, to map the project name `testproj` to the project with ID 51:

```
echo testproj:51 >> /etc/projid
```

4. Use the `project` subcommand of `xfs_quota` to define a managed tree in the XFS file system for the project:

```
sudo xfs_quota -x -c 'project -s project_name' mountpoint
```

For example, to define a managed tree in the `/myxfs` file system for the project `testproj`, which corresponds to the directory hierarchy `/myxfs/testdir`:

```
sudo xfs_quota -x -c 'project -s testproj' /myxfs
```

5. Use the `limit` subcommand to set limits on the disk usage of the project:

```
sudo xfs_quota -x -c 'limit -p arguments project_name' mountpoint
```

For example, to set a hard limit of 10 GB of disk space for the project `testproj`:

```
sudo xfs_quota -x -c 'limit -p bhard=10g testproj' /myxfs
```

For more information, see the `projects(5)`, `projid(5)`, and `xfs_quota(8)` manual pages.

Backing up and Restoring XFS File Systems

The `xfsdump` package contains the `xfsdump` and `xfsrestore` utilities. `xfsdump` examines the files in an XFS file system, determines which files need to be backed up, and copies them to the storage medium. Any backups that you create using `xfsdump` are portable between systems with different endian architectures. `xfsrestore` restores a full or incremental backup of an XFS file system. You can also restore individual files and directory hierarchies from backups.



Note:

Unlike an LVM snapshot, which immediately creates a sparse clone of a volume, `xfsdump` takes time to make a copy of the file system data.

You can use the `xfsdump` command to create a backup of an XFS file system on a device such as a tape drive, or in a backup file on a different file system. A backup can span multiple physical media that are written on the same device, and you can write multiple backups to the same medium. You can write only a single backup to a file. The command does not overwrite existing XFS backups that it finds on physical media. You must use the appropriate command to erase a physical medium if you need to overwrite any existing backups.

For example, the following command writes a level 0 (base) backup of the XFS file system, `/myxfs` to the device `/dev/st0` and assigns a session label to the backup:

```
sudo xfsdump -l 0 -L "Backup level 0 of /myxfs `date`" -f /dev/st0 /myxfs
```

You can make incremental dumps relative to an existing backup by using the command:

```
sudo xfsdump -l level -L "Backup level level of /myxfs `date`" -f /dev/st0 /myxfs
```

A level 1 backup records only file system changes since the level 0 backup, a level 2 backup records only the changes since the latest level 1 backup, and so on up to level 9.

If you interrupt a backup by typing `Ctrl-C` and you did not specify the `-J` option (suppress the dump inventory) to `xfsdump`, you can resume the dump at a later date by specifying the `-R` option:

```
sudo xfsdump -R -l 1 -L "Backup level 1 of /myxfs `date`" -f /dev/st0 /myxfs
```


In this example, the backup session label from the earlier, interrupted session is overridden.

You use the `xfsrestore` command to find out information about the backups you have made of an XFS file system or to restore data from a backup.

The `xfsrestore -I` command displays information about the available backups, including the session ID and session label. If you want to restore a specific backup session from a backup medium, you can specify either the session ID or the session label.

For example, to restore an XFS file system from a level 0 backup by specifying the session ID:

```
sudo xfsrestore -f /dev/st0 -S c76b3156-c37c-5b6e-7564-a0963ff8ca8f /myxfs
```

If you specify the `-r` option, you can cumulatively recover all data from a level 0 backup and the higher-level backups that are based on that backup:

```
sudo xfsrestore -r -f /dev/st0 -v silent /myxfs
```

The command searches the archive looking for backups based on the level 0 backup, and prompts you to choose whether you want to restore each backup in turn. After restoring the backup that you select, the command exits. You must run this command multiple times, first selecting to restore the level 0 backup, and then subsequent higher-level backups up to and including the most recent one that you require to restore the file system data.

**Note:**

After completing a cumulative restoration of an XFS file system, you should delete the housekeeping directory that `xfsrestore` creates in the destination directory.

You can recover a selected file or subdirectory contents from the backup medium, as shown in the following example, which recovers the contents of `/myxfs/profile/examples` to `/tmp/profile/examples` from the backup with a specified session label:

```
sudo xfsrestore -f /dev/sr0 -L "Backup level 0 of /myxfs Sat Mar 2 14:47:59 GMT 2013" -s profile/examples /usr/tmp
```

Alternatively, you can interactively browse a backup by specifying the `-i` option:

```
sudo xfsrestore -f /dev/sr0 -i
```

This form of the command allows you browse a backup as though it were a file system. You can change directories, list files, add files, delete files, or extract files from a backup.

To copy the entire contents of one XFS file system to another, you can combine `xfsdump` and `xfsrestore`, using the `-J` option to suppress the usual dump inventory housekeeping that the commands perform:

```
sudo xfsdump -J - /myxfs | xfsrestore -J - /myxfsclone
```

For more information, see the `xfsdump(8)` and `xfsrestore(8)` manual pages.

Defragmenting an XFS File System

You can use the `xfs_fsr` command to defragment whole XFS file systems or individual files within an XFS file system. As XFS is an extent-based file system, it is usually unnecessary to defragment a whole file system, and doing so is not recommended.

To defragment an individual file, specify the name of the file as the argument to `xfs_fsr`.

```
sudo xfs_fsr pathname
```

If you run the `xfs_fsr` command without any options, the command defragments all currently mounted, writeable XFS file systems that are listed in `/etc/mtab`. For a period of two hours, the command passes over each file system in turn, attempting to defragment the top ten percent of files that have the greatest number of extents. After two hours, the command records its progress in the file `/var/tmp/.fsrlast_xfs`, and it resumes from that point if you run the command again.

For more information, see the `xfs_fsr(8)` manual page.

Checking and Repairing an XFS File System



Note:

If you have an Oracle Linux Premier Support account and encounter a problem mounting an XFS file system, send a copy of the `/var/log/messages` file to Oracle Support and wait for advice.

If you cannot mount an XFS file system, you can use the `xfs_repair -n` command to check its consistency. Usually, you would only run this command on the device file of an unmounted file system that you believe has a problem. The `xfs_repair -n` command displays output to indicate changes that would be made to the file system in the case where it would need to complete a repair operation, but will not modify the file system directly.

If you can mount the file system and you do not have a suitable backup, you can use `xfsdump` to attempt to back up the existing file system data. However, the command might fail if the file system's metadata has become too corrupted.

You can use the `xfs_repair` command to attempt to repair an XFS file system specified by its device file. The command replays the journal log to fix any inconsistencies that might have resulted from the file system not being cleanly unmounted. Unless the file system has an inconsistency, it is usually not necessary to use the command, as the journal is replayed every time that you mount an XFS file system.

```
sudo xfs_repair device
```

If the journal log has become corrupted, you can reset the log by specifying the `-L` option to `xfs_repair`.

NOT_SUPPORTED:

Resetting the log can leave the file system in an inconsistent state, resulting in data loss and data corruption. Unless you are experienced in debugging and repairing XFS file systems using `xfs_db`, it is recommended that you instead recreate the file system and restore its contents from a backup.

If you cannot mount the file system or you do not have a suitable backup, running `xfs_repair` is the only viable option unless you are experienced in using `xfs_db`.

`xfs_db` provides an internal command set that allows you to debug and repair an XFS file system manually. The commands allow you to perform scans on the file system, and to navigate and display its data structures. If you specify the `-x` option to enable expert mode, you can modify the data structures.

```
sudo xfs_db [-x] device
```

For more information, see the `xfs_db(8)` and `xfs_repair(8)` manual pages, and the `help` command within `xfs_db`.

6

Shared File System Administration

This chapter describes administration tasks for the NFS and Samba shared file systems.

About Shared File Systems

Oracle Linux supports the following shared file system types:

NFS

The Network File System (NFS) is a distributed file system that allows a client computer to access files over a network as though the files were on local storage. See [About NFS](#).

Samba

Samba enables the provision of file and print services for Microsoft Windows clients and can integrate with a Windows workgroup, NT4 domain, or Active Directory domain. See [About Samba](#).

About NFS

A Network File System (NFS) server can share directory hierarchies in its local file systems with remote client systems over an IP-based network. After an NFS server exports a directory, NFS clients mount this directory if they have been granted permission to do so. The directory appears to the client systems as if it were a local directory. NFS centralizes storage provisioning and can improve data consistency and reliability.

Oracle Linux 7 supports the following versions of the NFS protocol:

- NFS version 3 (NFSv3), specified in [RFC 1813](#).
- NFS version 4 (NFSv4), specified in [RFC 7530](#).
- NFS version 4 minor version 1 (NFSv4.1), specified in [RFC 5661](#).

NFSv3 relies on Remote Procedure Call (RPC) services, which are controlled by the `rpcbind` service. `rpcbind` responds to requests for an RPC service and sets up connections for the requested service. In addition, separate services are used to handle locking and mounting protocols. Configuring a firewall to cope with the various ranges of ports that are used by all these services can be complex and error prone.

NFSv4 does not use `rpcbind` as the NFS server itself listens on TCP port 2049 for service requests. The mounting and locking protocols are also integrated into the NFSv4 protocol, so separate services are also not required for these protocols. These refinements mean that firewall configuration for NFSv4 is no more difficult than for a service such as HTTP.

Configuring an NFS Server

To configure an NFS server:

1. Install the `nfs-utils` package:

```
sudo yum install nfs-utils
```

2. Edit the `/etc/exports` file to define the directories that the server will make available for clients to mount, for example:

```
/var/folder 192.0.2.102(rw,async)
/usr/local/apps *(all_squash,anonuid=501,anongid=501,ro)
/var/projects/proj1 192.168.1.0/24(ro) mgmtpc(rw)
```

Each entry consists of the local path to the exported directory, followed by a list of clients that can mount the directory with client-specific mount options in parentheses. If this example:

- The client system with the IP address 192.0.2.102 can mount `/var/folder` with read and write permissions. All writes to the disk are asynchronous, which means that the server does not wait for write requests to be written to disk before responding to further requests from the client.
- All clients can mount `/usr/local/apps` read-only, and all connecting users including `root` are mapped to the local unprivileged user with UID 501 and GID 501.
- All clients on the 192.168.1.0 subnet can mount `/var/projects/proj1` read-only, and the client system named `mgmtpc` can mount the directory with read-write permissions.

 **Note:**

There is no space between a client specifier and the parenthesized list of options.

For more information, see the `exports(5)` manual page.

3. Start the `nfs-server` service, and configure the service to start following a system reboot:

```
sudo systemctl start nfs-server
sudo systemctl enable nfs-server
```

4. If the server will serve NFSv4 clients, edit `/etc/idmapd.conf` and edit the definition for the `Domain` parameter to specify the DNS domain name of the server, for example:

```
Domain = mydom.com
```

This setting prevents the owner and group being unexpectedly listed as the anonymous user or group (`nobody` or `nogroup`) on NFS clients when the `all_squash` mount option has not been specified.

5. If you need to allow access through the firewall for NFSv4 clients only, use the following commands:

```
sudo firewall-cmd --zone=zone --add-service=nfs
sudo firewall-cmd --permanent --zone=zone --add-service=nfs
```

This configuration assumes that `rpc.nfsd` listens for client requests on TCP port 2049.

6. If you need to allow access through the firewall for NFSv3 clients as well as NFSv4 clients:

- a. Edit `/etc/sysconfig/nfs` and create port settings for handling network mount requests and status monitoring:

```
# Port rpc.mountd should listen on.
MOUNTD_PORT=892
```

```
# Port rpc.statd should listen on.
STATD_PORT=662
```

The port values shown in this example are the default settings that are commented-out in the file.

- b. Edit `/etc/sysctl.conf` and configure settings for the TCP and UDP ports on which the network lock manager should listen:

```
fs.nfs.nlm_tcpport = 32803
fs.nfs.nlm_udpport = 32769
```

- c. To verify that none of the ports that you have specified in `/etc/sysconfig/nfs` or `/etc/sysctl.conf` is in use, enter the following commands:

```
sudo lsof -i tcp:32803
sudo lsof -i udp:32769
sudo lsof -i :892
sudo lsof -i :662
```

If any port is in use, use the `lsof -i` command to determine an unused port and amend the setting in `/etc/sysconfig/nfs` or `/etc/sysctl.conf` as appropriate.

- d. Shut down and reboot the server.

```
sudo systemctl reboot
```

NFS fails to start if one of the specified ports is in use, and reports an error in `/var/log/messages`. Edit `/etc/sysconfig/nfs` or `/etc/sysctl.conf` as appropriate to use a different port number for the service that could not start, and attempt to restart the `nfslock` and `nfs-server` services. You can use the `rpcinfo -p` command to confirm on which ports RPC services are listening.

- e. Restart the firewall service and configure the firewall to allow NFSv3 connections:

```
sudo systemctl restart firewalld
```

```
sudo firewall-cmd --zone=zone \
  --add-port=2049/tcp --add-port=2049/udp \
  --add-port=111/tcp --add-port=111/udp \
  --add-port=32803/tcp --add-port=32769/udp \
  --add-port=892/tcp --add-port=892/udp \
  --add-port=662/tcp --add-port=662/udp
```

```
sudo firewall-cmd --permanent --zone=zone \
  --add-port=2049/tcp --add-port=2049/udp \
  --add-port=111/tcp --add-port=111/udp \
  --add-port=32803/tcp --add-port=32769/udp \
  --add-port=892/tcp --add-port=892/udp \
  --add-port=662/tcp --add-port=662/udp
```

The port values shown in this example assume that the default port settings in `/etc/sysconfig/nfs` and `/etc/sysctl.conf` are available for use by RPC services. This configuration also assumes that `rpc.nfsd` and `rpcbind` listen on ports 2049 and 111 respectively.

7. Use the `showmount -e` command to display a list of the exported file systems, for example:

```
sudo showmount -e

Export list for host01.mydom.com
/var/folder 192.0.2.102
/usr/local/apps *
/var/projects/proj1 192.168.1.0/24 mgmtpc
```

`showmount -a` lists the current clients and the file systems that they have mounted, for example:

```
sudo showmount -a

mgmtpc.mydom.com:/var/projects/proj1
```

Note:

To be able to use the `showmount` command from NFSv4 clients, `MOUNTD_PORT` must be defined in `/etc/sysconfig/nfs` and a firewall rule must allow access on this TCP port.

If you want to export or unexport directories without editing `/etc/exports` and restarting the NFS service, use the `exportfs` command. The following example makes `/var/dev` available with read and write access by all clients, and ignores any existing entries in `/etc/exports`.

```
sudo exportfs -i -o ro */var/dev
```

For more information, see the `exportfs(8)`, `exports(5)`, and `showmount(8)` manual pages.

Mounting an NFS File System

To mount an NFS file system on a client:

1. Install the `nfs-utils` package:

```
sudo yum install nfs-utils
```

2. Use `showmount -e` to discover what file systems an NFS server exports, for example:

```
sudo showmount -e host01.mydom.com

Export list for host01.mydom.com
/var/folder 192.0.2.102
/usr/local/apps *
/var/projects/proj1 192.168.1.0/24 mgmtpc
```

3. Use the `mount` command to mount an exported NFS file system on an available mount point:

```
sudo mount -t nfs -o ro,nosuid host01.mydoc.com:/usr/local/apps /apps
```

This example mounts `/usr/local/apps` exported by `host01.mydoc.com` with read-only permissions on `/apps`. The `nosuid` option prevents remote users from gaining higher privileges by running a `setuid` program.

4. To configure the system to mount an NFS file system at boot time, add an entry for the file system to `/etc/fstab`, for example:

```
host01.mydoc.com:/usr/local/apps /apps nfs ro,nosuid 0 0
```

For more information, see the `mount(8)`, `nfs(5)`, and `showmount(8)` manual pages.

About Samba

Samba is an open-source implementation of the Server Message Block (SMB) protocol that allows Oracle Linux to interoperate with Windows systems as both a server and a client. Samba can share Oracle Linux files and printers with Windows systems, and it enables Oracle Linux users to access files on Windows systems. Samba uses the NetBIOS over TCP/IP protocol that allows computer applications that depend on the NetBIOS API to work on TCP/IP networks.

Configuring a Samba Server

To configure a Samba server:

1. Install the `samba` and `samba-winbind` packages:

```
sudo yum install samba samba-winbind
```

2. Edit `/etc/samba/smb.conf` and configure the sections to support the required services, for example:

```
[global]
security = ADS
realm = MYDOM.REALM
password server = krbsvr.mydom.com
load printers = yes
printing = cups
printcap name = cups

[printers]
comment = All Printers
path = /var/spool/samba
browseable = no
guest ok = yes
writable = no
printable = yes
printer admin = root, @ntadmins, @smbprintadm

[homes]
comment = User home directories
valid users = @smbusers
browseable = no
writable = yes
guest ok = no

[apps]
comment = Shared /usr/local/apps directory
path = /usr/local/apps
browseable = yes
```



```
writable = no
guest ok = yes
```

The `[global]` section contains settings for the Samba server. In this example, the server is assumed to be a member of an Active Directory (AD) domain that is running in native mode. Samba relies on tickets issued by the Kerberos server to authenticate clients who want to access local services.

For more information, see [About Samba Configuration for Windows Workgroups and Domains](#).

The `[printers]` section specifies support for print services. The `path` parameter specifies the location of a spooling directory that receives print jobs from Windows clients before submitting them to the local print spooler. Samba advertises all locally configured printers on the server.

The `[homes]` section provide a personal share for each user in the `smbusers` group. The settings for `browsable` and `writable` prevent other users from browsing home directories, while allowing full access to valid users.

The `[apps]` section specifies a share named `apps`, which grants Windows users browsing and read-only permission to the `/usr/local/apps` directory.

3. Configure the system firewall to allow incoming TCP connections to ports 139 and 445, and incoming UDP datagrams on ports 137 and 138:

```
sudo firewall-cmd --zone=zone --add-port=139/tcp --add-port=445/tcp --add-
port=137-138/udp
sudo firewall-cmd --permanent --zone=zone --add-port=139/tcp --add-
port=445/tcp --add-port=137-138/udp
```

Add similar rules for other networks from which Samba clients can connect.

The `nmbd` daemon services NetBIOS Name Service requests on UDP port 137 and NetBIOS Datagram Service requests on UDP port 138.

The `smbd` daemon services NetBIOS Session Service requests on TCP port 139 and Microsoft Directory Service requests on TCP port 445.

4. Start the `smb` service, and configure the service to start following a system reboot:

```
sudo systemctl start smb
sudo systemctl enable smb
```

If you change the `/etc/samba/smb.conf` file and any files that it references, the `smb` service will reload its configuration automatically after a delay of up to one minute. You can force `smb` to reload its configuration by sending a `SIGHUP` signal to the service daemon:

```
sudo killall -SIGHUP smbd
```

Making `smb` reload its configuration has no effect on established connections. You must restart the `smb` service or the existing users of the service must disconnect and then reconnect.

To restart the `smb` service, use the following command:

```
sudo systemctl restart smb
```

For more information, see the `smb.conf(5)` and `smbd(8)` manual pages and <https://www.samba.org/samba/docs/>.

About Samba Configuration for Windows Workgroups and Domains

Windows systems on an enterprise network usually belong either to a workgroup or to a domain.

Workgroups are usually only configured on networks that connect a small number of computers. A workgroup environment is a peer-to-peer network where systems do not rely on each other for services and there is no centralized management. User accounts, access control, and system resources are configured independently on each system. Such systems can share resources only if configured to do so.

A Samba server can act as a standalone server within a workgroup.

More typically, corporate networks configure domains to allow large numbers of networked systems to be administered centrally. A domain is a group of trusted computers that share security and access control. Systems known as domain controllers provides centralized management and security. Windows domains are usually configured to use Active Directory (AD), which uses the Lightweight Directory Access Protocol (LDAP) to implement versions of Kerberos and DNS providing authentication, access control to domain resources, and name service. Some Windows domains use Windows NT4 security, which does not use Kerberos to perform authentication.

A Samba server can be a member of an AD or NT4 security domain, but it cannot operate as a domain controller. As domain member Samba server must authenticate itself with a domain controller and so is controlled by the security rules of the domain. The domain controller authenticates clients, and the Samba server controls access to printers and network shares.

Configuring Samba as a Standalone Server

A standalone Samba server can be a member of a workgroup. The following `[global]` section from `/etc/samba/smb.conf` shows an example of how to configure a standalone server using share-level security:

```
[global]
security = share
workgroup = workgroup_name
netbios name = netbios_name
```

The client provides only a password and not a user name to the server. Typically, each share is associated with a `valid users` parameter and the server validates the password against the hashed passwords stored in `/etc/passwd`, `/etc/shadow`, NIS, or LDAP for the listed users. Using share-level security is discouraged in favor of user-level security, for example:

```
[global]
security = user
workgroup = workgroup_name
netbios name = netbios_name
```

In the user security model, a client must supply a valid user name and password. This model supports encrypted passwords. If the server successfully validates the client's user name and password, the client can mount multiple shares without being required to specify a password. Use the `smbpasswd` command to create an entry for a user in the Samba password file, for example:

```
# smbpasswd -a guest
New SMB password: password
```

```
Retype new SMB password: password
Added user guest.
```

The user must already exist as a user on the system. If a user is permitted to log into the server, he or she can use the `smbpasswd` command to change his or her password.

If a Windows user has a different user name from his or her user name on the Samba server, create a mapping between the names in the `/etc/samba/smbusers` file, for example:

```
root = admin administrator root
nobody = guest nobody pcguest smbguest
eddie = ejones
fiona = fchau
```

The first entry on each line is the user name on the Samba server. The entries after the equals sign (=) are the equivalent Windows user names.



Note:

Only the user security model uses Samba passwords.

The server security model, where the Samba server relies on another server to authenticate user names and passwords, is deprecated as it has numerous security and interoperability issues.

Configuring Samba as a Member of an ADS Domain

In the Activity Directory Server (ADS) security model, Samba acts as a domain member server in an ADS realm, and clients use Kerberos tickets for Active Directory authentication. You must configure Kerberos and join the server to the domain, which creates a machine account for your server on the domain controller.

To add a Samba server to an Active Directory domain:

1. Edit `/etc/samba/smb.conf` and configure the `[global]` section to use ADS:

```
[global]
security = ADS
realm = KERBEROS.REALM
```

It might also be necessary to specify the password server explicitly if different servers support AD services and Kerberos authentication:

```
password server = kerberos_server.your_domain
```

2. Install the `krb5-server` package:

```
# yum install krb5-server
```
3. Create a Kerberos ticket for the `Administrator` account in the Kerberos domain, for example:

```
# kinit Administrator@MYDOMAIN.COM
```

This command creates the Kerberos ticket that is required to join the server to the AD domain.

4. Join the server to the AD domain:

```
# net ads join -S winads.mydom.com -U Administrator%password
```

In this example, the AD server is `winads.mydom.com` and `password` is the password for the Administrator account.

The command creates a machine account in Active Directory for the Samba server and allows it to join the domain.

5. Restart the `smb` service:

```
# systemctl restart smb
```

Configuring Samba as a Member of a Windows NT4 Security Domain

Note:

If the Samba server acts as a Primary or Backup Domain Controller, do not use the domain security model. Configure the system as a standalone server that uses the user security model instead. See [Configuring Samba as a Standalone Server](#).

The domain security model is used with domains that implement Windows NT4 security. The Samba server must have a machine account in the domain (a domain security trust account). Samba authenticates user names and passwords with either a primary or a secondary domain controller.

To add a Samba server to an NT4 domain:

1. On the primary domain controller, use the Server Manager to add a machine account for the Samba server.
2. Edit `/etc/samba/smb.conf` and configure the `[global]` section to use ADS:

```
[global]
security = domain
workgroup = DOMAIN
netbios name = SERVERNAME
```

3. Join the server to the domain:

```
# net rpc join -S winpdc.mydom.com -U Administrator%password
```

In this example, the primary domain controller is `winpdc.mydom.com` and `password` is the password for the Administrator account.

4. Restart the `smb` service:

```
# systemctl restart smb
```

5. Create an account for each user who is allowed access to shares or printers:

```
# useradd -s /sbin/nologin username
```

```
# passwd username
```

In this example, the account's login shell is set to `/sbin/nologin` to prevent direct logins.

Accessing Samba Shares from a Windows Client

To access a share on a Samba server from Windows, open Computer or Windows Explorer, and enter the host name of the Samba server and the share name using the following format:

```
\\server_name\share_name
```

If you enter `\\server_name`, Windows displays the directories and printers that the server is sharing. You can also use the same syntax to map a network drive to a share name.

Accessing Samba Shares from an Oracle Linux Client

Note:

To be able to use the commands described in this section, use `yum` to install the `samba-client` and `cifs-utils` packages.

You can use the `findsmb` command to query a subnet for Samba servers. The command displays the IP address, NetBIOS name, workgroup, operating system and version for each server that it finds.

Alternatively, you can use the `smbtree` command, which is a text-based SMB network browser that displays the hierarchy of known domains, servers in those domains, and shares on those servers.

The GNOME and KDE desktops provide browser-based file managers that you can use to view Windows shares on the network. Enter `smb:` in the location bar of a file manager to browse network shares.

To connect to a Windows share from the command line, use the `smbclient` command:

```
smbclient //server_name/share_name [-U username]
```

After logging in, enter `help` at the `smb:\>` prompt to display a list of available commands.

To mount a Samba share, use a command such as the following:

```
sudo mount -t cifs //server_name/share_name mountpoint -o credentials=credfile
```

In the previous command, the credentials file contains settings for `username`, `password`, and `domain`, for example:

```
username=eddie  
password=clydenw  
domain=MYDOMWKG
```

The argument to `domain` can be the name of a domain or a workgroup.

 **Caution:**

As the credentials file contains a plain-text password, use `chmod` to make it readable only by you, for example:

```
sudo chmod 400 credfile
```

If the Samba server is a domain member server in an AD domain and your current login session was authenticated by the Kerberos server in the domain, you can use your existing session credentials by specifying the `sec=krb5` option instead of a credentials file:

```
sudo mount -t cifs //server_name/share_name mountpoint -o sec=krb5
```

For more information, see the `findsmb(1)`, `mount.cifs(8)`, `smbclient(1)`, and `smbtree(1)` manual pages.

7

Managing Oracle Cluster File System Version 2

This chapter describes how to configure and use Oracle Cluster File System Version 2 (OCFS2) file system.

About OCFS2

Oracle Cluster File System version 2 (OCFS2) is a general-purpose, high-performance, high-availability, shared-disk file system intended for use in clusters. It is also possible to mount an OCFS2 volume on a standalone, non-clustered system.

Although it might seem that there is no benefit in mounting `ocfs2` locally as compared to alternative file systems such as `ext4` or `btrfs`, you can use the `reflink` command with OCFS2 to create copy-on-write clones of individual files in a similar way to using the `cp --reflink` command with the `btrfs` file system. Typically, such clones allow you to save disk space when storing multiple copies of very similar files, such as VM images or Linux Containers. In addition, mounting a local OCFS2 file system allows you to subsequently migrate it to a cluster file system without requiring any conversion. Note that when using the `reflink` command, the resulting filesystem behaves like a clone of the original filesystem. This means that their UUIDs are identical. When using `reflink` to create a clone, you must change the UUID using the `tuneefs.ocfs2` command. See [Querying and Changing Volume Parameters](#) for more information.

Almost all applications can use OCFS2 as it provides local file-system semantics. Applications that are cluster-aware can use cache-coherent parallel I/O from multiple cluster nodes to balance activity across the cluster, or they can use of the available file-system functionality to fail over and run on another node in the event that a node fails. The following examples typify some use cases for OCFS2:

- Oracle VM to host shared access to virtual machine images.
- Oracle VM and VirtualBox to allow Linux guest machines to share a file system.
- Oracle Real Application Cluster (RAC) in database clusters.
- Oracle E-Business Suite in middleware clusters.

OCFS2 has a large number of features that make it suitable for deployment in an enterprise-level computing environment:

- Support for ordered and write-back data journaling that provides file system consistency in the event of power failure or system crash.
- Block sizes ranging from 512 bytes to 4 KB, and file-system cluster sizes ranging from 4 KB to 1 MB (both in increments of powers of 2). The maximum supported volume size is 16 TB, which corresponds to a cluster size of 4 KB. A volume size as large as 4 PB is theoretically possible for a cluster size of 1 MB, although this limit has not been tested.
- Extent-based allocations for efficient storage of very large files.

- Optimized allocation support for sparse files, inline-data, unwritten extents, hole punching, reflinks, and allocation reservation for high performance and efficient storage.
- Indexing of directories to allow efficient access to a directory even if it contains millions of objects.
- Metadata checksums for the detection of corrupted inodes and directories.
- Extended attributes to allow an unlimited number of `name:value` pairs to be attached to file system objects such as regular files, directories, and symbolic links.
- Advanced security support for POSIX ACLs and SELinux in addition to the traditional file-access permission model.
- Support for user and group quotas.
- Support for heterogeneous clusters of nodes with a mixture of 32-bit and 64-bit, little-endian (x86, x86_64, ia64) and big-endian (ppc64) architectures.
- An easy-to-configure, in-kernel cluster-stack (O2CB) with a distributed lock manager (DLM), which manages concurrent access from the cluster nodes.
- Support for buffered, direct, asynchronous, splice and memory-mapped I/O.
- A tool set that uses similar parameters to the `ext3` file system.

For more information about OCFS2, see <https://oss.oracle.com/projects/ocfs2/documentation/>.

Creating a Local OCFS2 File System

To create an OCFS2 file system that will be locally mounted and not associated with a cluster, use the following command:

```
sudo mkfs.ocfs2 -M local --fs-features=local -N 1 [options] device
```

For example, create a locally mountable OCFS2 volume on `/dev/sdc1` with one node slot and the label `localvol`:

```
sudo mkfs.ocfs2 -M local --fs-features=local -N 1 -L "localvol" /dev/sdc1
```

You can use the `tunefs.ocfs2` utility to convert a local OCFS2 file system to cluster use, for example:

```
sudo umount /dev/sdc1
sudo tunefs.ocfs2 -M cluster --fs-features=clusterinfo -N 8 /dev/sdc1
```

This example also increases the number of node slots from 1 to 8 to allow up to eight nodes to mount the file system.

Installing and Configuring OCFS2

This section describes procedures for setting up a cluster to use OCFS2.

Preparing a Cluster for OCFS2

For best performance, each node in the cluster should have at least two network interfaces. One interface is connected to a public network to allow general access to

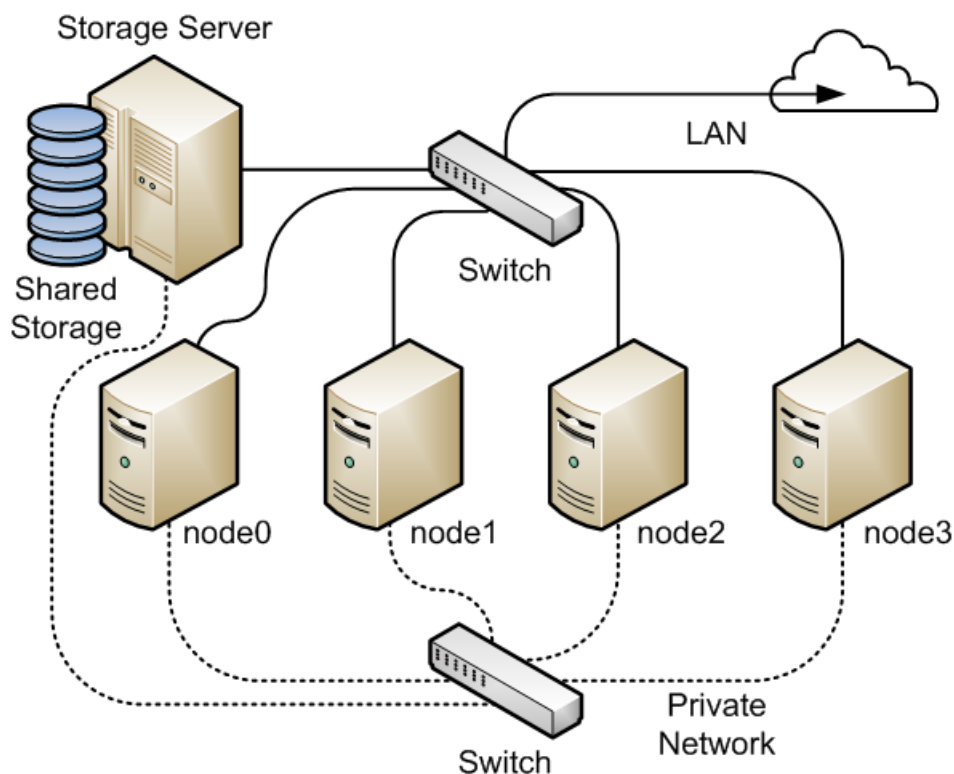
the systems. The other interface is used for private communication between the nodes; the *cluster heartbeat* that determines how the cluster nodes coordinate their access to shared resources and how they monitor each other's state. These interface must be connected via a network switch. Ensure that all network interfaces are configured and working before continuing to configure the cluster.

You have a choice of two cluster heartbeat configurations:

- Local heartbeat thread for each shared device. In this mode, a node starts a heartbeat thread when it mounts an OCFS2 volume and stops the thread when it unmounts the volume. This is the default heartbeat mode. There is a large CPU overhead on nodes that mount a large number of OCFS2 volumes as each mount requires a separate heartbeat thread. A large number of mounts also increases the risk of a node fencing itself out of the cluster due to a heartbeat I/O timeout on a single mount.
- Global heartbeat on specific shared devices. You can configure any OCFS2 volume as a global heartbeat device provided that it occupies a whole disk device and not a partition. In this mode, the heartbeat to the device starts when the cluster comes online and stops when the cluster goes offline. This mode is recommended for clusters that mount a large number of OCFS2 volumes. A node fences itself out of the cluster if a heartbeat I/O timeout occurs on more than half of the global heartbeat devices. To provide redundancy against failure of one of the devices, you should therefore configure at least three global heartbeat devices.

Figure 7-1 shows a cluster of four nodes connected via a network switch to a LAN and a network storage server. The nodes and the storage server are also connected via a switch to a private network that they use for the local cluster heartbeat.

Figure 7-1 Cluster Configuration Using a Private Network



It is possible to configure and use OCFS2 without using a private network but such a configuration increases the probability of a node fencing itself out of the cluster due to an I/O heartbeat timeout.

Configuring the Firewall

Configure or disable the firewall on each node to allow access on the interface that the cluster will use for private cluster communication. By default, the cluster uses both TCP and UDP over port 7777.

To allow incoming TCP connections and UDP datagrams on port 7777, use the following commands:

```
sudo firewall-cmd --zone=zone --add-port=7777/tcp --add-port=7777/udp
sudo firewall-cmd --permanent --zone=zone --add-port=7777/tcp --add-port=7777/udp
```

Configuring the Cluster Software

Ideally, each node should be running the same version of the OCFS2 software and a compatible version of the Oracle Linux Unbreakable Enterprise Kernel (UEK). It is possible for a cluster to run with mixed versions of the OCFS2 and UEK software, for example, while you are performing a rolling update of a cluster. The cluster node that is running the lowest version of the software determines the set of usable features.

Use `yum` to install or upgrade the following packages to the same version on each node:

- `kernel-uek`
- `ocfs2-tools`



Note:

If you want to use the global heartbeat feature, you must install `ocfs2-tools-1.8.0-11` or later.

Creating the Configuration File for the Cluster Stack

You can create the configuration file by using the `o2cb` command or a text editor.

To configure the cluster stack by using the `o2cb` command:

1. Use the following command to create a cluster definition.

```
sudo o2cb add-cluster cluster_name
```

For example, to define a cluster named `mycluster` with four nodes:

```
sudo o2cb add-cluster mycluster
```

The command creates the configuration file `/etc/ocfs2/cluster.conf` if it does not already exist.

2. For each node, use the following command to define the node.

```
sudo o2cb add-node cluster_name node_name --ip ip_address
```

The name of the node must be same as the value of system's `HOSTNAME` that is configured in `/etc/sysconfig/network`. The IP address is the one that the node will use for private communication in the cluster.

For example, to define a node named `node0` with the IP address `10.1.0.100` in the cluster `mycluster`:

```
sudo o2cb add-node mycluster node0 --ip 10.1.0.100
```

Note that OCFS2 only supports IPv4 addresses.

3. If you want the cluster to use global heartbeat devices, use the following commands.

```
sudo o2cb add-heartbeat cluster_name device1
.
.
.
sudo o2cb heartbeat-mode cluster_name global
```

 **Note:**

You must configure global heartbeat to use whole disk devices. You cannot configure a global heartbeat device on a disk partition.

For example, to use `/dev/sdd`, `/dev/sdg`, and `/dev/sdj` as global heartbeat devices:

```
sudo o2cb add-heartbeat mycluster /dev/sdd
sudo o2cb add-heartbeat mycluster /dev/sdg
sudo o2cb add-heartbeat mycluster /dev/sdj
sudo o2cb heartbeat-mode mycluster global
```

4. Copy the cluster configuration file `/etc/ocfs2/cluster.conf` to each node in the cluster.

 **Note:**

Any changes that you make to the cluster configuration file do not take effect until you restart the cluster stack.

The following sample configuration file `/etc/ocfs2/cluster.conf` defines a 4-node cluster named `mycluster` with a local heartbeat.

```
node:
  name = node0
  cluster = mycluster
  number = 0
  ip_address = 10.1.0.100
  ip_port = 7777

node:
  name = node1
  cluster = mycluster
  number = 1
  ip_address = 10.1.0.101
  ip_port = 7777

node:
```

```
name = node2
cluster = mycluster
number = 2
ip_address = 10.1.0.102
ip_port = 7777

node:
name = node3
cluster = mycluster
number = 3
ip_address = 10.1.0.103
ip_port = 7777

cluster:
name = mycluster
heartbeat_mode = local
node_count = 4
```

If you configure your cluster to use a global heartbeat, the file also include entries for the global heartbeat devices.

```
node:
name = node0
cluster = mycluster
number = 0
ip_address = 10.1.0.100
ip_port = 7777

node:
name = node1
cluster = mycluster
number = 1
ip_address = 10.1.0.101
ip_port = 7777

node:
name = node2
cluster = mycluster
number = 2
ip_address = 10.1.0.102
ip_port = 7777

node:
name = node3
cluster = mycluster
number = 3
ip_address = 10.1.0.103
ip_port = 7777

cluster:
name = mycluster
heartbeat_mode = global
node_count = 4

heartbeat:
cluster = mycluster
region = 7DA5015346C245E6A41AA85E2E7EA3CF

heartbeat:
cluster = mycluster
region = 4F9FBB0D9B6341729F21A8891B9A05BD
```

```
heartbeat:
    cluster = mycluster
    region = B423C7EEE9FC426790FC411972C91CC3
```

The cluster heartbeat mode is now shown as `global`, and the heartbeat regions are represented by the UUIDs of their block devices.

If you edit the configuration file manually, ensure that you use the following layout:

- The `cluster:`, `heartbeat:`, and `node:` headings must start in the first column.
- Each parameter entry must be indented by one tab space.
- A blank line must separate each section that defines the cluster, a heartbeat device, or a node.

Configuring the Cluster Stack


To configure the cluster stack:

1. Run the following command on each node of the cluster:

```
sudo /sbin/o2cb.init configure
```


The following table describes the values for which you are prompted.

Prompt	Description
Load O2CB driver on boot (y/n)	Whether the cluster stack driver should be loaded at boot time. The default response is <code>n</code> .
Cluster stack backing O2CB	The name of the cluster stack service. The default and usual response is <code>o2cb</code> .
Cluster to start at boot (Enter "none" to clear)	Enter the name of your cluster that you defined in the cluster configuration file, <code>/etc/ocfs2/cluster.conf</code> .
Specify heartbeat dead threshold (>=7)	The number of 2-second heartbeats that must elapse without response before a node is considered dead. To calculate the value to enter, divide the required threshold time period by 2 and add 1. For example, to set the threshold time period to 120 seconds, enter a value of 61. The default value is 31, which corresponds to a threshold time period of 60 seconds.

 **Note:**

If your system uses multipathed storage, the recommended value is 61 or greater.

Prompt	Description
Specify network idle timeout in ms (>=5000)	The time in milliseconds that must elapse before a network connection is considered dead. The default value is 30,000 milliseconds.
Specify network keepalive delay in ms (>=1000)	The maximum delay in milliseconds between sending keepalive packets to another node. The default and recommended value is 2,000 milliseconds.
Specify network reconnect delay in ms (>=2000)	The minimum delay in milliseconds between reconnection attempts if a network connection goes down. The default and recommended value is 2,000 milliseconds.

 **Note:**

For bonded network interfaces, the recommended value is 30,000 milliseconds or greater.

To verify the settings for the cluster stack, enter the `/sbin/o2cb.init status` command:

```
sudo /sbin/o2cb.init status

Driver for "configfs": Loaded
Filesystem "configfs": Mounted
Stack glue driver: Loaded
Stack plugin "o2cb": Loaded
Driver for "ocfs2_dlmfs": Loaded
Filesystem "ocfs2_dlmfs": Mounted
Checking O2CB cluster "mycluster": Online
  Heartbeat dead threshold: 61
  Network idle timeout: 30000
  Network keepalive delay: 2000
  Network reconnect delay: 2000
  Heartbeat mode: Local
Checking O2CB heartbeat: Active
```

In this example, the cluster is online and is using local heartbeat mode. If no volumes have been configured, the O2CB heartbeat is shown as `Not active` rather than `Active`.

The next example shows the command output for an online cluster that is using three global heartbeat devices:

```
# /sbin/o2cb.init status

Driver for "configfs": Loaded
Filesystem "configfs": Mounted
Stack glue driver: Loaded
Stack plugin "o2cb": Loaded
Driver for "ocfs2_dlmfs": Loaded
Filesystem "ocfs2_dlmfs": Mounted
Checking O2CB cluster "mycluster": Online
```

```
Heartbeat dead threshold: 61
Network idle timeout: 30000
Network keepalive delay: 2000
Network reconnect delay: 2000
Heartbeat mode: Global
Checking O2CB heartbeat: Active
7DA5015346C245E6A41AA85E2E7EA3CF /dev/sdd
4F9FBB0D9B6341729F21A8891B9A05BD /dev/sdg
B423C7EEE9FC426790FC411972C91CC3 /dev/sdj
```

2. Configure the `o2cb` and `ocfs2` services so that they start at boot time after networking is enabled:

```
sudo systemctl enable o2cb
sudo systemctl enable ocfs2
```

These settings allow the node to mount OCFS2 volumes automatically when the system starts.

Configuring the Kernel for Cluster Operation

For the correct operation of the cluster, you must configure the kernel settings shown in the following table:

Kernel Setting	Description
<code>panic</code>	Specifies the number of seconds after a panic before a system will automatically reset itself. If the value is 0, the system hangs, which allows you to collect detailed information about the panic for troubleshooting. This is the default value. To enable automatic reset, set a non-zero value. If you require a memory image (<code>vmcore</code>), allow enough time for <code>Kdump</code> to create this image. The suggested value is 30 seconds, although large systems will require a longer time.
<code>panic_on_oops</code>	Specifies that a system must panic if a kernel oops occurs. If a kernel thread required for cluster operation crashes, the system must reset itself. Otherwise, another node might not be able to tell whether a node is slow to respond or unable to respond, causing cluster operations to hang.

On each node, enter the following commands to set the recommended values for `panic` and `panic_on_oops`:

```
sudo sysctl kernel.panic=30
sudo sysctl kernel.panic_on_oops=1
```

To make the change persist across reboots, add the following entries to the `/etc/sysctl.conf` file:

```
# Define panic and panic_on_oops for cluster operation
kernel.panic=30
kernel.panic_on_oops=1
```

Starting and Stopping the Cluster Stack

The following list shows the commands that you can use to perform various operations on the cluster stack.

- `/sbin/o2cb.init status`: Check the status of the cluster stack.
- `/sbin/o2cb.init online`: Start the cluster stack.
- `/sbin/o2cb.init offline`: Stop the cluster stack.
- `/sbin/o2cb.init unload`: Unload the cluster stack.

Creating OCFS2 volumes

You can use the `mkfs.ocfs2` command to create an OCFS2 volume on a device. If you want to label the volume and mount it by specifying the label, the device must correspond to a partition. You cannot mount an unpartitioned disk device by specifying a label. The following table shows the most useful options that you can use when creating an OCFS2 volume.

Command Option	Description
<code>-bblock-size</code> <code>--block-sizeblock-size</code>	Specifies the unit size for I/O transactions to and from the file system, and the size of inode and extent blocks. The supported block sizes are 512 (512 bytes), 1K, 2K, and 4K. The default and recommended block size is 4K (4 kilobytes).
<code>-Ccluster-size</code> <code>--cluster-sizecluster-size</code>	Specifies the unit size for space used to allocate file data. The supported cluster sizes are 4K, 8K, 16K, 32K, 64K, 128K, 256K, 512K, and 1M (1 megabyte). The default cluster size is 4K (4 kilobytes).
<code>--fs-feature-level=feature-level</code>	Allows you select a set of file-system features: <p>default Enables support for the sparse files, unwritten extents, and inline data features.</p> <p>max-compat Enables only those features that are understood by older versions of OCFS2.</p> <p>max-features Enables all features that OCFS2 currently supports.</p>
<code>--fs_features=feature</code>	Allows you to enable or disable individual features such as support for sparse files, unwritten extents, and backup superblocks. For more information, see the <code>mkfs.ocfs2(8)</code> manual page.

Command Option	Description
<p><code>-Jsize=<i>journal-size</i></code> <code>--journal-optionssize=<i>journal-size</i></code></p>	<p>Specifies the size of the write-ahead journal. If not specified, the size is determined from the file system usage type that you specify to the <code>-T</code> option, and, otherwise, from the volume size. The default size of the journal is 64M (64 MB) for <code>datafiles</code>, 256M (256 MB) for <code>mail</code>, and 128M (128 MB) for <code>vmstore</code>.</p>
<p><code>-L<i>volume-label</i></code> <code>--label<i>volume-label</i></code></p>	<p>Specifies a descriptive name for the volume that allows you to identify it easily on different cluster nodes.</p>
<p><code>-N<i>number</i></code> <code>--node-slots<i>number</i></code></p>	<p>Determines the maximum number of nodes that can concurrently access a volume, which is limited by the number of node slots for system files such as the file-system journal. For best performance, set the number of node slots to at least twice the number of nodes. If you subsequently increase the number of node slots, performance can suffer because the journal will no longer be contiguously laid out on the outer edge of the disk platter.</p>
<p><code>-T<i>file-system-usage-type</i></code></p>	<p>Specifies the type of usage for the file system:</p> <p>datafiles Database files are typically few in number, fully allocated, and relatively large. Such files require few metadata changes, and do not benefit from having a large journal.</p> <p>mail Mail server files are typically many in number, and relatively small. Such files require many metadata changes, and benefit from having a large journal.</p> <p>vmstore Virtual machine image files are typically few in number, sparsely allocated, and relatively large. Such files require a moderate number of metadata changes and a medium sized journal.</p>

For example, create an OCFS2 volume on `/dev/sdc1` labeled as `myvol` using all the default settings for generic usage on file systems that are no larger than a few gigabytes. The default values are a 4 KB block and cluster size, eight node slots, a 256 MB journal, and support for default file-system features.

```
sudo mkfs.ocfs2 -L "myvol" /dev/sdc1
```

Create an OCFS2 volume on `/dev/sdd2` labeled as `dbvol` for use with database files. In this case, the cluster size is set to 128 KB and the journal size to 32 MB.

```
sudo mkfs.ocfs2 -L "dbvol1" -T datafiles /dev/sdd2
```

Create an OCFS2 volume on `/dev/sde1` with a 16 KB cluster size, a 128 MB journal, 16 node slots, and support enabled for all features except refcount trees.

```
sudo mkfs.ocfs2 -C 16K -J size=128M -N 16 --fs-feature-level=max-features --fs-features=norefcnt /dev/sde1
```

Note:

Do not create an OCFS2 volume on an LVM logical volume. LVM is not cluster-aware.

You cannot change the block and cluster size of an OCFS2 volume after it has been created. You can use the `tunefs.ocfs2` command to modify other settings for the file system with certain restrictions. For more information, see the `tunefs.ocfs2(8)` manual page.

If you intend the volume to store database files, do not specify a cluster size that is smaller than the block size of the database.

The default cluster size of 4 KB is not suitable if the file system is larger than a few gigabytes. The following table suggests minimum cluster size settings for different file system size ranges:

File System Size	Suggested Minimum Cluster Size
1 GB - 10 GB	8K
10GB - 100 GB	16K
100 GB - 1 TB	32K
1 TB - 10 TB	64K
10 TB - 16 TB	128K

Mounting OCFS2 Volumes

As shown in the following example, specify the `_netdev` option in `/etc/fstab` if you want the system to mount an OCFS2 volume at boot time after networking is started, and to unmount the file system before networking is stopped.

```
myocfs2vol1 /dbvol1 ocfs2 _netdev,defaults 0 0
```

Note:

The file system will not mount unless you have enabled the `o2cb` and `ocfs2` services to start after networking is started. See [Configuring the Cluster Stack](#).

Querying and Changing Volume Parameters

You can use the `tunefs.ocfs2` command to query or change volume parameters. For example, to find out the label, UUID and the number of node slots for a volume:

```
sudo tunefs.ocfs2 -Q "Label = %V\nUUID = %U\nNumSlots = %N\n" /dev/sdb
```

```
Label = myvol
UUID = CBB8D5E0C169497C8B52A0FD555C7A3E
NumSlots = 4
```

Generate a new UUID for a volume:

```
sudo tunefs.ocfs2 -U /dev/sda
sudo tunefs.ocfs2 -Q "Label = %V\nUUID = %U\nNumSlots = %N\n" /dev/sdb
```

```
Label = myvol
UUID = 48E56A2BBAB34A9EB1BE832B3C36AB5C
NumSlots = 4
```

Troubleshooting OCFS2

The following sections describes some techniques that you can use for investigating any problems that you encounter with OCFS2.

Recommended Tools for Debugging

To you want to capture an oops trace, it is recommended that you set up `netconsole` on the nodes.

If you want to capture the DLM's network traffic between the nodes, you can use `tcpdump`. For example, to capture TCP traffic on port 7777 for the private network interface `em2`, you could use a command such as the following:

```
sudo tcpdump -i em2 -C 10 -W 15 -s 10000 -Sw /tmp/`hostname`-s`_tcpdump.log -ttt 'port 7777' &
```

You can use the `debugfs.ocfs2` command, which is similar in behavior to the `debugfs` command for the `ext3` file system, and allows you to trace events in the OCFS2 driver, determine lock statuses, walk directory structures, examine inodes, and so on.

For more information, see the `debugfs.ocfs2(8)` manual page.

The `o2image` command saves an OCFS2 file system's metadata (including information about inodes, file names, and directory names) to an image file on another file system. As the image file contains only metadata, it is much smaller than the original file system. You can use `debugfs.ocfs2` to open the image file, and analyze the file system layout to determine the cause of a file system corruption or performance problem.

For example, the following command creates the image `/tmp/sda2.img` from the OCFS2 file system on the device `/dev/sda2`:

```
sudo o2image /dev/sda2 /tmp/sda2.img
```

For more information, see the `o2image(8)` manual page.

Mounting the debugfs File System

OCFS2 uses the `debugfs` file system to allow access from user space to information about its in-kernel state. You must mount the `debugfs` file system to be able to use the `debugfs.ocfs2` command.

To mount the `debugfs` file system, add the following line to `/etc/fstab`:

```
debugfs /sys/kernel/debug debugfs defaults 0 0
```

Then, run the `mount -a` command.

Configuring OCFS2 Tracing

The following table shows some of the commands that are useful for tracing problems in OCFS2.

Command	Description
<code>debugfs.ocfs2 -l</code>	List all trace bits and their statuses.
<code>debugfs.ocfs2 -l SUPER allow</code>	Enable tracing for the superblock.
<code>debugfs.ocfs2 -l SUPER off</code>	Disable tracing for the superblock.
<code>debugfs.ocfs2 -l SUPER deny</code>	Disallow tracing for the superblock, even if implicitly enabled by another tracing mode setting.
<code>debugfs.ocfs2 -l HEARTBEAT \</code> <code>ENTRY EXIT allow</code>	Enable heartbeat tracing.
<code>debugfs.ocfs2 -l HEARTBEAT off</code> <code>\</code> <code>ENTRY EXIT deny</code>	Disable heartbeat tracing. <code>ENTRY</code> and <code>EXIT</code> are set to <code>deny</code> as they exist in all trace paths.
<code>debugfs.ocfs2 -l ENTRY EXIT \</code> <code>NAMEI INODE allow</code>	Enable tracing for the file system.
<code>debugfs.ocfs2 -l ENTRY EXIT \</code> <code>deny NAMEI INODE allow</code>	Disable tracing for the file system.
<code>debugfs.ocfs2 -l ENTRY EXIT \</code> <code>DLM DLM_THREAD allow</code>	Enable tracing for the DLM.
<code>debugfs.ocfs2 -l ENTRY EXIT \</code> <code>deny DLM DLM_THREAD allow</code>	Disable tracing for the DLM.

One method for obtaining a trace is to enable the trace, sleep for a short while, and then disable the trace. As shown in the following example, to avoid seeing unnecessary output, you should reset the trace bits to their default settings after you have finished.

```
sudo debugfs.ocfs2 -l ENTRY EXIT NAMEI INODE allow && sleep 10 && debugfs.ocfs2 -l ENTRY EXIT deny NAMEI INODE off
```

To limit the amount of information displayed, enable only the trace bits that you believe are relevant to understanding the problem.

If you believe a specific file system command, such as `mv`, is causing an error, the following example shows the commands that you can use to help you trace the error.

```
sudo debugfs.ocfs2 -l ENTRY EXIT NAMEI INODE allow
sudo mv source destination & CMD_PID=$(jobs -p %-)
echo $CMD_PID
sudo debugfs.ocfs2 -l ENTRY EXIT deny NAMEI INODE off
```

As the trace is enabled for all mounted OCFS2 volumes, knowing the correct process ID can help you to interpret the trace.

For more information, see the `debugfs.ocfs2(8)` manual page.

Debugging File System Locks

If an OCFS2 volume hangs, you can use the following steps to help you determine which locks are busy and the processes that are likely to be holding the locks.

1. Mount the debug file system.

```
sudo mount -t debugfs debugfs /sys/kernel/debug
```

2. Dump the lock statuses for the file system device (`/dev/sdx1` in this example).

```
echo "fs_locks" | debugfs.ocfs2 /dev/sdx1 >/tmp/fslocks 62
```

```
Lockres: M00000000000006672078b84822 Mode: Protected Read
Flags: Initialized Attached
RO Holders: 0 EX Holders: 0
Pending Action: None Pending Unlock Action: None
Requested Mode: Protected Read Blocking Mode: Invalid
```

The `Lockres` field is the lock name used by the DLM. The lock name is a combination of a lock-type identifier, an inode number, and a generation number. The following list shows the possible lock types:

- D: File data.
- M: Metadata
- R: Rename
- S: Superblock
- W: Read-write

3. Use the `Lockres` value to obtain the inode number and generation number for the lock.

```
sudo echo "stat <M00000000000006672078b84822>" | debugfs.ocfs2 -n /dev/sdx1
```

```
Inode: 419616 Mode: 0666 Generation: 2025343010 (0x78b84822)
...
```

4. Determine the file system object to which the inode number relates by using the following command.

```
echo "locate <419616>" | debugfs.ocfs2 -n /dev/sdx1
```

```
419616 /linux-2.6.15/arch/i386/kernel/semaphore.c
```

5. Obtain the lock names that are associated with the file system object.

```
echo "encode /linux-2.6.15/arch/i386/kernel/semaphore.c" | debugfs.ocfs2 -
n /dev/sdx1
```

```
M00000000000006672078b84822 D0000000000006672078b84822
W00000000000006672078b84822
```

In this example, a metadata lock, a file data lock, and a read-write lock are associated with the file system object.

6. Determine the DLM domain of the file system.

```
echo "stats" | debugfs.ocfs2 -n /dev/sdX1 | grep UUID: | while read a b ; do
echo $b ; done
```

```
82DA8137A49A47E4B187F74E09FBBB4B
```

7. Use the values of the DLM domain and the lock name with the following command, which enables debugging for the DLM.

```
echo R 82DA8137A49A47E4B187F74E09FBBB4B M0000000000006672078b84822 > /
proc/fs/ocfs2_dlm/debug
```

8. Examine the debug messages.

```
sudo dmesg | tail
```

```
struct dlm_ctxt: 82DA8137A49A47E4B187F74E09FBBB4B, node=3, key=965960985
  lockres: M0000000000006672078b84822, owner=1, state=0 last used: 0,
  on purge list: no granted queue:
    type=3, conv=-1, node=3, cookie=11673330234144325711,
ast=(empty=y,pend=n),
  bast=(empty=y,pend=n)
  converting queue:
  blocked queue:
```

The DLM supports 3 lock modes: no lock (`type=0`), protected read (`type=3`), and exclusive (`type=5`). In this example, the lock is owned by node 1 (`owner=1`) and node 3 has been granted a protected-read lock on the file-system resource.

9. Run the following command, and look for processes that are in an uninterruptable sleep state as shown by the `D` flag in the `STAT` column.

```
ps -e -o pid,stat,comm,wchan=WIDE-WCHAN-COLUMN
```

At least one of the processes that are in the uninterruptable sleep state will be responsible for the hang on the other node.

If a process is waiting for I/O to complete, the problem could be anywhere in the I/O subsystem from the block device layer through the drivers to the disk array. If the hang concerns a user lock (`flock()`), the problem could lie in the application. If possible, kill the holder of the lock. If the hang is due to lack of memory or fragmented memory, you can free up memory by killing non-essential processes. The most immediate solution is to reset the node that is holding the lock. The DLM recovery process can then clear all the locks that the dead node owned, so letting the cluster continue to operate.

Configuring the Behavior of Fenced Nodes

If a node with a mounted OCFS2 volume believes that it is no longer in contact with the other cluster nodes, it removes itself from the cluster in a process termed *fencing*. Fencing prevents other nodes from hanging when they try to access resources held by the fenced node. By default, a fenced node restarts instead of panicking so that it can

quickly rejoin the cluster. Under some circumstances, you might want a fenced node to panic instead of restarting. For example, you might want to use `netconsole` to view the oops stack trace or to diagnose the cause of frequent reboots. To configure a node to panic when it next fences, run the following command on the node after the cluster starts:

```
echo panic > /sys/kernel/config/cluster/cluster_name/fence_method
```

In the previous command, `cluster_name` is the name of the cluster. To set the value after each reboot of the system, add this line to the `/etc/rc.local` file. To restore the default behavior, use the value `reset` instead of `panic`.

Use Cases for OCFS2

The following sections describe some typical use cases for OCFS2.

Load Balancing

You can use OCFS2 nodes to share resources between client systems. For example, the nodes could export a shared file system by using Samba or NFS. To distribute service requests between the nodes, you can use round-robin DNS, a network load balancer, or specify which node should be used on each client.

Oracle Real Application Cluster (RAC)

Oracle RAC uses its own cluster stack, Cluster Synchronization Services (CSS). You can use O2CB in conjunction with CSS, but you should note that each stack is configured independently for timeouts, nodes, and other cluster settings. You can use OCFS2 to host the voting disk files and the Oracle cluster registry (OCR), but not the grid infrastructure user's home, which must exist on a local file system on each node.

As both CSS and O2CB use the lowest node number as a tie breaker in quorum calculations, you should ensure that the node numbers are the same in both clusters. If necessary, edit the O2CB configuration file `/etc/ocfs2/cluster.conf` to make the node numbering consistent, and update this file on all nodes. The change takes effect when the cluster is restarted.

Oracle Databases

Specify the `noatime` option when mounting volumes that host Oracle datafiles, control files, redo logs, voting disk, and OCR. The `noatime` option disables unnecessary updates to the access time on the inodes.

Specify the `nointr` mount option to prevent signals interrupting I/O transactions that are in progress.

By default, the `init.ora` parameter `filesystemio_options` directs the database to perform direct I/O to the Oracle datafiles, control files, and redo logs. You should also specify the `datavolume` mount option for the volumes that contain the voting disk and OCR. Do not specify this option for volumes that host the Oracle user's home directory or Oracle E-Business Suite.

To avoid database blocks becoming fragmented across a disk, ensure that the file system cluster size is at least as big as the database block size, which is typically 8KB. If you specify the file system usage type as `datafiles` to the `mkfs.ocfs2` command, the file system cluster size is set to 128KB.

To allow multiple nodes to maximize throughput by concurrently streaming data to an Oracle datafile, OCFS2 deviates from the POSIX standard by not updating the modification time (`mtime`) on the disk when performing non-extending direct I/O writes. The value of `mtime` is updated in memory, but OCFS2 does not write the value to disk unless an application extends or truncates the file, or performs a operation to change the file metadata, such as using the `touch` command. This behavior leads to results in different nodes reporting different time stamps for the same file. You can use the following command to view the on-disk timestamp of a file:

```
sudo debugfs.ocfs2 -R "stat /file_path" device | grep "mtime:"
```