# Oracle Linux 9
# Managing Local File Systems

ORACLE®

Oracle Linux 9 Managing Local File Systems,

F52807-02

# Contents

## Preface

## 1   About File System Management in Oracle Linux

## 2   Performing Basic File System Administration

## 3    Managing the Btrfs File System

## 4    Managing the Ext File System

## 5    Managing the XFS File System

# Preface

Oracle Linux 9: Managing Local File Systems provides information about managing file systems and storage devices on Oracle Linux 9 systems.

## 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. |
| *italic* | 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. Information about supported file systems and minimum requirements for each supported file system type is also provided.

## Supported File Systems

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

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

> **Note:**
>
> In Oracle Linux 9, the Btrfs file system type is supported on Unbreakable Enterprise Kernel (UEK) releases *only*.

**ext4**
A version of the extended file system. Ext4 supports all of the features that are supported by Ext3, with addition support for extents (contiguous physical blocks), pre-allocation, delayed allocation, speedier file system checking, more robust journaling, and several other enhancements.

**XFS**
A high-performance, journaling file system that provides high scalability for I/O threads, file system bandwidth, and file and file system size, even for file systems that span many storage devices.

To list the file system types that your system supports, use the following command:

```
sudo ls /sbin/mkfs.*
```

Note that the following output might differ, depending on your setup:

```
sudo ls /sbin/mkfs.*
/sbin/mkfs.cramfs   /sbin/mkfs.ext4   /sbin/mkfs.msdos
/sbin/mkfs.ext2     /sbin/mkfs.fat    /sbin/mkfs.vfat
/sbin/mkfs.ext3     /sbin/mkfs.minix  /sbin/mkfs.xfs
```

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

# Maximum File and File System Size Requirements

File system limitations are affected by kernel versions and features, as well as the architecture of the system on which Oracle Linux is installed. The values that are shown in the table are estimates, based on the known variables that might affect the maximum theoretical value that can be achieved. The theoretical values might be higher than those depicted here, while the actual, achievable values might be lower than the values that are shown, depending on the hardware and kernel version that is used.

The following table describes the maximum file size and maximum file system size for the `btrfs`, `ext4`, and `xfs` file systems.

| File System Type | Maximum File Size | Maximum File System Size | Supported Kernels |
| --- | --- | --- | --- |
| `btrfs` | 8 EiB | 8 EiB | UEK, starting with Unbreakable Enterprise Kernel Release 7 (UEK R7) |
| `ext4` | 16 TiB | 1 EiB | RHCK and UEK |
| `xfs` | 8 EiB | 8 EiB | RHCK and UEK |

The limits for the `ext4` file system that are described in the previous table are higher than those recommended and might prove unstable. If you intend for the systems you work on to eventually use higher file system sizes or file sizes, then using either the Btrfs or XFS file system is recommended.

The maximum supported size for a bootable logical unit number (LUN) is 50 TB. GPT and UEFI support are required for LUNs that are larger than 2 TB.

The maximum size of the address space that is available to each process is 128 TB.

For information about maximum file sizes and maximum file system sizes for OCFS2, see the chapter on OCFS2 in Oracle Linux 9: Managing Shared File Systems.

# 2
# Performing Basic File System Administration

This chapter describes basic tasks for administering file systems. The chapter also describes how to configure Access Control Lists (ACLs) and how to configure and manage disk quotas.

## Building File Systems

The `mkfs` command syntax enables you to build a file system on a block device:

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

Typically, the `-t fstype` and `-L label` options are used with the `mkfs`command. The following example builds an `ext4` file system with the label Project:

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

If you do not specify the file system type, an `ext2` file system is created by default.

As an alternative, you can omit `-t fstype` and instead use the appropriate full `mkfs.<extension>` command as listed in `/sbin`. The following command produces the same result as the previous command:

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

To display the file system type, use the `blkid` command, for example:

```
sudo blkid /dev/sdb1
```

The output of the previous command would be similar to the following:

```
/dev/sdb1: UUID="ad8113d7-b279-4da8-b6e4-cfba045f66ff" BLOCK_SIZE="512" TYPE="ext4"
  PARTUUID="PARTUUID="6a0cf5e9-09e5-40cf-ab47-3166e1c60f24" LABEL="Projects"
```

Each file system type supports a number of features that you can enable or disable by specifying additional options with either the simple `mkfs` command format or the full `mkfs.<extension>` command. For example, you can use the `-J` option to specify the size and location of the journal that is used by the `ext*` 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 need to attach its block device to a mount point in the directory hierarchy. Any directory can be used to function as a mount point.

Typically, you create a new directory for a mount point. If you use an existing directory, the contents remain hidden until you unmount the overlying file system.

# About the mount Command

You use the `mount` command to attach the device containing the file system to the mount point as follows:

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

The device can be mounted by referencing its name, UUID, or label. For example, to mount the file system that was created in the previous section to `/var/projects`, any of the following commands can be used after you create the directory by running the following commands:

```
sudo mkdir /var/projects
```

```
sudo mount /dev/sdb1 /var/projects
```

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

```
sudo mount LABEL="Projects" /var/projects
```

Issuing the `mount` command by itself displays all of the currently mounted file systems. In the following example, an extract of the command's output indicates the following:

- `/dev/sdb1` with an `ext4` file system is mounted on `/var/projects` for both reading and writing

- `/dev/mapper/vg_host01-lv_root`, an LVM logical volume also with an `ext4` file system, is mounted on `/` for both reading and writing:

```
sudo mount
```

The output of the previous command would be similar to the following:

```
/dev/sdb1 on /var/projects type ext4 (rw)
/dev/mapper/vg_host01-lv_root on / type ext4 (rw)
...
```

Alternatively, you can use the `cat /proc/mounts` command to display information about mounted file systems.

The `df -h` command displays information about file systems and their use of disk space:

```
Filesystem      Size  Used Avail Use% Mounted on
...
/dev/sda3        46G   18G   29G  39% /
/dev/sda2       795M  452M  344M  57% /boot
/dev/sda1       100M  5.7M   95M   6% /boot/efi
...
```

To attach or bind a block device at multiple mount points, use the `mount -B` command.

You can also remount part of a directory hierarchy, which need not be a complete file system, somewhere else. For example, you would use the following command to mount `/var/projects/project1` on `/mnt`, for example:

```
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. However, any submounts are not replicated. These mirrors do not provide data redundancy.

To mount a file over another file, you would use the following command:

```
sudo touch /mnt/foo
```

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

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

To include submounts in the mirror, use the `-R` option to execute a recursive bind.

When you use the `-B` or `-R` option, 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 slave . You can specify the following options:

**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:

```
sudo touch /mnt/foo
```

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

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

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

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

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

# Using Additional Options of the mount Command

You can further determine the `mount` command behavior by using the `-o` option to specify *options* in a comma-separated list. Some of these options are as follows:

> **Note:**
>
> These options can also be entered in the `/etc/fstab` file.

**auto**
Causes the file system to be mounted automatically by using the `mount -a` command.

**exec**
Causes 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 That Contains a File System Image, Creating a File System on a File Within Another File System, 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 then reboot the system.

**noauto**
Prevents the file system from being mounted automatically when `mount -a` is issued.

**noexec**
Prevents the execution of any binary files located in the file system.

**nouser**
Prevents any user other than the `root` user 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.

The following examples show different ways to use the `mount -o` command syntax.

- Mount the `/dev/sdd1` file system as `/test` with read-only access and permit only the `root` user 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, and do not permit the execution of any binary files that are located in the file system:

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

## Mounting a File That Contains a File System Image

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

```
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 the `/etc/fstab` file, for example:

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

## About the File System Mount Table

The `/etc/fstab` file contains the file system mount table, which provides all of the information that the `mount` command requires to mount block devices or implement binding of mounts. If you add a file system, you must create the appropriate entry in the `/etc/fstab` file to ensure that the file system is mounted at boot time. The following are typical entries from this file:

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

The descriptions of each field in the previous output are as follows:

- The first field indicates the device to mount, which is 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
  ```

  Note that 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 third and fourth fields are specified as `none` and `bind`.

- The second field is either the mount point for a file system or `swap` to indicate a swap partition. The mount point must be a path to either a file or a directory.

- The third field is the file system type, such as `ext4` or `swap`.

- The fourth field specifies any mount options.

- The fifth column specifies whether the `dump` command should dump the file system (`1`) or not (`0`).

- The sixth column identifies the order by which the `fsck` command performs a file system check at boot time. The root file system should have the value `1`, while other file systems have `2`. A value of `0` skips checking, as is appropriate for swap, for file systems that are not mounted at boot time, and for binding of existing mounts.

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

```
path      mount_point    none      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 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. It also references entries in `/etc/auto.master`, which is the automounter configuration file.

## About the Automounter Configuration File

In the `/etc/auto.master` configuration file, 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
```

In the previous example, 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 similar to the following:

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

This entry is a directive to do the following:

- Mount the file system `/usr/man` that is exported by `host01` by specifying the `ro` and `soft` options.

- Create the `/usr/man` mount point if it does not already exist. If the mount point exists , the mounted file system hides any existing files that it contains.

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

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

An indirect map contains definitions of directories or *keys* that are automounted relative to the mount point (`/misc`) that is specified in the `/etc/auto.master` file. For example, the `/etc/auto.misc` map file might contain entries similar to 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
```

Note that the `/misc` directory must already exist; however, the automounter creates a mount point for the keys `xyz`, `cd` , and so on, if they do not already exist, and then removes them when it unmounts the file system.

For example, using the `ls /misc/xyz` command causes the automounter to the mount the `/xyz` directory, exported by `host01` as `/misc/xyz`.

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

```
/net        -hosts    -suid,dev,nointr
```

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

For more information about NFS administration, see the Using NFS in Oracle Linux chapter in Oracle Linux 8: Managing Shared File Systems Oracle Linux 9: Managing Shared File Systems. See also the `hosts.master(5)` and `auto.master(5)` manual pages.

## Installing and Enabling the Automounter

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 dnf install autofs
```

2. Edit the `/etc/auto.master` configuration file to define map entries that are appropriate to your file systems.

    See About the Automounter Configuration File for reference.

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

   ```
   sudo systemctl start autofs
   ```

   ```
   sudo systemctl enable autofs
   ```

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

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

```
sudo systemctl restart autofs
```

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

# Creating a File System on a File Within Another File System

1. Create an empty file of the required size:

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

   The output of the previous command would be as follows:

   ```
   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
   ```

   The output of the previous command would be as follows:

   ```
   mke2fs 1.44.6 (5-Mar-2019)
   Discarding device blocks: done
   Creating filesystem with 250000 4k blocks and 62592 inodes
   Filesystem UUID: 17ef1d96-c595-4f19-891b-112a56b54c82
   Superblock backups stored on blocks:
       32768, 98304, 163840, 229376

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

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 when you run the `sudo mount` command:

   ```
   ...
   /fsfile on /mnt type ext4 (rw,loop=/dev/loop0)
   ```

   ```
   sudo 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
```

# 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 the `btrfs`, `ext3`, `ext4`, `OCFS2`, and `XFS` file systems, as well as 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.

## Enabling ACL Support

1.  Ensure that the `acl` package is installed; and, if necessary install it as follows:

    ```
    sudo dnf install acl
    ```

2.  Edit the `/etc/fstab` file and change the entries for any file systems that 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:

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

## Setting and Displaying ACLs

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

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

ACL rules accept 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 as follows: `r`, `w`, and `x` for read, write, and execute, respectively, as used with the `chmod` command.

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
```

The output of this command would be as follows:

```
# 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 `ls -l` command displays a plus sign (+) after the permissions:

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

The following examples show additional ways in which you would set and display ACLs for directories and files:

To grant read access to a file or directory by a user:

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

To display the name, owner, group, and ACL for a file or directory:

```
sudo getfacl file
```

To 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
```

Note that the `-x` option removes rules for a user or group.

To remove the rules for a user from the ACL of a file use the following command:

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

To remove the rules for a group from the ACL of a file, use the following command:

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

Specifying the `-b` option removes all of the extended ACL entries from a file or directory, for example:

```
sudo setfacl -b file
```

To copy the ACL of file *f1* to file *f2*, use the following command:

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

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

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

To promote the ACL settings of a directory to default ACL settings that can be inherited, use the following command:

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

Specify the `-k` option to remove the default ACL from a directory, for example:

```
sudo setfacl -k directory
```

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

# About Disk Quotas

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 that are available to a user or group on the file system. Users or groups can exceed a soft limit for a period of time, which is known as a *grace period*.

Oracle Linux 9 does not provide support for user and group disk quotas for a Btrfs file system. However, quota support at the subvolume level is available for a Btrfs file system as a technology preview in this release. For more information, see Managing Quotas for Btrfs Subvolumes With Quota Groups.

For information about how to configure quotas for an XFS file system, see Managing Quotas on an XFS File System.

## Enabling Disk Quotas on File Systems

Disk quotas are enabled at mount. The following table describes the options that you can specify with the `mount` command to enable quotas.

| Mount Option | Description |
| --- | --- |
| `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. |

| Mount Option | Description |
|---|---|
| `uquota` | Enable user quotas and enforce usage limits. |

To enable quotas for a file system:

1. Install the `quota` package on the system, if it is not already installed:

   ```
   sudo dnf install quota
   ```

2. Add the `usrquota` or `grpquota` options to the file system's `/etc/fstab` entry:

   ```
   /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
   ```

   The previous command creates the files `aquota.user` and `aquota.group` in the root of the file system, which is `/home` in this example.

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

## Assigning Disk Quotas to Users and Groups

1. For a user, use the following command:

   ```
   sudo edquota username
   ```

   for a group, use the following command:

   ```
   sudo edquota -g group
   ```

   Running the previous command opens a text file opens in the default editor that is defined by the `EDITOR` environment variable, thus allowing you to 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 reflect the user's current 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 the number of blocks and inodes, then save the changes.

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

Note that when using XFS file systems, `xfs_quota` is the preferred tool to manage quota information. See Managing Quotas on an XFS File System for more information.

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

# Setting Project Quotas

Some file systems enable you to set quotas on individual directory hierarchies, which are known as *managed trees*. Each managed tree is uniquely identified by a *project ID* and an optional *project name*. The ability 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 for it:

1.  Mount the file system with project quotas enabled.

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

    For example, to enable project quotas for the `/myxfs` file system, you would use the following command:

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

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

    ```
    sudo echo project_ID:mountpoint/directory |sudo tee -a /etc/projects
    ```

    For example, you would set a project ID of 51 for the directory hierarchy `/myxfs/testdir` as follows:

    ```
    sudo echo 51:/myxfs/testdir |sudo tee -a /etc/projects
    ```

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

    ```
    sudo echo project_name:project_ID |sudo tee -a /etc/projid
    ```

    For example, you would map the project name `testproj` to the project with ID 51 as follows:

    ```
    sudo echo testproj:51 |sudo tee -a /etc/projid
    ```

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

With the file system mounted to enable project quotas and project IDs set for your directory hierarchy, you can set limits for the project quota using `edquota` or `xfs_quota`. Note that when using XFS file systems, `xfs_quota` is preferred. See Managing Quotas on an XFS File System for more information.

# Setting a Grace Period for Soft Limits

1.  Run the following command to set a grace period for soft limits:

    ```
    sudo edquota -t
    ```

    Running the previous command opens a text file in your default text editor, thus enabling you to specify the grace period, as shown in the following 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. Specify the grace periods for the soft limits on the number of blocks and inodes, then save the changes.

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

## Displaying Disk Quotas

To display a user's disk usage, use the `quota` command without any options or arguments:

```
sudo quota username
```

To display a group's disk usage, add the `-g` option, use the following command:

```
sudo quota -g group
```

To display information about file systems, where usage is over the quota limits, add the `-q` option, for example:

```
sudo quota -q
```

Users can also use the `quota` command to display disk usage for themselves and their group.

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, use the following command:

```
sudo quotaoff -guv filesystem
```

To disable disk quotas for all users, groups, and file systems, use the following command:

```
sudo quotaoff -aguv
```

Reactivate disk quotas for all users, groups, and file systems as follows:

```
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. 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 quotaoff -guv filesystem
   ```

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

# 3

# Managing the Btrfs File System

The Btrfs file system is designed to meet the expanding scalability requirements of large storage subsystems. Because the Btrfs file system uses B-trees in its implementation, its name is derived 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, irrespective of how large the tree grows.

The Btrfs file system provides the following important features:

- Copy-on-write functionality, which enables you to create both readable and writable snapshots and roll back a file system to a previous state, even after converting it from an `ext3` or `ext4` file system.

- Checksum functionality, which ensures data integrity.

- Snapshots send and receive for remote incremental backups.

- Transparent compression, which saves disk space.

- Transparent defragmentation for improved performance.

- Integrated, logical volume management, which enables you to implement RAID 0, RAID 1, RAID 10, RAID1C3 or RAID1C4 configurations, and dynamically add or remove storage capacity.

For more information, visit https://btrfs.wiki.kernel.org/.

For an overview of local file system management, see About File System Management in Oracle Linux.

> **Note:**
>
> In Oracle Linux 9, the Btrfs file system type, as well as all of the features that are documented in this chapter is supported on the Unbreakable Enterprise Kernel (UEK) release *only*. Working with Btrfs file system features requires that you boot the system by using UEK R7.

## Setting Up and Administering a Btrfs File System

This section describes procedures to create a Btrfs file system as well as administering it for a more efficient use.

# Creating a Btrfs File System

> **Note:**
>
> Note that the `/sbin/mkfs.*` tool is not provided by the same package as the kernel modules that are required by the `btrfs` command. Use the `modinfo btrfs` command to check whether the btrfs module is available in the kernel that is booted. Also, if the `btrfs-progs` package is not installed on your system, you will need to install it.

You can use the `mkfs.btrfs` command to create a Btrfs file system that is laid out across one or more block devices. The default configuration is to mirror the filesystem metadata across the devices. If you specify a single device, the metadata is duplicated on that device, unless you specify to use only one copy of the metadata. The devices can be whole block device(s), simple disk partitions, files, loopback devices (that is, disk images in memory), multipath devices, or LUNs that implement RAID in hardware.

See the `mkfs.btrfs(8)` manual page for more detailed information about the `mkfs.btrfs` command the various Btrfs configurations that you can create.

When you want to mount the file system, you can specify it by any of its component devices, for example:

```
sudo mkfs.btrfs -d raid10 -m raid10 /dev/sd[fghijk]

sudo mount /dev/sdf mountpoint
```

The `mkfs.btrfs` command automatically scans the new Btrfs devices into the kernel. To scan and assemble all of the relevant devices of the volume in the kernel you need to run the `btrfs device scan` command. You can undo this action by using the `btrfs device scan --forget` command. To assemble all of the relevant devices of the volume, you might need to run the `btrfs device scan` command.

You can obtain the RAID configuration for a mounted `btrfs` file system as follows:

```
sudo btrfs filesystem df mountpoint
```

Note that the `btrfs filesystem df` command displays more accurate information about the space that is used by a Btrfs file system than by using the `df` command.

You can also use the `btrfs filesystem usage` command to display information about all of the `btrfs` file systems that are on a system, for example:

```
sudo btrfs filesystem usage /btrfs

Overall:
    Device size: 1.95TiB
    Device allocated: 5.03GiB
    Device unallocated: 1.95TiB
    Device missing: 0.00B
    Used: 256.00KiB
    Free (estimated): 999.48GiB (min: 999.48GiB)
    Data ratio: 2.00
    Metadata ratio: 2.00
```

```
      Global reserve: 3.25MiB (used: 0.00B)

Data,RAID10: Size:2.00GiB, Used:0.00B (0.00%)
   /dev/nvme0n1p6 1.00GiB
   /dev/nvme0n1p7 1.00GiB
   /dev/nvme0n1p8 1.00GiB
   /dev/nvme0n1p9 1.00GiB

Metadata,RAID10: Size:512.00MiB, Used:112.00KiB (0.02%)
   /dev/nvme0n1p6 256.00MiB
   /dev/nvme0n1p7 256.00MiB
   /dev/nvme0n1p8 256.00MiB
   /dev/nvme0n1p9 256.00MiB

System,RAID10: Size:16.00MiB, Used:16.00KiB (0.10%)
   /dev/nvme0n1p6 8.00MiB
   /dev/nvme0n1p7 8.00MiB
   /dev/nvme0n1p8 8.00MiB
   /dev/nvme0n1p9 8.00MiB

Unallocated:
   /dev/nvme0n1p6 498.74GiB
   /dev/nvme0n1p7 498.74GiB
   /dev/nvme0n1p8 498.74GiB
   /dev/nvme0n1p9 498.74GiB
```

## Modifying a Btrfs File System

You can use the `btrfs` command to add or remove devices and rebalance the layout of the file system data and metadata across devices. The following table describes each of the commands that can use to perform these tasks.

| Command | Description |
|---|---|
| `btrfs device add` *device mountpoint* | Add a device to the file system that is mounted on the specified mount point, for example:<br><br>`btrfs device add /dev/sdd /myfs` |
| `btrfs device delete` *device mountpoint* | Remove a device from a mounted file system, for example:<br><br>`btrfs device delete /dev/sde / myfs` |
| `btrfs device delete missing` *mountpoint* | Remove a failed device from the file system that is mounted in degraded mode, for example:<br><br>`btrfs device remove missing /myfs`<br><br>To mount a file system in degraded mode, specify the `-o degraded` option to the `mount` command.<br><br>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. |

| Command | Description |
|---|---|
| `btrfs filesystem balance` *mountpoint* | 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, as well as defragment it to increase I/O performance.

The following three compression types are supported:

- `zlib`

- `lzo`

- `zstd`

There are three ways in which you can enable compression:

- By using the subvolume property, for example:

  ```
  sudo btrfs subvolume create /btrfs/sv1
  ```

  ```
  sudo btrfs property set /btrfs/sv1 compression zstd
  ```

- By using the mount option in one of the following ways:

  Use the `lzo` type to compress the file data on the whole file system:

  ```
  sudo mount -o compress=lzo /dev/sdb /btrfs
  ```

  Use the `zstd` type to compress a subvolume and then mount it at the `/btrfs1` mount point:

  ```
  sudo mount -o compress=zstd,subvolume=sv1 /devsdb /btrfs1
  ```

- By using defragment, for example:

  ```
  sudo btrfs filesystem defragment -czlib /btrfs1/akzo
  ```

You can enable compression at any point and only the new writes are compressed or defragmentation is run.

You can 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, use the following command:

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

You can use the following command to defragment and optionally compress individual file system objects such as directories and files within a Btrfs file system:

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

To set up automatic defragmentation, specify the `autodefrag` option when you mount the file system. However, note that automatic defragmentation is not recommended for large databases or for images of virtual machines.

> **Note:**
>
> Defragmenting a file or a subvolume with a copy-on-write copy 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 as long as there is space on the underlying devices to accommodate the change. You also use the `btrfs` command to decrease its size, if the file system has sufficient available free space. Note that running the command does not have any effect on the layout or size of the underlying devices.

You would increase the size of `/mybtrfs1` by 2 GB as follows:

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

The following example shows how to set the size of `/mybtrfs3` to 20 GB:

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

The following command decreases the size of `/mybtrfs2` by 4 GB:

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

## Creating Subvolumes and Snapshots

The top level of a Btrfs file system is a subvolume consisting of a named b-tree structure containing directories, files, and possibly further `btrfs` subvolumes that are also named b-trees, each of which also contains directories and files, and so on.

To create a subvolume, change directories to the position in the `btrfs` file system for which you want to create the subvolume, then run the following command:

```
sudo btrfs subvolume create subvolume_name
```

Snapshots are a type of subvolume that record 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 a alternate version of the original file system. The copy-on-write functionality of a `btrfs` file system means that snapshots are created quickly 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 that snapshot.

The following table describes commands to use to perform some common snapshot operations.

| Command | Description |
|---------|-------------|
| `btrfs subvolume snapshot` *pathname pathname* / *snapshot_path* | Create a snapshot *snapshot_path* of a parent subvolume or snapshot specified by *pathname*, for example:<br>`btrfs subvolume snapshot / mybtrfs /mybtrfs/snapshot1` |
| `btrfs subvolume list` *pathname* | List the subvolumes or snapshots of a subvolume or snapshot that is specified by *pathname*, for example:<br>`btrfs subvolume list /mybtrfs`<br><br>> **✎ Note:**<br>> You can use this command to determine the ID of a subvolume or snapshot. |
| `btrfs subvolume set-default` *ID pathname* | By default, mount the snapshot or subvolume that is specified by its ID instead of the parent subvolume, for example:<br>`btrfs subvolume set-default 4 / mybtrfs` |
| `btrfs subvolume get-default` *pathname* | Displays the ID of the default subvolume that is mounted for the specified subvolume, for example:<br>`btrfs subvolume get-default / mybtrfs` |

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 when the snapshot was taken. By default, the operating system mounts the parent Btrfs volume, which has an ID of `0`, unless you use the `set-default` option to change the default subvolume. If you set a new default subvolume, the system mounts that subvolume going forward. You can override the default setting by specifying any of the `mount` options that are described in the following table.

| Mount Option | Description |
|---|---|
| `subvolid=` *snapshot-ID* | Mount the subvolume or snapshot that is specified by its subvolume ID instead of the default subvolume. |
| `subvol=` *pathname* / *snapshot_path* | Mount the subvolume or snapshot that is 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.

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

When you no longer require a subvolume or snapshot, you can delete it as follows:

```
sudo btrfs subvolume delete subvolume_path
```

Deleting a subvolume deletes all of the subvolumes under 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 information about using the `snapper` command to create and manage Btrfs snapshots, see Automating File System Snapshots With the Snapper Utility.

> **NOT_SUPPORTED:**
>
> Snapshots record the state of the file system at a moment in time. As such, it is not possible to guarantee file system integrity for transactional processes that may have been in operation at the time when a snapshot was taken. While utilities like the `snapper` command may help to capture before and after snapshots for particular operations, such as when using the `dnf` command, these snapshots are still unaware of other processes that may be running on the system at the same time. If you have processes that may have intensive I/O or memory usage, such as database or middleware applications, you should stop these or ensure that all activity is complete before taking a snapshot to help to reduce the likelihood of data integrity or file system corruption issues within the snapshot.

# Creating Swap Files on a Btrfs File System

Swap space is used in Oracle Linux when the amount of physical memory (RAM) is full. In the event that the system needs more memory resources, and the RAM is full, inactive pages in memory are moved to the swap space. Although swap space is helpful for systems with a

small amount of RAM, you should not use swap space as a replacement for more RAM. You can allocate swap space to a dedicated swap partition, which is the recommended method. Alternatively, you can use a swap file; or, you can combine the use of swap partitions and swap files.

Swap files in Btrfs are supported with the following limitations:

- A swap file cannot be on a snapshotted subvolume. Instead, Oracle recommends that you create a new subvolume on which to place the swap file.

- Btrfs does not support swap files on file systems that span multiple devices.

The following are step-by-step instructions for creating a swap file in Btrfs. Prior to creating the new swap file, calculate the size of the swap file in MBs. Then, multiply that number by 1024 to determine the number of blocks the file will require. For example, the block size of a 64 MB swap file is 65536.

1. Create an empty file, for example:

   ```
   sudo dd if=/dev/zero of=/swapfile bs=1024 count=65536
   ```

2. Set up the swap file by running the following command:

   ```
   sudo mkswap /swapfile
   ```

3. Change the permissions on the file so that it is not world readable:

   ```
   sudo chmod 0600 /swapfile
   ```

4. Enable the swap file at boot time by editing the `/etc/fstab` file as the `root` user to include the following entry:

   ```
   /swapfile swap swap defaults 0 0
   ```

5. Regenerate the mount units and register the new configuration in the `/etc/fstab` file:

   ```
   sudo systemctl daemon-reload
   ```

6. Activate the new swap file:

   ```
   sudo swapon /swapfile
   ```

   Running the previous command activates the new swap file immediately.

   Alternatively, you can run the following command to test whether the new swap file was successfully created by inspecting the active swap space:

   ```
   sudo cat /proc/swaps
   ```

   ```
   sudo free -h
   ```

# Creating Backups and Using the Btrfs Send/Receive Feature

> **Note:**
>
> Working with the Btrfs send/receive feature requires that you boot the system by using UEK R7..

The send operation compares two subvolumes and writes a description of how to convert one subvolume, the *parent* subvolume, into the other subvolume, which is 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, for example:

```
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 enables you to save the output to a file. Note that 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 to another subvolume:

```
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] ... subvol
```

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

Use the receive operation to regenerate the sent subvolume at a specified path, for example:

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

# Creating a Reference Backup in Preparation for Creating an Incremental Backup

The following procedure describes how to create a reference backup, which is a prerequisite to setting up an incremental backup and restore process for a subvolume by using the send/receive feature.

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. Ensure that the snapshot has been written to disk by running the `sync` command.

   ```
   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
   ```

   The previous command creates the `/backupvol/backup_0` subvolume.

After creating the reference backup, you can then create incremental backups, as needed. See Creating an Incremental Backup.

## Creating an Incremental Backup

The following instructions describe how to create an incremental backup by using the send/receive feature. Note that before creating an incremental backup, you must first create a reference backup. See Creating a Reference Backup in Preparation for Creating an Incremental Backup.

To create an incremental backup:

1. Create a new snapshot of the subvolume.

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

2. Ensure that the snapshot has been written to disk by running the `sync` command.

   ```
   sudo sync
   ```

3. Send only the differences between the reference backup and the new backup to the backup area, for example:

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

   Running the previous comman creates the `/backupvol/backup_1` subvolume.

# Managing Quotas for Btrfs Subvolumes With Quota Groups

> **✎ Note:**
>
> Be aware that the quota groups feature is available as a Technology Preview *only* in Oracle Linux 9. Working with this feature requires that you boot the system by using UEK R7.

Enable quotas by running following command on a newly created Btrfs file system before any creating any subvolumes:

```
sudo btrfs quota enable volume
```

Assign a quota-group limit to a subvolume by using the following command:

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

The following example shows how you would use this command:

```
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

You can replace devices on a live file system without unmounting the file system or stopping any tasks that are using the file system. If the system crashes or loses power while the replacement is taking place, the operation resumes when the system next mounts the file system.

To replace a device on a mounted Btrfs file system, use the following command:

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

In the previous command, *source_dev* and *target_dev* specify the source device to be replaced (*source device*) and the replacement device (*target device*). The *mountpoint* specifies the file system that is using the source device. The target device must be the same size 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 that is obtained from other devices,. such as another available mirror. The source device is removed from the file system when the operation is complete.

Use the `btrfs replace status` *mountpoint* command to check the progress of the replacement operation and the `btrfs replace cancel` *mountpoint* command to cancel the operation.

# Creating Snapshots of Files

Use the `cp` command with the `--reflink` option 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 enables 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, you would create a snapshot named `bar` of a file named `foo` as follows:

```
cp -reflink foo bar
```

# Automating File System Snapshots With the Snapper Utility

The Snapper utility can be used to automate the management of file system snapshots. The utility can make it easier to create and delete snapshots, while enabling users to compare the differences between snapshots and revert changes at the file level. For information about the Snapper utility, visit the upstream project page at http://snapper.io/.

> **NOT_SUPPORTED:**
>
> Snapshots record the state of the file system at a moment in time. As such, it is not possible to guarantee file system integrity for transactional processes that may have been in operation at the time when a snapshot was taken. While utilities like the `snapper` command may help to capture before and after snapshots for particular operations, such as when using the `dnf` command, these snapshots are still unaware of other processes that may be running on the system at the same time. If you have processes that may have intensive I/O or memory usage, such as database or middleware applications, you should stop these or ensure that all activity is complete before taking a snapshot to help to reduce the likelihood of data integrity or file system corruption issues within the snapshot.

If not already installed, you can install the Snapper utility from the `ol9_UEKR7` yum repository, by running:

```
sudo dnf install -y snapper
```

# Creating a Snapper Configuration for a Subvolume

You can use the `snapper` command 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
```

In the previous command, *config_name* is the name of the configuration and *fs_name* is the path of the mounted Btrfs subvolume. Running the command does the following:

- Adds an entry for *config_name* to the `/etc/sysconfig/snapper` file.

- Creates the configuration file `/etc/snapper/configs/config_name` .

- 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 /
```

Use the `snapper list-configs` command to list all of the existing configurations:

```
sudo snapper list-configs

Config      | Subvolume
------------+----------
home_config | /home
root        | /
```

> **✎ Note:**
>
> The default snapper SELinux policy allows snapper to manage snapshots in the `/`, `/etc`, `/mnt`, `/usr`, `/var` and `HOME_ROOT` (usually `/home`). If you create a new directory, for example `/data` or `/srv`. You may need to set the SELinux file context for that directory so that snapper can create and manage snapshots for that directory. For example to enable snapper to manage snapshots on the `/data` directory, you can run:
>
> ```
> $ sudo semanage fcontext -a -t snapperd_data_t "/data/\.snapshots(/.*)?"
> $ sudo restorecon -R -v /data
> ```

## Creating Different Types of Snapshots

You can create the following three types of snapshots by using the `snapper` command:

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

To create a single snapshot of a subvolume, use the `snapper create` command, for example:

```
sudo snapper -c config_name create --description "description"
```

Single snapshots are useful for periodic backup purposes and can also be used to create a back-up timeline, as described in Automatic Snapper Snapshots. For actions that are likely to result in specific file system modifications that you may need to roll back, you can use pre and post snapshots to capture snapshots of the file system before and after a transaction.

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'
```

Specifying the `-p` option with the `snapper` command displays the number of the snapshot so that you can reference it when creating the post snapshot or when comparing the contents of the `pre` and `post` snapshots.

Note that you can use the `--command` option with the `snapper` command to wrap an operation with `pre` and `post` snapshots. For example:

```
snapper -c root create --command "cd /tmp/build; make install" \
    --description "Installing a home built binary"
```

Pre and post snapshots are frequently used when performing system changes that may be too complex to revert manually, such as when installing or upgrading packages. The DNF snapper plugin that is described in Automatic Snapper Snapshots uses pre and post snapshots in exactly this way and uses the description field to store the DNF transaction that triggered the snapshot.

For example, in the following set of snapshots, you can identify periodic, single snapshots that are triggered as part of a timeline and then a pre and post snapshot that is triggered by the DNF snapper plugin when the vim package is installed.

```
$ sudo snapper -c root list

 # | Type   | Pre # | Date                         | User | Cleanup  |
Description           | Userdata
---+--------+-------+------------------------------+------+----------
+------------------------+---------
0  | single |       |                              | root |          |
current               |
1  | single |       | Wed 25 Nov 2020 07:00:30 EST | root | timeline |
timeline              |
2  | single |       | Wed 25 Nov 2020 08:00:01 EST | root | timeline |
timeline              |
3  | single |       | Wed 25 Nov 2020 09:00:01 EST | root | timeline |
timeline              |
4  | pre    |       | Wed 25 Nov 2020 09:07:21 EST | root | number
| /usr/bin/dnf install vim |
5  | post   |     4 | Wed 25 Nov 2020 09:07:25 EST | root | number
| /usr/bin/dnf install vim |
6  | single |       | Wed 25 Nov 2020 10:00:01 EST | root | timeline |
timeline              |
```

## Automatic Snapper Snapshots

By default, each snapper configuration contains settings for a periodic backup, which is controlled by the TIMELINE_CREATE configuration variable in the /etc/snapper/configs/*config_name* file. Automatic snapshots are triggered by a systemd timer unit that you must enable to allow the timeline to be created:

```
sudo systemctl enable --now snapper-timeline.timer
```

A second systemd timer unit handles the cleanup of stale snapshots so that your snapshots remain manageable. You should enable this unit as well, for example:

```
sudo systemctl enable --now snapper-cleanup.timer
```

When the systemd timer units are enabled, periodic snapshot events trigger automatically for every snapper configuration that has the TIMELINE_CREATE variable enabled. If you wish to disable periodic snapshots for a particular configuration, change the variable value to no in the configuration file.

By default, the snapper timeline configuration keeps 10 hourly, 10 daily, 10 monthly, and 10 yearly snapshots. Snapshots are pruned by the cleanup timer. For busy subvolumes such as the root subvolume, you might want to modify these values to better cater to your requirements. You set these values by changing the following configuration variables:

```
TIMELINE_LIMIT_HOURLY="10"
TIMELINE_LIMIT_DAILY="10"
TIMELINE_LIMIT_WEEKLY="10"
TIMELINE_LIMIT_MONTHLY="10"
TIMELINE_LIMIT_YEARLY="10"
```

The cleanup timer also prunes other snapshots to keep the total number of snapshots reduced. See the `SNAPPER(8)` and `SNAPPER-CONFIGS(5)` manual pages for more information.

You can install the DNF snapper plugin on a system to automatically trigger pre and post snapshots for DNF transactions. This feature can help you roll back changes in cases where system package upgrades cause a failure that you need to debug or to enable you to analyze which files were modified during an installation or upgrade. Note that this plugin requires no user configuration or interaction to work. To install the plugin, use the following command:

```
sudo dnf install python3-dnf-plugin-snapper
```

When installed, a snapshot is triggered for each subsequent DNF transaction. See https://dnf-plugins-extras.readthedocs.io/en/latest/snapper.html for more information.

# Working With Btrfs Snapshots by Using Snapper

To list the snapshots that exist for a snapper configuration or subvolume, run:

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

To display the files and directories that have been added, removed, or modified between two snapshots, use the `status` subcommand and specify the numbers of the two snapshots that you want to compare:

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

To display the differences between the contents of all the files in between two snapshots, use the `diff` subcommand:

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

You can also display the difference in a single file over two snapshots by providing the full path to the file:

```
sudo snapper -c config_name diff N .. N' /path/to/file
```

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'
```

Note that undoing a change does not revert the file system to the previous snapshot but it reverts modifications made to existing files in the snapshot. This means that files created after the snapshot was taken continue to remain after an undochange operation. The `undochange` subcommand does not check data integrity for its changes. You should be careful of using this command without clearly evaluating the implications of the changes that it is likely to make.

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

You can mount any snapshot generated by snapper just as you would work with any other Btrfs snapshot. You may need to correlate the snapshot volume id with the snapper snapshot number to work out which snapshot you should mount or restore. Run the `snapper list` command to identify the number of the snapshot you wish to roll back to. For example, to see all pre and post snapshots, to roll back to a snapshot from before a DNF package update was run:

```
sudo snapper -c root list -t pre-post
```

Running the previous command produces the following output:

```
Pre # | Post # | Pre Date                   | Post Date                  |
Description               | Userdata
------+--------+---------------------------+----------------------------
+-------------------------------+---------
   4  |     5  | Wed 25 Nov 2020 09:07:21 EST | Wed 25 Nov 2020 09:07:25 EST
| /usr/bin/dnf install vim        |
 127  |   128  | Mon 30 Nov 2020 08:25:42 EST | Mon 30 Nov 2020 08:30:57 EST
| /usr/bin/dnf update           |
```

Note that the number of the pre snapshot that we intend to mount is 127 in this case. Use the `btrfs subvolume list` command to obtain the subvolume ID for the snapper snapshot and use this or the path to the snapshot subvolume to mount the file system, for example:

```
sudo btrfs subvolume list /|grep .snapshots.*127
```

The output of the previous command is as follows:

```
ID 521 gen 11533 top level 268 path .snapshots/127/snapshot
```

Then, run the following command:

```
sudo mount -o subvolid=521 /dev/sda2 /mnt
```

You can also use this information to boot into a snapshot of the root file system. See Mounting Alternate Snapshots as the root File System for more infomration.

# Working With a Btrfs root File System

> **⚠ Important:**
>
> In Oracle Linux 9, the Btrfs file system and all of the features that are documented in this chapter are supported in the Unbreakable Enterprise Kernel (UEK) release *only*. Working with Btrfs file system features requires that you boot the system by using UEK R7.

You can create a Btrfs root file system during an installation. To do so, you must boot the system by using UEK R7.

To find out the ID of the parent of the root file system subvolume, use the following command:

```
sudo btrfs subvolume list /
```

Note in the output of the previous command that the top level ID is set with an ID of 5. The top level of the file system is effectively the root of the file system and can be used to access all of the subvolumes within the file system:

```
ID 256 gen 1591 top level 5 path boot
ID 258 gen 1591 top level 5 path root
ID 259 gen 1514 top level 5 path home
ID 262 gen 1514 top level 258 path var/lib/portables"
```

In the previous example, the installation `root` file system subvolume has an ID of 258. The subvolume with ID 258 (`root`) is currently mounted as `/`. The default subvolume (`root`) 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 and indicates the subvolume ID (258):

```
sudo mount|grep 'on / '
```

```
/dev/sda2 on / type btrfs (rw,relatime,seclabel,space_cache,subvolid=258,subvol=/root)
```

Note that the top-level file system in the previous output is not mounted by default. To mount the top-level file system volume, use the following commands:

```
sudo mkdir /mnt
sudo mount -o subvolid=5 /dev/sda2 /mnt
```

If you list the contents of `/mnt`, you can view each of the subvolumes within the file system volume, including the `root` subvolume, for example:

```
ls /mnt
```

Running the previous command displays the following output:

```
boot  home  root
```

Note that the contents of `/` and `/mnt/root` are identical, as s shown in the following example where a file (`foo`) that is created in `/mnt/root` is also visible in `/`:

```
sudo touch /mnt/root/foo
ls /
```

Running the previous command displays the following output:

```
bin   boot  dev  etc  foo  home  instroot  lib  lib64  media  mnt  opt
proc  root  run  sbin  srv  sys    tmp  usr  var
```

Now, list the contents of `/mnt/root`:

```
sudo ls /mnt/root
```

```
bin   boot  dev  etc  foo  home  instroot  lib  lib64  media  mnt  opt
proc  root  run  sbin  srv  sys    tmp  usr  var
```

Remove the `/foo` directory, then list the contents of `/`:

```
sudo rm -f /foo
sudo ls /
```

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

List the contents of `/mnt/root` again:

```
sudo ls /mnt/root

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

# 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/sda2 /mnt
   ```

2. Change directories to the mount point, then take the snapshot. In the following example, the `install` subvolume is currently mounted as the `root` file system:

   ```
   sudo cd /mnt
   sudo mkdir -p /mnt/snapshots
   sudo btrfs subvolume snapshot root snapshots/root_snapshot_1
   ```

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

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

   The list of subvolumes now includes the newly created snapshot.

   ```
   sudo btrfs subvolume list /

   ID 256 gen 1332 top level 5 path boot
   ID 258 gen 1349 top level 5 path root
   ID 259 gen 1309 top level 5 path home
   ID 261 gen 1309 top level 258 path var/lib/portables
   ID 264 gen 1348 top level 5 path snapshots/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:

```
sudo btrfs subvolume set-default 264 /
```

To ensure that the GRUB command line does not overwrite your settings, make the following update:

```
current_grub_kernel=$(sudo grubby --default-kernel);
sudo grubby --remove-args="rootflags=subvol=root" --update-
kernel $current_grub_kernel
```

Reboot the system for the changes to take effect.

Check that the snapshot subvolume ID and subvolume is mounted as the root filesystem:

```
sudo mount|grep 'on / '

/dev/sda2 on / type btrfs (rw,relatime,seclabel,space_cache,subvolid=264,subvol=/
snapshots/root-snapshot1)
```

# Deleting Snapshots of the root File System

> **Note:**
>
> A snapshot cannot be deleted if it is set as the default ID for a subvolume. Deleting a snapshot while it is in use as the root file system may cause system failure and requires a hard physical reset. Before deleting a snapshot that is set as the default subvolume for the root File System, change the default ID and reboot the system, for example:
>
> ```
> sudo btrfs subvolume set-default 258 /
> reboot
> ```

To delete a snapshot, do the following:

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

    ```
    sudo mount -o subvolid=5 /dev/sda2 /mnt
    ```

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

    ```
    $ sudo cd /mnt
    sudo btrfs subvolume delete snapshots/root-snapshot1
    ```

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

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

    The list of subvolumes now does not include `snapshots/root-snapshot1`.

    ```
    sudo btrfs subvolume list /

    ID 256 gen 1332 top level 5 path boot
    ID 258 gen 1349 top level 5 path root
    ID 259 gen 1309 top level 5 path home
    ID 261 gen 1309 top level 258 path var/lib/portables
    ```

# 4
# Managing the Ext File System

The extended file system, or Ext, is the first file system that was written exclusively for the Linux kernel and is in common usage across many Linux distributions. Ext has evolved through several successive updates and is currently available as the Ext4 file system, which is largely backward compatible with previous Ext file system releases but includes many additional features. Key features available in Ext4, include:

- Large file system support: Ext4 can theoretically support volumes with sizes up to 1 EiB and single files with sizes up to 16 TiB.

- Use of extents instead of block mapping: improves large file performance and reduces fragmentation.

- Support for fallocate for persistent pre-allocation of on-disk space for a file: improves performance and helps to ensure contiguous disk allocation for a file.

- Use of allocate-on-flush: helps with performance and reduces fragmentation by delaying disk space allocation until the moment that data is flushed to disk.

- Checksum functionality: ensures data integrity.

For more information, visit https://ext4.wiki.kernel.org/index.php/Main_Page.

This chapter describes tasks for administering the Ext file system in Oracle Linux 9. For an overview of local file system management, see About File System Management in Oracle Linux.

## Converting an Ext File System to a Btrfs File System

You can use the `btrfs-convert` utility to convert an `ext` file system to a Btrfs file system. The utility preserves an image of the original file system in a snapshot named `ext`$N$`_saved`, such as `ext4_saved`. This snapshot enables you to roll back the conversion, even if you have made changes to the `btrfs` file system.

Note that you cannot convert the root file system or a bootable partition, such as `/boot`, to Btrfs.

> **✎ Note:**
>
> The conversion to Btrfs is not supported on the Arm (aarch64) platform. If you are running Oracle Linux 9 on the aarch64 platform, you can migrate data from one file system to another file system.

If you convert the root file system to btrfs, you can use snapshots to roll back changes such as upgrades that you have made to the file system.

# Converting a Non-Root File Ext File System to a Btrfs File System

> **⚠ Caution:**
>
> Before performing a file system conversion, back up the file system from which you can restore its state.

To convert an `ext` 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`.

   ```
   /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 Ext File System to a Higher Version

> **⚠ Caution:**
>
> Before performing a file system conversion, make a backup of the file system from which you can restore its state.

1. Unmount the current file system:

   ```
   sudo umount filesystem
   ```

2. Check the system by running the `fsck` command appropriate to the file system. For example, if the file system is `ext2`, you would use `fsck.ext2` as follows:

   ```
   sudo fsck.ext2 -f device
   ```

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

   ```
   sudo tune2fs -j device
   ```

This command adds an `ext3` journal inode to the file system.

4. Check the file system by using the `fsck` command appropriate to the higher version. For example, if the new file system is `ext3`, you would use `fsck.ext3` as follows:

```
sudo fsck.ext3 -f device
```

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

6. Remount the file system.

```
sudo mount filesystem
```

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

# Converting a Root Ext File System to a Higher Version

> ⚠️ **Caution:**
>
> Before performing a root file system conversion, make a full system backup from which you can restore its state.

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

```
sudo tune2fs -j device
```

2. Determine which device is currently mounted as the root file system by running the `mount` command.

   For example, the output of the following command shows that 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 the Oracle Software Delivery Cloud at https://edelivery.oracle.com/linux. For convenience, these images are also available from the Oracle Linux yum server at https://yum.oracle.com/oracle-linux-isos.html.

5. From the installation menu, select **Rescue Installed System**.

   a. Select a language and keyboard, when prompted.

   b. Select **Local CD/DVD** as the installation media.

   c. Select **No** to bypass starting the network interface.

   d. Select **Skip** to bypass selecting a rescue environment.

6. Select **Start shell** to obtain a `bash` shell promptat 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. If the volume group is `vg_host01`, type:

```
sudo lvchange -ay vg_host01
```

8.  Check the file system by using the `fsck` command appropriate to the new file system. If the high version is `ext3`, you would type the command as follows:

    ```
    sudo fsck.ext3 -f device
    ```

    In the previous command, *device* is the root file system device, `/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 it.

    ```
    sudo mkdir /mnt1
    sudo mount -t ext3 device /mnt1
    ```

10. Edit the `/mnt1/etc/fstab` file to change the root file system type to `ext3`.

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

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

    ```
    sudo touch /mnt1/.autorelabel
    ```

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

    > ⚠ **Caution:**
    >
    > If you do not create the `.autorelabel` file, you might not be able to boot the system successfully. In the file is not created and the reboot fails, you can either disable SELinux temporarily by specifying the`selinux=0` value in the kernel boot parameters; or, you can run SELinux in permissive mode by specifying `enforcing=0`.

12. Unmount the converted root file system.

    ```
    sudo umount /mnt1
    ```

13. Remove the installation media, then reboot the system.

# Checking and Repairing an Ext File System

You use the `fsck` utility to check and repair 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 being performed. You might want to run `fsck` manually if a file system has not been checked for several months.

The following procedure describes how to check and repair an Ext file system.

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

To check and repair an Ext file system:

1. Unmount the file system:

   ```
   sudo umount filesystem
   ```

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

   ```
   sudo fsck [-y] file-system
   ```

   In the previous example, *file-system* specifies a device name, a mount point, or a label or UUID specifier, for example:

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

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

For the `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 for Ext File Systems

> **Note:**
>
> The following procedure applies to Ext file systems *only*. XFS file systems, which are the default file system type in Oracle Linux 9, detect errors automatically and do not require periodic file system checks at boot time.

To change the number of mounts before the system automatically checks the file system for consistency, use the following command syntax:

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

In the previous command, *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 *mount_count* value for each file system reduces the probability that the system checks all of the file systems at the same time.

To specify the maximum interval between file system checks, use the following command syntax:

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

In the previous command, *unit* can be d, w, or m, for days, weeks, or months, respectively.

The default unit is d (for days). An *interval* of 0 disables checking, 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.

# 5
# Managing 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 then 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, with each server consisting of multiple FC-connected disk arrays.

XFS is supported for use with the `root` (/) or `boot` file systems on Oracle Linux 9.

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:

- 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.

- 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.

- 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.

- Supports the reflink and deduplication features, which provides the following benefits:
  - Each copy can have different file metadata (permissions, and so on) because each copy has its own distinct inode. Only the data extents are shared.
  - The file system ensures that any write causes a copy-on-write, without applications having to do anything special.
  - Changing one extent continues to allow all of the other extents to remain shared. In this way, space is saved on a per-extent basis. Note, however, that a change to a hard-linked file does require a new copy of the entire file.

- 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 an XFS File System.

- 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.

For more information about XFS, see https://xfs.wiki.kernel.org/.

For an overview of local file system management, see About File System Management in Oracle Linux.

# Installing 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 `ol9_x86_64_baseos_latest` channel.

2. On your system, install the `xfsprogs` and `xfsdump` packages:

   ```
   sudo dnf install xfsprogs xfsdump
   ```

3. If you require the XFS development and QA packages, additionally subscribe your system to the `ol9_x86_64_optional` channel and install them:

   ```
   sudo dnf 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
```

Running the previous command produces the following output:

```
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
```

The following examples shows how you would create an XFS file system with a stripe-unit size of 32 KB and 6 units per stripe. To do so, you would specify the `su` and `sw` arguments to the `-d` option:

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

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

# Enabling External XFS Journals

The default location for an XFS journal is on the same block device as the data. Because 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.

# Disabling XFS 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. However, if required, you can disable them by specifying the `-l lazy-count=0` option to the `mkfs.xfs` command.

# Modifying an XFS File System

> **Note:**
>
> It is not possible to 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, use the following command:

```
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, use the following command:

```
sudo xfs_admin -u /dev/sdb
sudo xfs_admin -U generate /dev/sdb
```

To clear the UUID altogether, use the following command:

```
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 reenable lazy counters, use the following commands:

```
sudo xfs_admin -c 0 /dev/sdb
sudo xfs_admin -c 1 /dev/sdb
```

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. Also, no command currently exists 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, you would increase the size of `/myxfs1` to 4 TB, assuming a block size of 4 KB, as follows:

```
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.

# Creating an XFS File System With the Reflink Feature

In Oracle Linux 9, you can create XFS file systems by using the Reflink feature. This feature reduces disk space consumption and accelerates the copying of files. For example, if you use a reflink-aware tool to copy a directory tree, the files in the copy will share disk space with the original. It takes much less time make a reflink copy of the directory tree than to create a regular copy of a file system. In addition, no additional storage will be used.

Note that the reflink feature is enabled by default in Oracle Linux 9 when formatting by using the `mkfs.xfs` command.

To begin working with XFS's reflink support, start by formatting a file system:

```
sudo mkfs.xfs /dev/sda1

meta-data=/dev/sda1              isize=512    agcount=4, agsize=6553600 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=1, sparse=1, rmapbt=0
         =                       reflink=1
data     =                       bsize=4096   blocks=26214400, imaxpct=25
         =                       sunit=0      swidth=0 blks
```

```
naming   =version 2              bsize=4096   ascii-ci=0, ftype=1
log      =internal log           bsize=4096   blocks=12800, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
```

> **Note:**
>
> If you do not see the exact phrase "reflink=1" in the `mkfs` command output, your system is too old to support the reflink feature on XFS.

Next, mount the file system:

```
sudo mount /dev/sda1 /storage
```

The files system is now ready to absorb new files.

For a detailed demonstration on using the reflink feature, see https://blogs.oracle.com/linux/xfs-data-block-sharing-reflink.

# 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 an XFS file system, use the `-f` option with the `xfs_freeze` command:

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

To unfreeze an XFS file system, use the `-u` option with the `xfs_freeze` command:

```
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.

# Managing Quotas on an XFS File System

Use the `xfs_quota` tool to manage quotas on an XFS file system. This tool is catered specifically to quota implementation in XFS. While other quota tools, such as `edquota` may allow you to edit XFS quotas, `xfs_quota` is preferred.

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

**Display Block Usage Information**

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

```
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
```

**Set Quota Limits**

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
```

Note that this command requires that you have 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`:

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

**Set Project Quota Limits**

The instructions that follow here assume that you have already mounted the file system using the `pquota` option and have created a project ID in `/etc/projects`. See Setting Project Quotas.

1. 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, you would define a managed tree in the `/myxfs` file system for the project `testproj`, which corresponds to the directory hierarchy `/myxfs/testdir`, as follows:

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

2. 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`, you would use the following command:

```
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 an XFS File System

The `xfsdump` package contains the `xfsdump` and `xfsrestore` utilities. The `xfsdump` command 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 by using the `xfsdump` command are portable between systems with different endian architectures. The `xfsrestore` command 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. Additionally, you can write multiple backups to the same medium. Note that you can write only a single backup to a file. The command does not overwrite existing XFS backups that are found on physical media. If you need to overwrite any existing backups, you must use the appropriate command to erase a physical medium.

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 that are relative to an existing backup by using the same command, for example:

```
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, for example:

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

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

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, you would use the following command:

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

Specify the `-r` option to cumulatively recover all of the data from a level 0 backup, as well as higher-level backups that are based on that backup:

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

This command searches for backups in the archive, based on the level 0 backup, and then prompts you to choose whether you want to restore each backup, in turn. After restoring the selected backup, the command exits. Note that 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 backup that you require in order to restore the file system data.

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

As shown in the following example, you can recover a selected file or subdirectory contents from the backup medium. Running the command recovers the contents of `/myxfs/profile/examples` to `/tmp/profile/examples`, from the backup with the 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, for example:

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

The previous form of the command enables 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 the `xfsdump` and `xfsrestore` command by using the `-J` option to suppress the usual dump inventory housekeeping that the commands perform, for example:

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

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

# 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 indicates 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 the `xfsdump` command to attempt a back up of the existing file system data. However, note that the command might fail if the file system's metadata has become corrupted.

You can use the `xfs_repair` command to attempt to repair an XFS file system that is 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, you usually do not need to use the follwoing 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 with debugging and repairing XFS file systems by using the `xfs_db` command, 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 the `xfs_repair` command is the only viable option, unless you are experienced in using the `xfs_db` command.

`xfs_db` provides an internal command set that allows you to debug and repair an XFS file system manually. The commands enable you to perform scans on the file system, as well as 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 run the `help` command within `xfs_db`.

# 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, use the following command to specify the name of the file as the argument to `xfs_fsr`:

```
sudo xfs_fsr pathname
```

Running the `xfs_fsr` command without any options defragments all of the currently mounted and 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, and attempts to defragment the top ten percent of files with the greatest number of extents. After two hours, the command records its progress in the `/var/tmp/.fsrlast_xfs` file. If you run the command again, the process is resumed from that point.

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