Oracle Linux 9

Managing Local File Systems

F52807-06
July 2025

Oracle Linux 9 Managing Local File Systems,

F52807-06

Contents

Preface

Oracle Linux 9: Performing File System Administration provides information about managing file systems and storage devices on Oracle Linux 9 systems.

Documentation License

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/.

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 Systems

You can format block devices and logical volumes with any of several different file systems on Oracle Linux. File systems include different features that can make them ideal for particular environments or workloads.

The most commonly used file systems on Oracle Linux include the following:

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

> **Note:**
>
> In Oracle Linux, the Btrfs file system type is supported on Unbreakable Enterprise Kernel (UEK) releases *only*. If you boot the system using RHCK, any Btrfs file systems are inaccessible.

See also Oracle Linux 9: Managing the Btrfs File System

Ext4
A version of the extended file system. Ext4 includes the same features that are available in Ext3, but includes the addition of the following features: *extents* or contiguous physical blocks, preallocation, delayed allocation, speedier file system checking, more robust journaling, and several other enhancements.
See also Oracle Linux 9: Managing the Ext File System

XFS
A high-performance, journaling file system that provides high scalability for I/O threads, file system bandwidth, file size, and file system size, even for file systems that span many storage devices. XFS is the default file system selection when installing Oracle Linux.
See also Oracle Linux 9: Managing the XFS File System

File system limitations are affected by kernel versions and features, and also by the architecture of the system on which Oracle Linux is installed. You can review the limits on the different core file systems described in Oracle Linux: Limits.

To list recognized local file system types on a system, use the following command:

```
ls -l /sbin/mkfs.*
```

Note that the following output might differ, depending on the packages that are installed on the system, but output might appear as follows:

```
/sbin/mkfs.btrfs
/sbin/mkfs.cramfs
/sbin/mkfs.exfat
```

```
/sbin/mkfs.ext2
/sbin/mkfs.ext3
/sbin/mkfs.ext4
/sbin/mkfs.fat
/sbin/mkfs.minix
/sbin/mkfs.msdos
/sbin/mkfs.ocfs2
/sbin/mkfs.udf
/sbin/mkfs.vfat
/sbin/mkfs.xfs
/sbin/mkfs.xmem
```

These executable files are used to make the file system type that's specified by their extension. See Formatting a File System for more information.

In addition to local file systems, Oracle Linux can use shared file systems that are available across a network. Examples include:

NFS
NFS is a distributed file system protocol that lets systems share files over a network. NFS is in common usage in Linux and UNIX environments. An NFS server exports directories or files to clients, which can then mount these resources as if they were local. NFS is typically used when setting up a shared storage solution within an organization where client systems use Linux or UNIX platforms. NFS exposes an existing locally formatted file system over the network. See Oracle Linux 9: Managing the Network File System for more information.

OCFS2
The Oracle Cluster File System version 2 is a cluster file system designed for high-performance computing environments. It provides a scalable and fault-tolerant way to manage shared storage among nodes in a cluster. Unlike traditional file systems, OCFS2 lets servers concurrently read and write to the same block device, making it ideal for applications requiring simultaneous access to shared data. Use cases include large-scale databases, virtualization platforms, and cloud infrastructure where several instances need concurrent access to shared storage. Block devices are formatted with OCFS2 across the cluster. See Oracle Linux 9: Managing the Oracle Cluster File System Version 2 for more information.

CIFS/SMB
CIFS (Common Internet File System) and SMB (Server Message Block) are network file-sharing protocols developed by Microsoft and designed to let Windows-based systems share files with other devices on a network, including Linux systems running Samba software. SMBv3 includes significant improvements and security enhancements over CIFS and is the preferred protocol choice. See Oracle Linux 9: Managing Samba for more information.

2
Formatting a File System

The `mkfs.`*fstype* command is used to create a file system on a specified device or file image.

The `mkfs.`*fstype* command creates a Linux file system on a device or file image.

```
sudo mkfs.fstype [-L label][options] device
```

Many file system types are available to provide variations on the command name, including the most commonly used file systems on Oracle Linux:

- `mkfs.btrfs` - Format a file system using Btrfs. Requires UEK to be installed on the system. Note that if UEK is installed on the system, but the system is running RHCK, you can create a Btrfs file system, but you can't mount it until the system is booted into UEK. See also Oracle Linux 9: Managing the Btrfs File System
- `mkfs.xfs` - Format a file system using XFS. See also Oracle Linux 9: Managing the XFS File System
- `mkfs.ext4` - Format a file system using Ext4. See also Oracle Linux 9: Managing the Ext File System

The *device* argument is either the device name or a file image that can contain the file system.

For example:

```
sudo mkfs.xfs /dev/sdb1
```

> **Note:**
>
> The `mkfs -t` *fstype* command is a wrapper command for the `mkfs.fstype` commands and is deprecated. It defaults to use `mkfs.ext2` if no file system type is specified.

You can use the `-L` option to specify a file system *label* for most file system types.

While you don't need to specify options for the `mkfs.fstype` commands to work for most common use cases because the default values suffice, you might use specific options when working with particular hardware or when you intend to enable specific features within the file system. Many options are available for each file system and options are specific to each file system type. For more information, see the `mkfs.`*fstype*`(8)` manual pages.

3
Managing File System Mounts

To access a file system's contents, you need to attach its block device to a mount point in the root file system's directory hierarchy. Any directory can function as a mount point.

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

Viewing File System Mount Information

The `findmnt` command provides a single tool that can be used to provide tabulated and structured information about all file systems that are mounted on a system.

The following examples show how to use the `findmnt` command. You can find out more information in the `findmnt(8)` manual page.

- List all mounted file systems.

  ```
  findmnt
  ```

  By default, `findmnt` shows all the mounted file systems that are visible to the kernel in `/proc/self/mountinfo`. You can run `findmnt -m` to search the system mtab, or `findmnt -s` to limit the search to entries in the system fstab.

- Provide disk space usage information for file system mounts.

  ```
  findmnt -D
  ```

  Output is formatted similarly to the output that you can get from running `df -H` and excludes pseudo file systems such as procfs and sysfs.

- Limit file system listings by file system type.

  Limit information to file systems matching particular file system types by using the `--types` option. For example, you can run:

  ```
  findmnt --types xfs,ext4,btrfs,vfat
  ```

- Return the file system information for a particular mount point.

  ```
  findmnt /home
  ```

- Show selected output fields.

  Use the `-o` option to control the different output fields that are shown in a listing. For example, you can run:

  ```
  findmnt -o TARGET,UUID,FSTYPE,OPTIONS,USE% /home
  ```

Using the `mount` Command

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, any of the following mount commands are appropriate:

```
sudo mount /dev/sdb1 /mnt
```

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

```
sudo mount LABEL="Projects" /mnt
```

The *mount_point* must exist as a directory within an existing file system. Typically the mount point is an empty directory, but if it isn't empty the contents of the directory aren't accessible while a device is mounted onto it.

Options

The following options are commonly used when mounting file systems. For more comprehensive coverage of available options, see the `mount(8)` manual page.

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

`exec`
Allows the execution of any binary files in the file system.

`loop`
Uses a loop device (`/dev/loop*`) to mount a file that contains a file system image. See `losetup(8)` manual page. See alsoUsing the `loop` Device to Mount a File System Image

> **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. Or, 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 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's already mounted. You would typically 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.

Example 3-1 Mount a Device With Read-Only Access

Mount the `/dev/sdd1` device onto the mount point `/test` with read-only access and grant only the `root` user to mount or unmount the file system:

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

Example 3-2 Mount an ISO Image

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

```
sudo mount -o ro,loop ./Oracle_Linux.iso /media/cdrom
```

Example 3-3 Remount a File System With Read-Write Access and Prevent Running of Binary Files

Remount the `/test` file system with both read and write access, and prohibit the running of any binary files that are in the file system:

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

Using Bind Mounts

A bind mount can be used to remount a directory or file at another location in the file system hierarchy.

Bind mounts are commonly used to expose data from an underlying file system to chroot or container environments. Bind mounts are also useful when working with applications that might expect data to appear in alternative locations on a file system.

Each directory hierarchy acts as a mirror of the other. The same files are accessible in either location.

For a standard bind mount, any submounts aren't replicated. You can also use a recursive bind mount to replicate any submounts within the new mount location.

The mount command also includes options to prevent a bind on a mount point, to share mount actions for submounts between mirrors.

The mirrors that are created by using bind mounts don't provide data redundancy.

- To perform a standard bind mount of a directory or file in the file system to another directory or file, use the `mount -B` or `mount --bind` command:

  ```
  sudo mount --bind source target
  ```

  Replace *source* with the path to the directory or file you want to mount, and *target* with the path where you want to mount it. For example:

  ```
  sudo mount --bind /mnt/data /home/user/data
  ```

  This command binds the `/mnt/data` directory to the `/home/user/data` directory.

  To mount a file over another file, you can use the same command, but use a file as the source and a file in a different location as the target. For example:

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

  The `/etc/hosts` file is mounted over the `/mnt/foo` file. The existing file that acts as a mount point isn't accessible until you unmount the overlying file.
- To perform a recursive bind mount, use the `mount --rbind` or `mount -R` command.

  ```
  sudo mount --rbind source target
  ```

  Replace *source* with the path to the directory or file you want to mount, and *target* with the path where you want to mount it. For example:

  ```
  sudo mount --R /mnt/data /home/user/data
  ```

  In this case if `/mnt/data` contains any submounts, the submounts are also replicated within `/home/user/data`.
- To change the mount options on a bind mount, you must remount the bind mount an specify the required options.

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

  ```
  sudo mount -o remount,ro /home/user/data
  ```

- Control how mount and unmount actions are propagated to submounts for any mount point.

  You can mark the submounts in a mount point as being shared, private, or secondary. You can specify the following options:

`mount --make-shared` *mount_point*

Any mounts or unmounts under 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 under the specified mount point don't 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 under the specified mount point don't propagate to other mirrors, but this mount hierarchy does reflect mounts or unmount changes that you make to other mirrors.

For example, to propagate mount actions within a bind mount, run:

```
sudo mount --bind --make-shared /mnt/data /home/user/data
```

If a new mount is performed within the `/mnt/data` subtree, the mount is also propagated to the same location within the `/home/user/data` subtree.

- 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
  ```

Using the `loop` Device to Mount a File System Image

You can create a file system image within an existing file system and mount the file system image by using a `loop` device.

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

```
sudo mount -t iso9660 -o loop /tmp/OracleLinux.iso /ISO
```

The mount command can detect the file system type and understands that when you're mounting a file image to a mount point, a loop device is intended. Therefore, you can also run the mount command without specifying the `-t` and `-o` options:

```
sudo mount /tmp/OracleLinux.iso /ISO
```

You can also create a file system image with an alternative file system within an existing file system. The following procedure describes how to create an empty file, format it with a file system and then mount it using a loop device.

1. Create an empty file of the required size.

   ```
   sudo dd if=/dev/zero of=/fsfile bs=1M count=4096
   ```

Or alternatively, run:

```
sudo fallocate -l 4G /fsfile
```

2. Create a file system on the image file.

   For example, to format the file with the Ext4 file system, you can run:

   ```
   sudo mkfs.ext4 -F /fsfile
   ```

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

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

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

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

Moving a File System to a New Mount Point

You can move a file system mounted at a mount point to a new mount point by using the `mount --move` command.

The `mount --move` command. can move a file system from one mount point to another without unmounting it.

Moving the file system can be useful if you need to switch mount points without interrupting processes that are running and accessing files. Moves are more commonly performed in early stage boot processes, or in containerized. environments. The move operation removes the original mount after the file system is moved to the new target. If a process has an open file based on the original mount point, the process is unaffected because the file descriptor points to the inode and not the path, and the file system continues to be available because it's not unmounted during the move operation.

- To move a mounted file system, directory hierarchy, or file to a new mount point, use the `mount --move` or `mount -M` command.

  For example, run:

  ```
  sudo mount -M /mnt/users /home
  ```

  The source mount that you're moving must be a mount point and can't be a directory within a mount. Moving a mount under a shared mount is invalid and errors out.

Unmounting a File System

You can unmount a file system that's no longer in use on the system by using the `umount` command.

When you unmount a file system, any pending writes are flushed to disk to ensure data integrity, the file system releases any resources in use, such as memory buffers and cache entries, and file system metadata is updated to reflect that the file system is no longer in use. The mount point where the file system was mounted is removed from the system namespace and the device is freed to be removed from the system.

A file system can't be unmounted if it's busy with open files, or if some process has a directory on the file system open.

Some options are available for the `umount` command to override default behavior. For example, you can perform a lazy unmount, which doesn't wait until all pending writes are flushed to disk, by using the `-l` option. This option can be useful when working with network mounted file systems that might have hung because of a network outage, and which might cause a significant delay when rebooting a system.

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

- To unmount a file system, use the `umount` command.

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

  Or, you can specify the block device if it's mounted on only one mount point.

Managing the File System Mount Table

File systems can be configured in the file system mount table, or fstab, so that they can be automatically mounted at boot, or to make it easier to mount file systems at commonly used mount points with standard options applied at mount time.

The file system mount table is contained in the `/etc/fstab` file, which provides all the information that the `mount` command requires to mount block devices or implement binding of mounts. The following are typical entries from this file:

```
/dev/mapper/ocivolume-root                /          xfs     defaults
0 0
UUID=5097b6ba-ed0e-418a-9c2c-fb25d577991f /boot      xfs     defaults
0 0
UUID=349C-BCCC                            /boot/efi  vfat
defaults,uid=0,gid=0,umask=077,shortname=winnt 0 2
/dev/mapper/ocivolume-oled                /var/oled  xfs     defaults
0 0
tmpfs                                     /dev/shm   tmpfs
defaults,nodev,nosuid,noexec      0 0
/.swapfile                                none       swap    sw   0 0
```

The descriptions of each field in the 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's 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 the mount point for a file system. The mount point must be a path to either a file or a directory. Note that a swap partition isn't mounted to a traditional mount point.

- The third field is the file system type, such as `xfs` or `swap`.
- The fourth field specifies any mount options.
- The fifth column is a legacy entry that was used to control the now deprecated `dump` command. This field must be set to `0`. The `dump` command is removed in this Oracle Linux release.
- The sixth column identifies the order by which the `fsck` command performs a file system check at boot time. The root file system has the value `1`, while other file systems have `2`. A value of `0` skips checking, as is appropriate for swap, for file systems that aren't 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.

The fstab is maintained as a simplified interface to systemd mount targets. At boot time, systemd reads the fstab and creates ephemeral mount target units for each of the entries.

Systemd mount units can provide much tighter control over when file systems are mounted. See Using systemd `mount` Targets for more information.

Editing `/etc/fstab` For Persistent Mounts

Edit the `/etc/fstab` file to add a new file system mount.

The `/etc/fstab` file contains information about the file systems that are mounted automatically when the system boots. The file maintains compatibility across Oracle Linux releases and continues to be used by many system administrators instead of configuring individual systemd mount targets.

See Managing the File System Mount Table for more information.

1. Open the `/etc/fstab` file in a text editor as the root user.
2. Add a new line to the end of the file, to describe the system mount, or edit any existing lines to match any new requirements.

   Line entries match the following format:

   ```
   devicemount_pointfile_system_typeoptionsdumpfsck
   ```

   - *device*: the device name or the UUID of the device that contains the file system.
   - *mount_point*: the existing directory where the file system can be mounted.
   - *file_system_type*: the type of file system.
   - *options*: any file system specific options for the mount.
   - *dump*: this option is a legacy option and can be set to 0.
   - *fsck*: the order in which the file system is checked by the fsck utility, can be set to 0 or a positive integer.

   For example, an entry might look as follows:

   ```
   UUID=79428c8f-d67a-49ec-ba20-8a86337f4447 /mnt/data  xfs  defaults  0 0
   ```

3. Save and close the file.
4. Remount all file systems in `/etc/fstab` to apply any changes.

   ```
   sudo mount -a
   ```

   Note that the `mount -a` command mounts all file systems in `/etc/fstab` that don't have the `noauto` option set. If you added a mount that included the `noauto` option you must explicitly mount it for it to become active.

Managing Systemd Mounts

Systemd provides a powerful way to manage file system mounts by using mount units.

All mount points defined in the file system mount table, or fstab, are processed by systemd at boot and are treated as ephemeral systemd mount targets.

However, by defining a systemd mount target explicitly, you can take advantage of many of systemd's broader capabilities. For example, you can set dependencies so that a mount is only performed after a certain dependency, such as network availability, is reached.

Systemd mount targets provide more configurable options, can provide better logging around system mounts, and can improve error handling and troubleshooting.

See the `systemd.mount(5)` manual page and Oracle Linux 9: Managing the System With systemd for more information.

You can view all system mounts, including the ephemeral mount targets loaded from the fstab by using the `systemctl list-units -t mount` command:

```
sudo systemctl list-units -t mount
```

Using systemd `mount` Targets

You can optionally create systemd `mount` targets to manage file system mounts that take advantage of other systemd capabilities.

> **Tip:**
>
> If you define systemd mount targets, it can be useful to include a comment in `/etc/fstab` so that system administrators are more aware that a newer convention is in use on the system.

1. Create a mount unit file in the `/etc/systemd/system` directory.

   Use the `systemctl edit` command to create the new systemd unit file.

   ```
   sudo systemctl edit --force --full data.mount
   ```

   The command opens the default text file editor to create the target unit, `/etc/systemd/system/data.mount`.

Note that you use the `--force` option to create a unit that doesn't exist yet. The target unit file suffix must be `.mount` to create a mount unit.

a. To define a mount point you must enter the appropriate entry for a systemd mount unit.

   For example, you could enter something similar to:

   ```
   [Unit]
   Description=/mnt/data Mount Point

   [Mount]
   What=UUID="a80f8f10-75b6-45de-b63e-64b8b6a3a94b"
   Where=/mnt/data
   Type=xfs
   Options=defaults

   [Install]
   WantedBy=multi-user.target
   ```

   Replace the values:

   - *What*: the device or file system to mount, preferably using the block device UUID for stability across system reboots.
   - *Where*: the mount point where the file system is mounted. The directory location must already exist on the underlying file system.
   - *Type*: the file system type.
   - *Options*: the mount options.

b. Save and exit the text editor.

   If the unit configuration is empty, the mount unit file isn't created.

2. Reload the systemd daemon:

   Reload the systemd daemon to pick up the new mount unit:

   ```
   sudo systemctl daemon-reload
   ```

3. Enable and start the mount unit.

   Enabling the unit ensures that it's remounted at boot.

   ```
   sudo systemctl enable --now data.mount
   ```

4. Optionally, include a comment in `/etc/fstab` to help other system administrators.

   Add a line to the end of `/etc/fstab` by running a command similar to the following:

   ```
   echo '# data.mount systemd target added for /mnt/data' | sudo tee -a /etc/
   fstab
   ```

Managing On-Demand Mounts Using `autofs`

On-demand mounts are used to automatically mount file systems when they're accessed, rather than maintaining connections for those mounts all the time. When a file system becomes inactive for a certain period, the file system is unmounted.

Using on-demand mounts frees up system resources and improves system performance. Furthermore, when working with network based file systems, such as NFS, and Samba, using on-demand mounts can reduce the likelihood of stale mounts.

On-demand mounts are handled by the automounter, which consists of two components: the `autofs` kernel module and the `automount` user-space daemon. Configuration is handled in a set of configuration files including `/etc/autofs.conf` and `/etc/auto.master`.

See also the `autofs(5)`, `auto.master(5)`, and `autofs.conf(5)` manual pages.

On-demand mounts are commonly used with network based file systems such as NFS. For more information about NFS administration, see Oracle Linux 9: Managing the Network File System.

Installing and Enabling autofs

Install the `autofs` package to use on-demand mounting of file systems. Enable and start the `autofs` systemd service after configuration to make it active and persistent across system reboots.

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 the file systems.

   See Configuring autofs for more information.

3. Enable and start the `autofs` service:

   ```
   sudo systemctl enable --now autofs
   ```

Configuring autofs

Autofs uses a combination of configuration files to manage on-demand mounting of file systems.

Master Map File (/etc/auto.master)

The master map file, at `/etc/auto.master`, serves is the primary configuration file for autofs mounts. It contains a list of entries that specify the relationship between mount points and their corresponding map files.

- **Mount point**: The directory where the file system can be mounted.
- **Map file**: The location of the map file that defines the file system details.
- **Options**: Optional parameters that control the autofs behavior, such as the timeout value.

```
/mnt/shared /etc/auto.shared --timeout=300
```

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
```

The previous example shows the following types of map entries:

- `/-`: direct map entry. Direct map entries always specify `/-` as the mount point.
- `/misc`: indirect map entry.
- `/net`: host map entry. Host maps always specify the keyword `-hosts` instead of a map file.

Map Files

Map files contain the actual file system definitions, specifying the type of file system, mount options, and other relevant details. You can define two types of mappings:

- **Direct map files**: 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:

  ```
  /mnt/data    -fstype=nfs,ro,soft      host01:/export/data
  ```

  This entry is a directive to do the following:

  - Mount the file system `/export/data` that's exported by `host01` by specifying the `ro` and `soft` options.
  - Create the `/mnt/data` mount point if it doesn't already exist. If the mount point exists , the mounted file system hides any existing files that it contains.
- **Indirect map files**: An indirect map contains definitions of directories or *keys* that are automounted relative to the mount point (`/misc`) that's 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=xfs                           :/dev/hda1
  fenetres   -fstype=smb3,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 don't already exist, and then removes them when it unmounts the file system. So if a program accesses `/misc/xyz/`, the mount point is created and mounted on demand, so that you can access the file system immediately.

Other Configuration Files

Other configuration files are available to customize autofs behavior, including:

- `/etc/autofs.conf`: The global configuration file for autofs, used to control functionality such as logging, debugging, and the location of the master map file.

- `/etc/auto.net`: A useful script that's used by autofs to automatically mount NFS shares for resolvable hosts on the network that have NFS shares exposed. If a host map entry exists, and a command references an NFS server that's relative to the mount point (`/net`) by name, the automounter mounts all the directories that the server exports within 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` under the `/net/host03` directory. By default, the automounter uses the `nosuid,nodev,intr` mount options. These options are configurable in `/etc/auto.master`.

  The name of the NFS server must be resolvable to an IP address in DNS or must be present in the `/etc/hosts` file. You can equally reference hosts directly by their IP address. For example, `ls /net/192.168.0.3/` mounts all NFS exports on the server at the IP address 192.168.0.3.

- `/etc/auto.smb`: A useful script that's used by autofs to automatically mount CIFS shares for resolvable Windows or Samba file servers on the network. autofs automatically scans the host for CIFS file shares and exposes these within the `/cifs/host/` directory. To use this, you must install the `samba-client` package and add an entry to `/etc/auto.master`:

  ```
  /cifs  /etc/auto.smb --timeout=300
  ```

4

Discarding Unused Blocks on a File System

Some file systems, such as XFS, Btrfs, and Ext4, provide a discard feature that discards unused blocks that are no longer in use by the mounted file system. This file system optimization can improve performance by reclaiming space more efficiently and reducing fragmentation.

The discard feature can improve garbage collection for the file system. When a file is deleted, the file system marks the corresponding blocks as free. However, the actual data remains on the disk until it's overwritten. Discarding unused blocks ensures that the garbage collector can reclaim those blocks more efficiently.

Solid-State Drives (SSDs) have limited write cycles before they start to degrade. Discarding unused blocks also helps reduce the number of writes to the SSD, extending its lifespan, and reduces wear on the disk.

In thin provisioning environments, the discard feature helps optimize storage allocation by releasing unused blocks back to the pool, reducing waste, and improving resource usage. Discarding unused blocks also helps prevent unnecessary growth of the file system and reduces the risk of running out of space.

File systems that include the discard feature also include a `discard` mount option, to perform online discard operations that discard blocks as they change from the `used` to the `free` state. You can also use the `fstrim` command to manually discard unused blocks on a file system that includes the feature, or set a systemd timer unit to perform the discard operation at frequent intervals.

Using `fstrim` to Discard Unused Blocks

The `fstrim` command is used to discard unused blocks on a file system, which can improve performance and reduce wear on storage devices based on flash technology, such as solid-state drives (SSDs) and NVMe.

The `fstrim` command only works on file systems and devices that can support the feature. If you run the command for a device that doesn't include the discard feature, an error message is returned.

- To perform a discard on a selected file system, provide the `fstrim` command with the path to the mount point of the file system.

  For example run:

  ```
  sudo fstrim /mnt
  ```

- You can perform a discard on all mounted file systems by using the `--all` option:

  ```
  sudo fstrim --all
  ```

Enabling Automatic Block Discard Using the `fstrim` Systemd Timer

Configure the `fstrim` systemd timer unit to periodically trim unused blocks on a file system.

Enabling the `fstrim` timer unit automate the discard process, ensuring that the file system remains optimized over time. The default timer unit is configured to run the `fstrim` operation weekly, but you can change the frequency by creating a systemd override drop-in unit and setting the `OnCalendar` directive.

1. To enable the `fstrim` timer unit, run:

   ```
   sudo systemctl enable --now fstrim.timer
   ```

2. To verify that the `fstrim` timer unit is enabled and active, run:

   ```
   sudo systemctl status fstrim.timer
   ```

5
Configuring 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 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`, `OCFS2`, `ext3`, `ext4`, and `XFS` file systems, including 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 doesn't have an access ACL.

Enabling ACL Support

1. Ensure that the `acl` package is installed.

   ```
   sudo dnf install acl
   ```

2. Edit the `/etc/fstab` file and change the entries for any file systems that you want to use ACLs.

   Edit the `/etc/fstab` file and add the `acl` option to the file system entry. For example, the file might look as follows:

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

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

3. Remount the file systems that you edited.

   For example, you could run:

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

Setting and Displaying ACLs

Use the `setfacl` and `getfacl` commands to set and display ACLs on the file system.

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 no user is 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 no group is specified.

[d:]m[:][: ***permissions*****]**
Sets the effective rights mask, which is the union of all permissions of the owning group and all 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 and are used with the `chmod` command.

- `r`: read
- `w`: write
- `x`: execute

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 how to 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 changing 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:

  ```
  sudo setfacl -x u:user file
  ```
- To remove the rules for a group from the ACL of a file:

  ```
  sudo setfacl -x g:group file
  ```
- To remove all the extended ACL entries from a file or directory, specify the `-b` option:

  ```
  sudo setfacl -b file
  ```
- To copy the ACL of file *f1* to file *f2*:

  ```
  sudo getfacl f1 | setfacl --set-file=- f2
  ```
- To set a default ACL of read and execute access for other on a directory:

  ```
  sudo setfacl -m d:o:rx directory
  ```
- To promote the ACL settings of a directory to default ACL settings that can be inherited:

  ```
  sudo getfacl --access directory | setfacl -d -M- directory
  ```
- to remove the default ACL from a directory, specify the `-k` option:

  ```
  sudo setfacl -k directory
  ```

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

6

About Disk Quotas

You can set disk quotas to restrict the amount of disk space or *blocks* that users or groups can use, to limit the number of files or *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, which is known as a *grace period*.

Oracle Linux also includes project quotas for XFS and Ext file systems. Projects can be mapped to directories within the file system to apply hard and soft limits on file creation and disk usage for a project. For example, you can use project quotas to prevent logging utilities from using up excessive disk space or from creating too many files in the `/var/log` directory.

For information about how to configure quotas for an XFS file system, see Oracle Linux 9: Managing the XFS File System.

For information about how to configure quotas for an Ext file system, see Oracle Linux 9: Managing the Ext File System.

Oracle Linux doesn't provide support for user and group disk quotas for a Btrfs file system. However, quota handling at the subvolume level is available for a Btrfs file system as a technology preview in this release. See Oracle Linux 9: Managing the Btrfs File System.