

Disk and Storage Management

LinuXpert Systems

No.5, Dr. Ambedkar St, Rathinapuram,

Nanganallur, Chennai – 600 061.

Phone: 98841 65649, 90808 01382

E-mail: baskar@linuxpert.in

TABLE OF CONTENTS

1. df 1

2. du 7

3. fdisk 11

4. parted 18

5. mkfs 24

6. mount 28

7. umount 34

8. blkid 41

9. fsck 47

10. tune2fs 54

1. df

The **`df` (Disk Filesystem)** command is used to **display information about disk space usage** on mounted filesystems. It shows how much storage is **total, used, available**, and the **percentage of disk space used**.

Syntax

```
df [OP
TIONS] [FILE]
```

- **OPTIONS** – Modify the output format.
- **FILE** – Display information for the filesystem containing the specified file or directory.

Basic Example

```
df
```

Sample Output

```
Filesystem     1K-blocks     Used Available Use% Mounted on
/dev/sda2       51200000 20000000  28500000  42% /
tmpfs            2048000        0   2048000   0% /dev/shm
/dev/sda1         512000   150000    362000  30% /boot
```

Explanation of Columns

Column	Description
Filesystem	Name of the mounted filesystem
1K-blocks	Total size in 1 KB blocks
Used	Space currently used
Available	Free space available to non-root users
Use%	Percentage of filesystem used
Mounted on	Mount point (directory where the filesystem is mounted)

Common Options

1. Human-readable output (`-h`)

Displays sizes in KB, MB, GB, or TB.

```
df -h
```

Example:

```
Filesystem      Size Used Avail Use% Mounted on
/dev/sda2        50G  20G   28G  42% /
```

This is the most commonly used option.

---

2. Show filesystem type (`-T`)

```
df -T
```

Output:

```
Filesystem     Type   Size Used Avail Use% Mounted on
/dev/sda2      ext4    50G  20G   28G  42% /
```

Useful for identifying filesystems such as `ext4`, `xfs`, `btrfs`, `vfat`, etc.

---

3. Human-readable with filesystem type

```
df -Th
```

Example:

```
Filesystem Type Size Used Avail Use% Mounted on
/dev/sda2  ext4 50G 20G 28G 42% /
```

---

4. Display inode usage (`-i`)

Instead of disk blocks, show inode usage.

```
df -i
```

Example:

```
Filesystem     Inodes IUsed IFree IUse% Mounted on
/dev/sda2     3276800 55000 3221800   2% /
```

Useful when a filesystem reports "No space left on device" even though disk space remains—running out of inodes can cause that.

---

5. Human-readable inode information

```
df -ih
```

---

6. Show a specific filesystem

```
df /home
```

Output:

```
Filesystem      Size Used Avail Use% Mounted on
/dev/sda2        50G  20G   28G  42% /
```

Shows the filesystem containing `/home`.

---

7. Show all filesystems (`-a`)

```
df -a
```

Includes pseudo-filesystems such as `proc`, `sysfs`, and `tmpfs`.

---

8. Show only local filesystems (`-l`)

```
df -l
```

Excludes network-mounted filesystems.

9. Exclude a filesystem type

Exclude `tmpfs`:

```
df -x tmpfs
```

Exclude `tmpfs` and `devtmpfs`:

```
df -x tmpfs -x devtmpfs
```

---

10. Display information in MB

```
df -BM
```

Example:

```
Filesystem     1M-blocks Used Available Use%
/dev/sda2          51200 20000     28500 42%
```

---

11. Display information in GB

```
df -BG
```

---

12. POSIX output format

```
df -P
```

Provides consistent formatting suitable for scripts.

Practical Examples

Check overall disk usage

```
df -h
```

Check `/var`

```
df -h /var
```

Check USB drive

```
df -h /media/usb
```

View only ext4 filesystems

```
df -Th | grep ext4
```

Find nearly full filesystems

```
df -h | grep '9[0-9]%\|100%'
```

Shows filesystems using 90% or more of their capacity.

Difference Between `df` and `du`

`df`	`du`
Shows filesystem usage	Shows directory or file usage
Reports total and available disk space	Reports size of files/directories
Works at the filesystem level	Works at the file and directory level

Example:

```
df -h

Filesystem      Size Used Avail Use%
/dev/sda2        50G  20G   28G 42%

du -sh /home/user

3.8G    /home/user
```

Exit Status

- **0** – Command completed successfully.
- **Non-zero** – An error occurred (such as an invalid option or inaccessible path).

Summary

- **Command:** `df`
- **Purpose:** Display disk space usage of mounted filesystems.
- **Most common option:** `df -h`
- **Useful options:**
  - `-h` – Human-readable sizes
  - `-T` – Show filesystem type
  - `-i` – Show inode usage
  - `-a` – Include all filesystems
  - `-l` – Show only local filesystems
  - `-x TYPE` – Exclude a filesystem type
  - `-P` – POSIX output for scripting

For day-to-day system administration, `df -h` is the standard command to quickly check how much free disk space is available on your Linux system.

2. du

The du (**Disk Usage**) command is used to **estimate and display the disk space used by files and directories**. It helps you identify which files or folders are consuming storage.

Syntax

```
du [OPTIONS] [FILE or DIRECTORY]
```

- **OPTIONS** – Modifies the command's behavior.
- **FILE or DIRECTORY** – The file or directory whose disk usage you want to check. If omitted, du checks the current directory.

Basic Examples

1. Check disk usage of the current directory

```
du
```

Example Output

```
4       ./Documents
12      ./Downloads
20      .
```

---

2. Check a specific directory

```
du /home/bharath
```

---

3. Show sizes in human-readable format

```
du -h
```

Example Output

```
4.0K    ./Documents
12M     ./Downloads
1.5G    ./Videos
```

The `-h` option displays sizes in KB, MB, GB, etc.

---

4. Show only the total size of a directory

```
du -sh /home/bharath
```

Example Output

```
8.4G    /home/bharath
```

- `-s` = Summary only
- `-h` = Human-readable

---

5. Show sizes of all subdirectories

```
du -h /home/bharath
```

---

6. Display directory sizes up to one level deep

```
du -h --max-depth=1
```

Example Output

```
120M    ./Downloads
1.2G    ./Videos
85M     ./Documents
1.4G    .
```

---

7. Find the largest directories

```
du -h --max-depth=1 | sort -hr
```

Example Output

```
8.2G    .
5.0G    ./Videos
2.1G    ./Downloads
500M    ./Documents
```

8. Show disk usage of individual files

```
du -ah
```

---

9. Exclude a directory

```
du -h --exclude=Downloads
```

---

10. Compare multiple directories

```
du -sh Documents Downloads Videos
```

Example Output

```
150M    Documents
2.3G    Downloads
12G     Videos
```

Common Options

Option	Description
`-h`	Human-readable output (KB, MB, GB)
`-s`	Show only the total size (summary)
`-a`	Include files as well as directories
`-c`	Display a grand total
`-d N`	Show directory sizes up to depth `N`
`--max-depth=N`	Same as `-d N` (GNU version)
`-x`	Stay on one filesystem (don't cross mount points)
`--exclude=PATTERN`	Exclude matching files/directories

Difference Between `du` and `df`

du	df
Shows space used by files and directories	Shows free and used space on filesystems
File/directory level	Filesystem/partition level
Helps find large folders	Helps monitor disk capacity

Example:

```
du -sh /home/bharath
```

Shows:

```
8.4G /home/bharath
```

```
df -h
```

Shows:

```
Filesystem      Size Used Avail Use%
/dev/sda2        50G  32G   16G  67%
```

3. fdisk

`fdisk` is a **disk partition management** utility used to **create, delete, modify, and view partitions** on hard drives, SSDs, and other block storage devices. It works primarily with **MBR (Master Boot Record)** partition tables, but modern versions also support **GPT (GUID Partition Table)**.

It is one of the most commonly used tools by Linux system administrators when preparing storage devices.

Syntax

```
fdisk [options] <disk>
```

Example:

```
fdisk /dev/sdb
```

This opens the partition editor for the disk `/dev/sdb`.

Common Options

Option	Description
`-l`	List partition tables of all disks
`-L`	List available partition types
`-s`	Display partition size (older systems)
`-x`	Display detailed partition information
`-u`	Display sectors instead of cylinders (older systems)

Examples

1. List all disks and partitions

```
sudo fdisk -l
```

Output:

```
Disk /dev/sda: 500 GiB
Disklabel type: gpt

Device        Start       End   Sectors  Size Type
/dev/sda1      2048   1050623   1048576  512M EFI System
/dev/sda2   1050624 976773119 975722496 465G Linux filesystem
```

This command only displays information.

---

2. View one specific disk

```
sudo fdisk -l /dev/sdb
```

---

3. Open partition editor

```
sudo fdisk /dev/sdb
```

After opening, an interactive menu appears.

Interactive Commands

When inside `fdisk`, type a single letter.

Command	Purpose
`m`	Show help menu
`p`	Print partition table
`n`	Create new partition
`d`	Delete partition
`t`	Change partition type
`a`	Toggle bootable flag
`g`	Create a new GPT partition table
`o`	Create a new MBR partition table
`w`	Write changes to disk and exit
`q`	Quit without saving
`i`	Show partition information

Example: Create a New Partition

Open the disk:

```
sudo fdisk /dev/sdb
```

Inside `fdisk`:

```
Command (m for help): n
```

Choose:

```
Partition type: primary
Partition number: 1
First sector: (Press Enter)
Last sector: +5G
```

Save:

```
Command (m for help): w
```

The partition table is written to disk.

Example: Delete a Partition

```
sudo fdisk /dev/sdb
```

Inside:

```
Command (m for help): d
```

Select partition:

```
Partition number: 2
```

Save:

```
w
```

Example: Print Current Partitions

```
Command (m for help): p
```

Example output:

```
Device      Start      End  Sectors Size Type
/dev/sdb1    2048 20973567 20971520  10G Linux filesystem
```

Creating a GPT Partition Table

Inside `fdisk`:

```
g
```

Creates a new GPT partition table.

Creating an MBR Partition Table

```
o
```

Creates a DOS/MBR partition table.

Checking the Changes

After creating partitions:

```
lsblk
```

or

```
sudo fdisk -l
```

Refresh the Kernel Partition Table

Sometimes Linux needs to re-read the partition table:

```
sudo partprobe
```

or

```
sudo partx -u /dev/sdb
```

Typical Workflow

```
sudo fdisk /dev/sdb
```

Inside `fdisk`:

```
g      # Create GPT table (optional)
n      # New partition
Enter
Enter
+20G
w      # Save changes
```

Then create a filesystem:

```
sudo mkfs.ext4 /dev/sdb1
```

Mount it:

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

Advantages

- Installed on almost all Linux distributions.
- Simple and lightweight.
- Suitable for scripting (with care).
- Supports both MBR and GPT.
- Reliable for partition management.

Limitations

- Text-based interface (no graphical UI).
- Editing partitions on a mounted or in-use disk can be risky.
- Resizing partitions is limited compared to tools such as `parted` or `gparted`.
- Changes made with `w` are written immediately and can result in data loss if used incorrectly.

Exit Without Saving

```
Command (m for help): q
```

Exit and Save

```
Command (m for help): w
```

Related Commands

Command	Purpose
`lsblk`	Display block devices and partitions
`blkid`	Show filesystem UUIDs and types
`parted`	Advanced partition editor
`partprobe`	Notify the kernel of partition table changes
`partx`	Inform the kernel about partition table updates
`mkfs.ext4`	Create an ext4 filesystem
`mount`	Mount a filesystem
`umount`	Unmount a filesystem

Warning: `fdisk` modifies partition tables directly. Always verify you are working on the correct disk (for example, `/dev/sdb` instead of `/dev/sda`) before using the `w` command, as changes can destroy existing data.

4. parted

`parted` is a **disk partition management utility** used to create, delete, resize, and modify partitions on storage devices such as HDDs, SSDs, USB drives, and SD cards.

Unlike older tools like `fdisk`, `parted` supports both **MBR (MS-DOS)** and **GPT (GUID Partition Table)** partition tables and can handle disks larger than 2 TB.

Syntax

```
parted [options] [device] [command]
```

Example:

```
parted /dev/sda
```

This opens `parted` in interactive mode for the disk `/dev/sda`.

Common Options

Option	Description
`-l`	List all disks and partitions
`-s`	Script mode (non-interactive)
`--help`	Display help
`--version`	Show version information

Common Commands Inside `parted`

Command	Description
`print`	Display partition table
`mklabel`	Create a new partition table
`mkpart`	Create a partition
`rm`	Delete a partition
`resizepart`	Resize a partition
`name`	Name a GPT partition
`quit`	Exit parted

Examples

1. List all disks

```
sudo parted -l
```

Example output:

```
Model: ATA Samsung SSD
Disk /dev/sda: 500GB

Number  Start   End    Size    Type
 1      1049kB  538MB  537MB   primary
 2      538MB   500GB  499GB   primary
```

2. Open a disk

```
sudo parted /dev/sdb
```

Interactive prompt:

```
(parted)
```

---

3. View partitions

Inside `parted`:

```
(parted) print
```

Output:

```
Number  Start   End     Size
1       1MB     512MB   511MB
2       512MB   64GB    63.5GB
```

---

4. Create a GPT partition table

```
(parted) mklabel gpt
```

Warning: This erases the existing partition table.

5. Create a new partition

```
(parted) mkpart primary ext4 1MiB 20GiB
```

Meaning:

- Filesystem type: `ext4` (informational only)
- Starts at 1 MiB
- Ends at 20 GiB

---

6. Delete a partition

```
(parted) rm 2
```

Deletes partition number 2.

---

7. Resize a partition

```
(parted) resizepart 2 100GB
```

Changes the end of partition 2 to 100 GB.

> Note: `parted` only changes the partition boundary. You usually need to resize the filesystem separately (for example, using `resize2fs` for ext4).

---

8. Name a GPT partition

```
(parted) name 1 Boot
```

---

9. Exit

```
(parted) quit
```

Non-Interactive (Script Mode)

Create a partition without entering the interactive shell:

```
sudo parted -s /dev/sdb mklabel gpt
```

Create a partition:

```
sudo parted -s /dev/sdb mkpart primary ext4 1MiB 10GiB
```

Understanding the Units

`parted` supports multiple units:

Unit	Meaning
B	Bytes
KB	Kilobytes
MB	Megabytes
GB	Gigabytes
TB	Terabytes
KiB	Kibibytes
MiB	Mebibytes
GiB	Gibibytes
%	Percentage of disk

Example:

```
mkpart primary ext4 0% 50%
```

Creates a partition using the first half of the disk.

MBR vs GPT

Feature	MBR	GPT
Maximum disk size	2 TB	Over 2 TB
Maximum primary partitions	4	Typically 128
Redundant partition table	No	Yes
UEFI support	Limited	Yes

`parted` works with both formats.

Common Workflow

```
sudo parted /dev/sdb

(parted) print
(parted) mklabel gpt
(parted) mkpart primary ext4 1MiB 50GiB
(parted) print
(parted) quit
```

Then create a filesystem:

```
sudo mkfs.ext4 /dev/sdb1
```

When to Use `parted`

Use `parted` when you need to:

- Create or delete partitions.
- Work with GPT partition tables.
- Partition disks larger than 2 TB.
- Resize partitions.
- Automate partitioning using scripts.
- Prepare disks for Linux installation or storage.

Advantages

- Supports both MBR and GPT.
- Handles very large disks.
- Can resize partitions.
- Suitable for scripting (`-s` option).
- Available on most Linux distributions.

Disadvantages

- Modifying partitions can lead to **data loss** if done incorrectly.
- Resizing partitions does not automatically resize the filesystem.
- Changes to the partition table may require a reboot or `partprobe` to notify the kernel.

Related Commands

Command	Purpose
`fdisk`	Partition MBR disks (also supports GPT in newer versions)
`cfdisk`	Menu-driven partition editor
`gdisk`	GPT-specific partition editor
`lsblk`	Display block devices and partitions
`blkid`	Show filesystem UUIDs and types
`partprobe`	Notify the kernel of partition table changes
`mkfs`	Create a filesystem on a partition
`mount`	Mount a filesystem
`df`	Display mounted filesystem usage

Warning: Operations such as `mklabel`, `mkpart`, `rm`, and `resizepart` modify the partition table. Always verify the target device (for example, `/dev/sdb` vs. `/dev/sda`) and back up important data before making changes.

5. mkfs

The `mkfs` command is used to **create (format) a file system** on a storage device or partition. Before a partition can store files, it must have a file system such as ext4, XFS, FAT32, or NTFS.

Full Form: Make **F**ile **S**ystem

Syntax

```
mkfs [options] device
```

or

```
mkfs -t filesystem_type device
```

Common File System Types

File System	Description
ext4	Default Linux file system
ext3	Older Linux file system
ext2	Linux file system without journaling
xfs	High-performance Linux file system
btrfs	Modern Linux file system with snapshots
vfat (FAT32)	Compatible with Windows and Linux
ntfs	Windows NT File System

Common Options

Option	Description
`-t type`	Specify the file system type
`-V`	Display verbose information
`-h`	Show help

Examples

1. Create an ext4 file system

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

or

```
sudo mkfs.ext4 /dev/sdb1
```

Creates an **ext4** file system on the partition `/dev/sdb1`.

---

2. Create an XFS file system

```
sudo mkfs -t xfs /dev/sdb1
```

or

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

---

3. Create a FAT32 file system

```
sudo mkfs -t vfat /dev/sdb1
```

or

```
sudo mkfs.vfat /dev/sdb1
```

Useful for USB flash drives that need Windows compatibility.

---

4. Create an ext3 file system

```
sudo mkfs.ext3 /dev/sdb1
```

---

5. Display verbose output

```
sudo mkfs -V -t ext4 /dev/sdb1
```

Before Using `mkfs`

Suppose a new disk is connected.

Check available disks

```
lsblk
```

Example output:

```
NAME   SIZE TYPE MOUNTPOINT
sda    500G disk
├─sda1 100G part /
├─sda2 400G part /home
sdb    100G disk
└─sdb1 100G part
```

To format `sdb1`:

```
sudo mkfs.ext4 /dev/sdb1
```

Mount After Formatting

Create a mount point:

```
sudo mkdir /data
```

Mount the partition:

```
sudo mount /dev/sdb1 /data
```

Verify:

```
df -h
```

Check the File System

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

Example output:

```
/dev/sdb1: UUID="d9d8..." TYPE="ext4"
```

Warning

`mkfs` **erases all existing data** on the specified partition or device.
For example:

```
sudo mkfs.ext4 /dev/sdb1
```

All files previously stored on `/dev/sdb1` will be permanently removed.

Always verify the correct device before running `mkfs`:

```
lsblk
```

Related Commands

Command	Purpose
`fdisk`	Create and manage disk partitions
`parted`	Advanced partition management
`lsblk`	List disks and partitions
`blkid`	Display file system type and UUID
`mount`	Mount a file system
`umount`	Unmount a file system
`fsck`	Check and repair a file system
`df`	Show mounted file system disk usage

Summary

- `mkfs` stands for **Make File System**.
- It formats a partition or disk with a file system.
- Common Linux file systems include **ext4**, **XFS**, and **Btrfs**.
- The general syntax is:

  ```
  mkfs -t <filesystem_type> <device>
  ```

- Formatting destroys all existing data on the target device, so always confirm the correct partition with `lsblk` before proceeding.

6. mount

The `mount` command is used to **attach (mount) a filesystem** on a storage device (such as a hard disk, USB drive, CD/DVD, or network share) to the Linux directory tree so that its files can be accessed.

In Linux, every file and directory is part of a single directory hierarchy starting from the root directory (`/`). Before you can use a storage device, its filesystem must be mounted to a directory called a **mount point**.

Syntax

```
mount [options] <device> <mount_point>
```

or

```
mount [options]
```

(to display currently mounted filesystems)

Basic Examples

1. View all mounted filesystems

```
mount
```

Sample Output

```
/dev/sda2 on / type ext4 (rw,relatime)
proc on /proc type proc (rw,nosuid,nodev,noexec)
tmpfs on /run type tmpfs (rw,nosuid,nodev)
```

This lists all mounted filesystems and their options.

---

2. Mount a USB drive

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

This mounts the partition `/dev/sdb1` to the `/mnt` directory.

You can now access its files:

```
cd /mnt
ls
```

---

3. Mount using a custom directory

Create a mount point:

```
sudo mkdir /myusb
```

Mount the drive:

```
sudo mount /dev/sdb1 /myusb
```

---

4. Mount a CD/DVD

```
sudo mount /dev/sr0 /media/cdrom
```

---

5. Mount an ISO image

```
sudo mount -o loop ubuntu.iso /mnt
```

`-o loop` treats the ISO file as a virtual disk.

---

6. Mount a filesystem as read-only

```
sudo mount -o ro /dev/sdb1 /mnt
```

Users can read files but cannot modify them.

7. Remount a filesystem

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

Change the root filesystem to read-write.

---

8. Mount all filesystems listed in `/etc/fstab`

```
sudo mount -a
```

This mounts every filesystem defined in `/etc/fstab` that is not already mounted.

Common Options

Option	Description
`-a`	Mount all filesystems from `/etc/fstab`
`-o`	Specify mount options
`-t`	Specify filesystem type
`-r`	Mount read-only
`-w`	Mount read-write
`-v`	Verbose mode
`-L`	Mount by filesystem label
`-U`	Mount by UUID

Common Mount Options (`-o`)

Option	Meaning
`ro`	Read-only
`rw`	Read-write
`noexec`	Prevent execution of programs
`exec`	Allow execution
`nosuid`	Ignore SUID/SGID bits
`nodev`	Ignore device files
`defaults`	Default mount options
`remount`	Change mount options without unmounting
`loop`	Mount an image file

Example:

```
sudo mount -o rw,noexec /dev/sdb1 /mnt
```

Specify Filesystem Type

Mount an ext4 partition:

```
sudo mount -t ext4 /dev/sdb1 /mnt
```

Mount a FAT32 partition:

```
sudo mount -t vfat /dev/sdb1 /mnt
```

Mount an NTFS partition:

```
sudo mount -t ntfs /dev/sdb1 /mnt
```

Mount by UUID

Find UUIDs:

```
blkid
```

Mount using the UUID:

```
sudo mount UUID=1234-ABCD /mnt
```

Mount by Label

```
sudo mount LABEL=DATA /mnt
```

Verify the Mount

```
mount | grep sdb1
```

or

```
findmnt
```

or

```
df -h
```

Unmount When Finished

After using the device, unmount it:

```
sudo umount /mnt
```

or

```
sudo umount /dev/sdb1
```

> **Note:** The command is `umount` (without the first **n**), not `unmount`.

Typical Workflow

```
lsblk
```

Identify the device:

```
sdb
└── sdb1
```

Create a mount point:

```
sudo mkdir /mnt/usb
```

Mount it:

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

Access the files:

```
cd /mnt/usb
ls
```

When finished:

```
cd ~
sudo umount /mnt/usb
```

Real-World Uses

- Access files on USB flash drives.
- Mount external HDDs and SSDs.
- Mount CD/DVD media.
- Mount ISO images.
- Mount additional hard disk partitions.
- Mount network filesystems (NFS, SMB/CIFS).
- Remount filesystems with different permissions (e.g., read-only or read-write).

Important Notes

- Most mount operations require **root (sudo)** privileges.
- The mount point directory must already exist.
- A mounted filesystem hides any existing files in the mount point until it is unmounted.
- Use `lsblk`, `blkid`, or `fdisk -l` to identify device names before mounting.

Summary

Feature	Description
Purpose	Attach a filesystem to the Linux directory tree
Requires root?	Usually Yes
Common command	`sudo mount /dev/sdb1 /mnt`
Show mounted filesystems	`mount`
Mount all from `/etc/fstab`	`mount -a`
Unmount command	`umount`
Supports	ext4, xfs, NTFS, FAT32, ISO, NFS, SMB/CIFS, and more

7. umount

The `umount` command is used to **unmount a mounted filesystem** from the Linux directory tree. Once a filesystem is unmounted, its files and directories are no longer accessible through the mount point.

Note: The command is spelled **`umount`**, not `unmount`.

Syntax

```
umount [OPTIONS] <mount_point | device>
```

or

```
umount /dev/sdb1
```

or

```
umount /mnt/usb
```

Why is `umount` used?

Linux allows storage devices (hard disks, USB drives, ISO images, network shares, etc.) to be attached to the filesystem using the `mount` command.

Before removing a device, you should unmount it to:

- Flush cached data to disk
- Prevent filesystem corruption
- Release the filesystem from the operating system
- Safely remove the device

Basic Examples

1. Unmount using the mount point

```
umount /mnt/usb
```

Unmounts the filesystem mounted on `/mnt/usb`.

---

2. Unmount using the device name

```
umount /dev/sdb1
```

Unmounts the partition `/dev/sdb1`.

---

3. Check mounted filesystems

Before unmounting:

```
mount
```

or

```
findmnt
```

Example:

```
/dev/sdb1 on /mnt/usb type ext4
```

Then:

```
umount /mnt/usb
```

Common Options

Option	Description
`-l`	Lazy unmount
`-f`	Force unmount (mainly for network filesystems)
`-R`	Recursively unmount
`-a`	Unmount all filesystems listed in `/etc/mtab` (except required ones)
`-v`	Verbose output
`-r`	Remount read-only if unmount fails

Option Examples

Lazy unmount

```
umount -l /mnt/usb
```

The filesystem is detached immediately, but cleanup occurs once it is no longer in use.

Useful when a process is still using the filesystem.

Force unmount

```
umount -f /mnt/nfs
```

Typically used for inaccessible network filesystems (such as NFS).

Recursive unmount

```
umount -R /mnt/storage
```

Unmounts the specified mount point and any filesystems mounted beneath it.

Verbose mode

```
umount -v /mnt/usb
```

Example output:

```
/mnt/usb unmounted
```

Common Error

Device is busy

Example:

```
umount /mnt/usb
```

Output:

```
umount: /mnt/usb: target is busy
```

This means some process is still accessing the filesystem.

Find the processes using it:

```
lsof /mnt/usb
```

or

```
fuser -vm /mnt/usb
```

Stop the process, then unmount:

```
kill <PID>
umount /mnt/usb
```

Practical Example

Step 1: Insert a USB drive

Check its name:

```
lsblk
```

Output:

```
sdb
└── sdb1
```

Step 2: Mount it

```
mount /dev/sdb1 /mnt/usb
```

Step 3: Copy files

```
cp report.pdf /mnt/usb/
```

Step 4: Unmount

```
umount /mnt/usb
```

Now it is safe to unplug the USB drive.

Unmount an ISO image

```
umount /mnt/iso
```

Unmount an NFS share

```
umount /mnt/nfs
```

If the server is unreachable:

```
umount -f /mnt/nfs
```

Unmount all removable filesystems

```
umount -a
```

Warning: Be careful with `umount -a`, as it attempts to unmount many mounted filesystems. It is usually used during system shutdown or maintenance.

Permissions

Unmounting typically requires root privileges:

```
sudo umount /mnt/usb
```

Some removable devices mounted by desktop environments may be unmounted by the user who mounted them.

Related Commands

Command	Purpose
`mount`	Mount a filesystem
`findmnt`	Display mounted filesystems
`lsblk`	List block devices
`df`	Show mounted filesystem disk usage
`blkid`	Display block device attributes (UUID, filesystem type)
`lsof`	List open files
`fuser`	Show processes using files or filesystems
`sync`	Flush filesystem buffers to disk

Summary

- **Purpose:** Detach a mounted filesystem from the Linux directory tree.
- **Syntax:** `umount <mount_point>` or `umount <device>`
- **Common uses:** Safely remove USB drives, detach external disks, unmount ISO images, disconnect network shares, and prepare storage for maintenance.
- **Common issue:** If you see **"target is busy"**, identify the processes using the filesystem with `lsof` or `fuser`, stop them if appropriate, and then retry the unmount.

8. blkid

The `blkid` command is used in Linux to **display or identify information about block devices**, such as hard disks, SSDs, USB drives, partitions, and their filesystems.

It is commonly used by system administrators to determine:

- Filesystem type (ext4, xfs, ntfs, vfat, etc.)
- UUID (Universally Unique Identifier)
- Partition labels
- Partition UUID (PARTUUID)

Syntax

```
blkid [options] [device]
```

Basic Example

```
blkid
```

Sample Output

```
/dev/sda1: UUID="9d8b8a18-f85e-4d8f-a4f5-4a6d1a7d55e1" TYPE="ext4"
PARTUUID="3d8a2f3b-01"
/dev/sda2: UUID="1E2A-9F10" TYPE="vfat" PARTLABEL="EFI System Partition"
/dev/sdb1: UUID="A84C4B6F4C4B35E2" TYPE="ntfs"
```

Explanation

- **/dev/sda1** – Device name
- **UUID** – Unique identifier of the filesystem
- **TYPE** – Filesystem type
- **PARTUUID** – Unique identifier of the partition
- **PARTLABEL** – Partition label (if available)

Why is `blkid` useful?

It helps identify storage devices even if device names change.

For example:

Today:

```
USB Drive → /dev/sdb1
```

Tomorrow:

```
USB Drive → /dev/sdc1
```

The device name changed, but its **UUID** remains the same.

That's why Linux uses UUIDs in `/etc/fstab` for mounting filesystems reliably.

Example:

```
UUID=9d8b8a18-f85e-4d8f-a4f5-4a6d1a7d55e1 /home ext4 defaults 0 2
```

Common Options

1. Show all block devices

```
blkid
```

Displays information for all detected block devices.

---

2. Show information for a specific device

```
blkid /dev/sda1
```

Example output:

```
/dev/sda1: UUID="9d8b8a18-f85e-4d8f-a4f5-4a6d1a7d55e1" TYPE="ext4"
```

3. Show only a specific field

Display only the UUID:

```
blkid -s UUID /dev/sda1
```

Output:

```
UUID="9d8b8a18-f85e-4d8f-a4f5-4a6d1a7d55e1"
```

Display filesystem type:

```
blkid -s TYPE /dev/sda1
```

Output:

```
TYPE="ext4"
```

---

4. Display values only

```
blkid -s UUID -o value /dev/sda1
```

Output:

```
9d8b8a18-f85e-4d8f-a4f5-4a6d1a7d55e1
```

Useful in shell scripts.

---

5. Find devices by UUID

```
blkid -U 9d8b8a18-f85e-4d8f-a4f5-4a6d1a7d55e1
```

Output:

```
/dev/sda1
```

---

6. Find devices by label

```
blkid -L DATA
```

Output:

```
/dev/sdb1
```

7. Probe a device

```
blkid -p /dev/sdb1
```

Performs a low-level probe to detect filesystem signatures.

Common Fields

Field	Meaning
UUID	Filesystem unique identifier
TYPE	Filesystem type
LABEL	Filesystem label
PARTUUID	Partition unique identifier
PARTLABEL	Partition label
SEC_TYPE	Secondary filesystem type (if applicable)

Practical Examples

Check the filesystem type

```
blkid /dev/sdb1
```

Output:

```
TYPE="ntfs"
```

Get the UUID for `/etc/fstab`

```
blkid -s UUID -o value /dev/sda2
```

Output:

```
1E2A-9F10
```

List only ext4 partitions

```
blkid | grep ext4
```

Example output:

```
/dev/sda1: UUID="..." TYPE="ext4"
/dev/sda3: UUID="..." TYPE="ext4"
```

Difference Between `blkid` and `lsblk`

`blkid`	`lsblk`
Shows filesystem metadata	Shows block device hierarchy
Displays UUID, LABEL, TYPE	Displays disks, partitions, mount points, sizes
Best for identifying filesystems	Best for viewing storage layout

Example:

```
lsblk

NAME   SIZE TYPE MOUNTPOINT
sda    500G disk
├─sda1 100G part /
├─sda2   1G part /boot
└─sda3 399G part /home

blkid

/dev/sda1: UUID="..." TYPE="ext4"
/dev/sda2: UUID="..." TYPE="ext4"
/dev/sda3: UUID="..." TYPE="xfs"
```

When to Use `blkid`

- Find the UUID of a partition.
- Check the filesystem type before mounting.
- Configure `/etc/fstab`.
- Identify USB drives and external disks.
- Verify partition labels and UUIDs.
- Troubleshoot storage and mounting issues.

Summary

Feature	Description
Command	`blkid`
Purpose	Display block device attributes and filesystem metadata
Common Use	Find UUIDs, filesystem types, labels, and partition identifiers
Typical Users	Linux administrators, system engineers, and users managing storage devices

9. fsck

`fsck` (**F**ile **S**ystem **C**hec**k**) is a Linux utility used to **check and repair filesystem errors** on storage devices such as hard disks, SSDs, USB drives, and partitions.

It scans a filesystem for inconsistencies and, if possible, fixes corrupted metadata and filesystem structures.

> **Important:** `fsck` should **never be run on a mounted read/write filesystem**, as it may cause data corruption. Always unmount the filesystem first, or run it from recovery mode.

Syntax

```
fsck [options] <device>
```

Example:

```
fsck /dev/sdb1
```

Common Options

Option	Description
`-A`	Check all filesystems listed in `/etc/fstab`
`-C`	Display progress bar
`-f`	Force filesystem check even if clean
`-N`	Show what would be done without executing
`-R`	Skip the root filesystem when using `-A`
`-t`	Specify filesystem type
`-y`	Automatically answer "yes" to all repair questions
`-n`	Answer "no" to all repair questions (read-only check)
`-V`	Verbose output

Basic Examples

1. Check a filesystem

```
fsck /dev/sda2
```

Checks the filesystem and asks before repairing errors.

---

2. Automatically repair errors

```
fsck -y /dev/sda2
```

Automatically fixes all repairable errors.

Useful for unattended recovery.

---

3. Force a filesystem check

```
fsck -f /dev/sda2
```

Runs a check even if the filesystem appears clean.

---

4. Dry run

```
fsck -N /dev/sda2
```

Shows the command that would be executed without actually checking the filesystem.

---

5. Check all filesystems

```
fsck -A
```

Checks all filesystems listed in `/etc/fstab`.

6. Show progress

```
fsck -C /dev/sda2
```

Displays a progress indicator while checking.

---

7. Specify filesystem type

```
fsck -t ext4 /dev/sda2
```

Checks an `ext4` filesystem.

Checking the Root Filesystem

The root (`/`) filesystem is normally mounted, so you cannot safely run:

```
fsck /
```

Instead:

1. Boot into **Recovery Mode**, or
2. Boot from a **Live USB**, or
3. Force a check on the next reboot:

```
sudo touch /forcefsck
sudo reboot
```

Check an Unmounted Partition

Unmount the partition first:

```
sudo umount /dev/sdb1
sudo fsck /dev/sdb1
```

Filesystem-Specific Checkers

`fsck` is a front-end that automatically calls the appropriate checker:

Filesystem	Command Used
ext2	`e2fsck`
ext3	`e2fsck`
ext4	`e2fsck`
XFS	`xfs_repair` (not through `fsck` while mounted)
FAT	`fsck.fat`
VFAT	`fsck.vfat`
NTFS	`ntfsfix`

Example:

```
fsck.ext4 /dev/sda2
```

or

```
e2fsck /dev/sda2
```

Typical Workflow

Find partitions:

```
lsblk
```

Example output:

```
sda
├── sda1
├── sda2
└── sda3
```

Unmount the partition:

```
sudo umount /dev/sda3
```

Run the check:

```
sudo fsck /dev/sda3
```

Sample Output

```
fsck from util-linux 2.39
e2fsck 1.47.0

/dev/sda2: clean,
254328/655360 files,
1023456/2621440 blocks
```

If errors are found:

```
Inode 12345 has invalid mode.

Fix? yes
```

Using `-y` answers these prompts automatically.

Exit Status Codes

Exit Code	Meaning
`0`	No errors found
`1`	Filesystem errors corrected
`2`	System should be rebooted
`4`	Filesystem errors left uncorrected
`8`	Operational error
`16`	Usage or syntax error
`32`	Check cancelled by user
`128`	Shared library error

Example:

```
echo $?
```

Displays the exit status of the last `fsck` command.

Practical Example

Suppose `/dev/sdb1` has filesystem errors.

```
sudo umount /dev/sdb1
sudo fsck -f /dev/sdb1
```

If you want automatic repairs:

```
sudo fsck -fy /dev/sdb1
```

When to Use `fsck`

- After an improper shutdown or power failure.
- When the system reports filesystem errors.
- Before mounting a disk suspected of corruption.
- During disk maintenance.
- To verify the integrity of removable drives (USB drives, SD cards).

Best Practices

- **Always unmount** the filesystem before running `fsck` (except when checking the root filesystem from recovery mode).
- **Back up important data** before repairing a damaged filesystem, as severe corruption can lead to data loss.
- Use `-n` first if you only want to inspect for errors without making changes.
- Use `-y` with caution because it automatically accepts all repairs.
- If `fsck` repeatedly reports new errors, the underlying storage device may be failing. Check its SMART health with tools such as `smartctl` (from the `smartmontools` package).

Summary

- **Full form:** File System Check
- **Purpose:** Detects and repairs filesystem errors.
- **Syntax:** `fsck [options] <device>`
- **Safe practice:** Run it only on unmounted filesystems (or from recovery mode for the root filesystem).
- **Common options:** `-f`, `-y`, `-n`, `-A`, `-C`, `-t`, `-V`.
- **Typical use:** `sudo fsck -fy /dev/sdb1` to force a check and automatically repair errors on an unmounted partition.

10. tune2fs

`tune2fs` is a Linux command used to **view or modify the parameters of an ext2, ext3, or ext4 filesystem** without deleting or recreating it. It is commonly used by system administrators to change filesystem settings, labels, UUIDs, mount counts, and filesystem check intervals.

Syntax

```
tune2fs [options] device
```

- **device** – The filesystem device (for example, `/dev/sda1`).

What `tune2fs` Can Do

- Display filesystem information
- Change the filesystem label
- Change the UUID
- Set reserved block percentage
- Configure automatic filesystem checks
- Enable or disable filesystem features
- Change error handling behavior
- Set mount count limits

Common Options

Option	Description
`-l`	Display filesystem information
`-L label`	Set filesystem label
`-U UUID`	Change filesystem UUID
`-m percentage`	Set reserved block percentage
`-c count`	Set maximum mount count before fsck
`-i interval`	Set time interval between filesystem checks
`-e behavior`	Set action when filesystem errors occur
`-o options`	Set default mount options
`-E options`	Set extended filesystem options

Examples

1. Display Filesystem Information

```
sudo tune2fs -l /dev/sda1
```

Example output:

```
Filesystem volume name:   DATA
Filesystem UUID:          a6d2-4b8f
Filesystem state:         clean
Mount count:              15
Maximum mount count:      30
Reserved block count:     131072
Filesystem features:      has_journal ext_attr resize_inode
```

Explanation

Displays detailed information about the filesystem.

---

2. Change the Filesystem Label

```
sudo tune2fs -L BackupDrive /dev/sdb1
```

Now the partition label becomes:

```
BackupDrive
```

Useful for identifying partitions.

---

3. View the Label

```
sudo tune2fs -l /dev/sdb1 | grep "volume name"
```

Output:

```
Filesystem volume name: BackupDrive
```

4. Change the UUID

Generate a random UUID:

```
sudo tune2fs -U random /dev/sdb1
```

Assign a specific UUID:

```
sudo tune2fs -U 11111111-2222-3333-4444-555555555555 /dev/sdb1
```

This is useful after cloning disks to avoid duplicate UUIDs.

---

5. Set Reserved Block Percentage

By default, ext filesystems reserve **5%** of disk space for the root user.

Change it to **1%**:

```
sudo tune2fs -m 1 /dev/sda1
```

Check:

```
sudo tune2fs -l /dev/sda1 | grep "Reserved block"
```

Useful for large data partitions where reserving 5% would waste significant space.

---

6. Set Maximum Mount Count

Run a filesystem check after 40 mounts:

```
sudo tune2fs -c 40 /dev/sda1
```

Disable mount-count checks:

```
sudo tune2fs -c 0 /dev/sda1
```

7. Set Filesystem Check Interval

Run `fsck` every three months:

```
sudo tune2fs -i 3m /dev/sda1
```

Other examples:

```
sudo tune2fs -i 30d /dev/sda1
sudo tune2fs -i 1w /dev/sda1
sudo tune2fs -i 1y /dev/sda1
```

Units:

- `d` – days
- `w` – weeks
- `m` – months
- `y` – years

---

8. Set Error Handling

Remount filesystem as read-only on errors:

```
sudo tune2fs -e remount-ro /dev/sda1
```

Other options:

```
sudo tune2fs -e continue /dev/sda1
sudo tune2fs -e panic /dev/sda1
```

Option	Meaning
`continue`	Continue despite errors
`remount-ro`	Remount as read-only
`panic`	Trigger a kernel panic

9. Enable Default Mount Options

```
sudo tune2fs -o journal_data /dev/sda1
```

or

```
sudo tune2fs -o acl,user_xattr /dev/sda1
```

---

10. Display Mount Count

```
sudo tune2fs -l /dev/sda1 | grep "Mount count"
```

Output:

```
Mount count: 12
Maximum mount count: 30
```

Real-World Uses

Server Administration

Reduce reserved space on a 10 TB storage drive:

```
sudo tune2fs -m 1 /dev/sdc1
```

Rename a Partition

```
sudo tune2fs -L Projects /dev/sdb2
```

Fix Duplicate UUIDs After Cloning

```
sudo tune2fs -U random /dev/sdb1
```

Configure Automatic Filesystem Checks

```
sudo tune2fs -c 30 -i 6m /dev/sda1
```

This schedules a filesystem check after **30 mounts** or **6 months**, whichever comes first.

Important Notes

- `tune2fs` works **only** with **ext2**, **ext3**, and **ext4** filesystems.
- Most operations require **root (`sudo`) privileges**.
- Some changes should be made while the filesystem is **unmounted** or mounted read-only to avoid issues.
- Changing the **UUID** may require updating `/etc/fstab` or the bootloader configuration if they reference the old UUID.

Difference Between `tune2fs` and `mkfs`

`tune2fs`	`mkfs`
Modifies an existing ext filesystem	Creates a new filesystem
Preserves existing data (for supported changes)	Erases existing data
Changes labels, UUIDs, reserved blocks, check intervals, etc.	Formats a partition from scratch

Summary

- **Purpose:** View and modify parameters of existing ext2/ext3/ext4 filesystems.
- **Common uses:** Change labels, UUIDs, reserved block percentage, mount limits, and filesystem check intervals.
- **Filesystem support:** ext2, ext3, ext4 only.
- **Privileges:** Usually requires `sudo`.
- **Use with care:** Some changes can affect system boot or mount behaviour, especially if you change a filesystem's UUID without updating references to it.