# Linux

**Linux** is an **open-source, Unix-like operating system (OS)** that manages hardware and software on a computer. It is widely used for **servers, development, security, networking, and personal computing**.

- **Created by**: Linus Torvalds in **1991**
- **Kernel Type**: Monolithic (Linux Kernel)
- **License**: Open-source (GNU GPL)

**Key characteristics**

**1. Open Source**

- The source code of Linux is freely available for anyone to view, modify, and distribute.
- It follows the GNU General Public License (GPL).

**2. Multi-User System**

- Multiple users can access system resources simultaneously without interfering with each other.

**3. Multitasking**

- Supports multiple processes running at the same time efficiently.

**4. Security and Permissions**

- Built-in user authentication and permission-based access control.
- Supports file encryption, firewalls, and security modules like SELinux.

**5. Portability**

- Runs on different hardware architectures (PCs, servers, embedded systems, IoT devices).

**Loading the OS in BIOS-MBR Boot Process**

1. **BIOS Loads MBR**
    - BIOS finds the bootable disk and loads the **MBR (Master Boot Record)** from sector 0.
2. **MBR Bootloader Execution**
    - The MBR contains a **small bootloader** that locates the active partition and loads its **Volume Boot Record (VBR)**.
3. **Second-Stage Bootloader**
    - The VBR loads a **full bootloader** (e.g., **GRUB, NTLDR, or BOOTMGR**) from the partition.
    - This bootloader provides a menu to select the OS (if multiple OSes exist).
4. **Kernel Loading**
    - The bootloader loads the **OS kernel** into memory.
    - The kernel initializes hardware, mounts the filesystem, and starts system services.
5. **User Space Initialization**
    - The system enters **protected mode**, starts essential services, and presents a login screen or desktop.

**Understanding chmod Permissions in Linux**

The chmod (**Change Mode**) command in Linux is used to change **file or directory permissions**.

---

**1. Linux File Permissions**

Every file and directory in Linux has **three types of permissions** for **three types of users**:

**User Types:**

| User Type | Symbol | Description |
|---|---|---|
| **Owner** | u | The user who owns the file. |
| **Group** | g | The group associated with the file. |
| **Others** | o | All other users. |

**Permission Types:**

| Permission | Symbol | Numeric Value | Description |
|---|---|---|---|
| **Read** | r | 4 | Allows viewing file contents. |
| **Write** | w | 2 | Allows modifying file contents. |
| **Execute** | x | 1 | Allows executing files or accessing directories. |

---

**2. Viewing File Permissions**

To check file permissions, use:

sh

CopyEdit

ls -l file.txt

**Example Output:**

sh

CopyEdit

-rwxr--r-- 1 user group 1234 Jan 23 10:00 file.txt

- -rwxr--r--
    - - (First - means it's a file; d for directories)
    - rwx (Owner: Read, Write, Execute)
    - r-- (Group: Read only)

- r-- (Others: Read only)

---

**3. Changing Permissions Using chmod**

There are **two methods** to change permissions:

**A) Numeric (Octal) Method**

Each permission type has a numeric value:

- r = 4
- w = 2
- x = 1

Permissions are calculated by adding the values:

| Permission | Value |
|---|---|
| rwx | 4+2+1 = 7 |
| rw- | 4+2 = 6 |
| r-x | 4+1 = 5 |
| r-- | 4 |
| --- | 0 |

**Examples:**

sh

CopyEdit

```
chmod 777 file.txt   # Everyone gets full permissions (rwxrwxrwx)
chmod 755 file.txt   # Owner: rwx, Group: r-x, Others: r-x
chmod 644 file.txt   # Owner: rw-, Group: r--, Others: r--
chmod 600 file.txt   # Owner: rw-, Group: ---, Others: ---
chmod 400 file.txt   # Owner: r--, Group: ---, Others: ---
```

---

**B) Symbolic Method**

The symbolic method uses **letters** (u, g, o, a) to modify permissions.

- u (User/Owner)
- g (Group)
- o (Others)
- a (All: u + g + o)

**Adding (+), Removing (-), or Setting (=) permissions**

sh

CopyEdit

```
chmod u+x file.txt    # Add execute permission for the owner
chmod g-w file.txt    # Remove write permission for group
chmod o=r file.txt   # Set others to read-only
chmod a+rw file.txt  # Give read & write to everyone
```

---

**4. Changing Directory Permissions**

For directories, the x permission allows users to **enter** the directory.

sh

CopyEdit

```
chmod 755 mydir    # Owner: rwx, Group: r-x, Others: r-x
chmod 700 mydir    # Only owner can access
chmod 777 mydir    # Everyone can access (not recommended)
```

**Using -R for Recursive Permission Change**

To apply permissions to all files & subdirectories:

sh

CopyEdit

```
chmod -R 755 mydir
```

---

**5. Special Permissions (SUID, SGID, Sticky Bit)**

These special permissions add **additional security controls**.

| Type | Symbol | Numeric Value | Description |
|---|---|---|---|
| **Set User ID (SUID)** | s | 4xxx | Runs file as the owner (not caller). |
| **Set Group ID (SGID)** | s | 2xxx | Files inherit the group from parent. |
| **Sticky Bit** | t | 1xxx | Users can only delete their own files in a shared directory. |

**Example:**

sh

CopyEdit

```
chmod 4755 myscript.sh   # SUID enabled (rwsr-xr-x)
chmod 2755 mydir       # SGID enabled (rwxr-sr-x)
chmod 1777 /tmp        # Sticky bit set (rwxrwxrwt)
```

**1. Root Directory (/)**

**What is /?**

The **root directory (/) is the highest-level directory** in Linux. All files and directories originate from here.

**Contents of /**

Running ls / will list essential system directories:

sh

CopyEdit

ls /

**Common Directories in /:**

| Directory | Purpose |
|---|---|
| /bin | Essential system binaries (e.g., ls, cat, echo) |
| /boot | Boot-related files (kernel, GRUB) |
| /dev | Device files (USB, disks, etc.) |
| /etc | System configuration files |
| /home | User home directories |
| /lib | System libraries |
| /media | Removable media (USB, CD) |
| /mnt | Temporary mount point |
| /opt | Optional software packages |
| /proc | Virtual filesystem for system processes |
| /root | Root user's home directory |
| /sbin | System binaries for administrators |
| /tmp | Temporary files |
| /usr | User utilities, applications, and libraries |
| /var | Variable data (logs, cache, spool) |

**Navigation Example:**

sh

CopyEdit

cd /

ls

This moves you to the root directory and lists its contents.

---

**2. Current Directory (.) and Parent Directory (..)**

**What is .?**

- . (dot) refers to the **current directory**.
- Running a command like ./script.sh executes a script in the current directory.

**What is ..?**

- .. (double dot) refers to the **parent directory** (one level up in the hierarchy).
- Example: If you're in /home/user/Documents, running cd .. moves you to /home/user.

**Examples:**

sh

CopyEdit

pwd  # Shows current directory (e.g., /home/user)

cd .  # Stays in the same directory

cd ..  # Moves one level up (e.g., from /home/user to /home)

ls ..  # Lists contents of the parent directory

---

**3. Changing Directories (cd)**

The cd (**change directory**) command is used to navigate the filesystem.

**A) cd / – Move to the Root Directory**

sh

CopyEdit

cd /

pwd  # Output: /

- Takes you to the **root directory (/)**, the top-level directory of the Linux system.

---

**B) cd /home/user – Navigate to a Specific Directory**cd /home/user

| | |
|---|---|
| cd / | Moves to the root directory (/) |
| cd . | Stays in the current directory |
| cd .. | Moves one level up |
| cd ~ | Moves to the home directory (/home/user) |
| cd - | Switches to the previous directory |

| | |
|---|---|
| cd /home/user | Moves to /home/user |
| cd ../Documents | Moves up one level, then to Documents |

pwd  # Output: /home/user

- Moves directly to the **home directory of a user** (/home/user).

---

**C) cd ~ – Go to the Home Directory**

cd ~

pwd  # Output: /home/user

- The ~ (tilde) represents the **current user's home directory** (/home/user).

---

**D) cd - – Switch to the Previous Directory**

cd /var/log

cd /home/user

cd -  # Returns to /var/log

- This switches back to the last directory you were in.

---

**4. Relative vs. Absolute Paths**

There are **two ways** to specify a directory path:

**A) Absolute Path**

- Starts from the **root (/)**.
- Always **full path**.
- **Example:**

| | |
|---|---|
| rm file.txt | Remove a file |
| rm -i file.txt | Remove a file with confirmation |
| rm -f file.txt | Force remove a file |
| rmdir mydir | Remove an empty directory |
| rm -r mydir | Remove a non-empty directory |
| rm -rf mydir | Force remove a directory and its contents |
| rm *.log | Remove all .log files |
| rm -r dirname/* | Remove all files inside a directory but keep the directory |

1)WSL -WSL allows running wsl on windows

*To check version - wsl -l -v

*wsl –install

*Login to ubuntu – wsl -d ubuntu -u root

* Shutdown WSL - wsl --shutdown

*Check WSL status: wsl --status

*Update WSL: wsl –update

2) date

Custom Format (YYYY-MM-DD HH:MM:SS)

date +"%Y-%m-%d %H:%M:%S"

date –-set “yyyy-mm-dd”

sudo date -s=”2020-06-07 12:30:15”

3)wild character - $

ls * -all files

ls p* - start with p

cd $home – take in to home directory

4) Editor commands

1. Basic Modes

- Normal Mode (for navigation and commands): Press Esc to enter.

- Insert Mode (for editing): Press i to enter.
- Command Mode (for saving, exiting, etc.): Press : to enter.

---

2. Navigation in Normal Mode

- Move Cursor
    - Arrow keys: Move cursor up, down, left, right.
    - h: Left.
    - j: Down.
    - k: Up.
    - l: Right.
- Move by Words
    - w: Move to the start of the next word.
    - e: Move to the end of the current/next word.
    - b: Move to the beginning of the previous word.
- Move by Line
    - 0: Move to the beginning of the line.
    - $: Move to the end of the line.
    - gg: Move to the top of the file.
    - G: Move to the end of the file.

---

3. Editing Commands

- Insert Text
    - i: Insert before the cursor.
    - I: Insert at the beginning of the line.
    - a: Append after the cursor.
    - A: Append at the end of the line.
    - o: Open a new line below.
    - O: Open a new line above.
- Delete Text
    - x: Delete character under the cursor.
    - dw: Delete a word.
    - dd: Delete the entire line.

    - d$: Delete from the cursor to the end of the line.
- Undo/Redo
    - u: Undo the last action.
    - Ctrl + r: Redo the undone action.
- Copy and Paste
    - yy: Yank (copy) a line.
    - p: Paste after the cursor.
    - P: Paste before the cursor.

---

4. Searching & Replacing

- Search for a Word
    - /word: Search for "word" forward.
    - ?word: Search for "word" backward.
    - n: Move to the next match.
    - N: Move to the previous match.
- Replace Text
    - :s/old/new/: Replace first occurrence in the current line.
    - :s/old/new/g: Replace all occurrences in the current line.
    - :%s/old/new/g: Replace all occurrences in the entire file.

---

5. Saving and Exiting

- Save File
    - :w: Save the file.
    - :w filename: Save the file with a new name.
- Quit Vim
    - :q: Quit (if no changes are made).
    - :q!: Quit without saving changes.
    - :wq or :x: Save and quit.
    - ZZ: Save and quit (without :).

---

6. Other Useful Commands

- Show Line Numbers

- :set number: Enable line numbers.
- :set nonumber: Disable line numbers.

- Open a File
  - :e filename: Open a file.
  - :sp filename: Open a file in a split window.

5)cal -Shows the calendar for the current month.

cal 12 2025-Shows the calendar for December 2025.

ncal – shows calendar in vertical format.

**6)ps** -Shows processes for the current shell session.

ps aux: List all processes running on the system, including other users' processes.

a: Shows processes for all users.

u: Displays processes with user-oriented output.

x: Shows processes not attached to a terminal.

ps -ef: Another common option to show all processes with detailed information (like PID, PPID, etc.).

ps -e: Show all processes running on the system

ps -f: Display full format with additional details like parent PID.

ps aux | grep <process_name>: Find a specific process by name

ps -e | grep <process_name>

top: Continuously monitor processes and resource usage.

7)cat - Display the content of a file - cat filename

cat > filename - Create a new file or overwrite an existing file

cat >> filename - Append content to a file

touch filename - Create a new empty file

**8)cp**

cp source_file destination_file - Copy a file to another location

cp source_file /path/to/destination/ - Copy a file to a directory

cp -i source_file destination_file - Use the -i option to ask for confirmation before overwriting

cp -f source_file destination_file – force copying

cp -v source_file destination_file - Use the -v option for verbose output:

cp -uv source_file destination_file/cp -u -v source_file destination_file - This will copy the file only if the source file is newer than the destination or if the destination doesn't exist.

cp -r: Copy directories recursively

cp *.txt /path/to/destination/ - Copy all .txt files from the current directory

**9)mv**

mv source_file /path/to/destination/ - move a file to different directory

mv file.txt /home/user/Documents/

mv file1 file2 file3 /path/to/destination/

mv -r source_directory /path/to/destination/ - option to move a directory with its contents

mv -i source_file destination_file - -i (interactive) option to confirm before overwriting an existing file.

mv -f source_file destination_file

mv *.txt /path/to/destination/

mv -iv source_file destination_file -Interactive move with verbose output:

mv -fv source_file destination_file - Force move with verbose output:

**10)find**

find /path/to/search -name "filename" – find files by name

find /home/user/Documents -name "*.txt" - This searches for all .txt files in the /home/user/Documents directory.

Find files by type
Use -type to specify the type of file:

- f: Regular files.
- d: Directories.

find /path/to/search -type f

 Use -size to find files by size:

- Example (finding files greater than 1MB):

find /path/to/search -size +1M

 Find files by modification time
Use -mtime to find files modified within the last X days.

find /path/to/search -mtime -7

This searches for files modified in the last 7 days.

 Find files and execute a command
You can use -exec to perform an action on the found files.
Example:

find /path/to/search -name "*.log" -exec rm {} \;

**11)sort**

 Sort lines of a file in ascending order

sort filename

 Sort lines of a file in descending order
Use the -r option for reverse (descending) order:

sort -r filename

 Sort numbers in a file (numerical sort)
Use the -n option to sort numbers numerically (instead of lexicographically):

sort -n filename

 Sort by a specific column (e.g., column 2)
Use the -k option to specify a column:

sort -k2 filename

**12)cut**

The cut command is used to extract specific parts of a line from a file or output, based on delimiters, byte positions, or fields.

cut -d ',' -f 2 file.csv - This extracts the 2nd column from a comma-separated file.

cut -d ':' -f 1,3 /etc/passwd - This extracts the 1st and 3rd fields from /etc/passwd, which is colon-separated.

cut -d ' ' -f 2-4 file.txt - This extracts fields 2 to 4 from a space-separated file.

**13) search and replace**

:s/all/many/g

head filename

head -n 20 filename

head -c 50 filename

tail filename

tail -n 20 filename

tail -c 50 filename

tail -f logfile.log – keep showing updates

14)kill

kill PID

kill -9 PID

kill -1 PID

15)tar

tar -cvf archive.tar file.txt – Archieve a single file

tar -cvf archive.tar file1.txt file2.txt – archieve multiple files

tar -xvf archive.tar – Extract a archieve file

tar -xvf archive.tar -C /destination/path/ - Extract to a specific directory

tar -xvf archive.tar file.txt – Extract from a specific file

tar -czvf archive.tar.gz my_folder/ - Create a compressed .tar.gz archive:

tar -xzvf archive.tar.gz - Extract a .tar.gz archive

16)df -hT -

df -h  Human-readable disk usage (without filesystem types).

df -T  Show filesystem types (without human-readable format).

du -sh  /path  Show disk usage for a specific directory.

du -h – Diskspace  Usage within Folder

* ifconfig (**interface configuration**) command is used to view and manage network interfaces in Linux

* ipconfig command is used in Windows to display and manage network configuration

**17)netstat** - The netstat (network statistics) command is used to display active network connections

-a  - show all active connections

-t  - show only tcp connections

-u -show only udp connections

**18)wget**

wget command is a network downloader used to retrieve files from the web using HTTP, HTTPS, and FTP protocols.

wget https://example.com/file.zip

wget -O newfile.zip https://example.com/file.zip - will save the file and able to show output.

**19)curl** -  curl command in Linux is a powerful tool for **transferring data to or from a server**. It supports a variety of protocols, such as HTTP, HTTPS, FTP, FTPS, and many others.

curl -O https://example.com/file.zip

**19)shutdown -** sudo shutdown now

sudo shutdown +10 – shut down the system at a specific time

sudo shutdown -r now – reboot the system

**20)uname -a** - Kernel version, machine architecture, OS name, etc.

**21)who** - Displays who is currently logged into the system

**22)whoami** - Shows the current logged-in user.

**23) alias** - Creates shortcuts for longer commands

alias ll='ls -l'

unalias ll – removes an alias

**24)zip**

Compresses files into a .zip archive.

zip archive.zip file1.txt file2.txt

**25)unzip**

Extracts files from a .zip archive**.**

unzip archive.zip

**26) time** - Measures the time taken by a command to execute.

**27)locate** - Finds files by searching through a prebuilt index (requires the updatedb command to be run periodically).

Locate -I filename

**28)cmp**- Compares two files byte by byte and reports the first difference.

cmp file1.txt file2.txt

**29)diff** - Compares two files line by line and shows the differences

diff file1.txt file2.txt

**30)wc** - Counts words, lines, characters, and bytes in a file.

wc -l file.txt\

**31)tr** - Translates or deletes characters from input.

echo "hello" | tr 'a-z' 'A-Z' or echo "hello" | tr '[:lower:]' '[:upper:]'

echo "hello123" | tr -d '0-9' – delete characters

echo "hello123" | tr -d 'aeiouAEIOU'

echo "hello123" | tr -d '[:space:]'

**soft link vs hard link**

A **soft link** is like a **shortcut** to the original file.

**Key Features:**

- Points to the **file name** (not the actual data).
- If the **original file is deleted**, the soft link becomes **broken**.
- Can link to files on **different partitions or file systems**.
- Can point to **directories**.

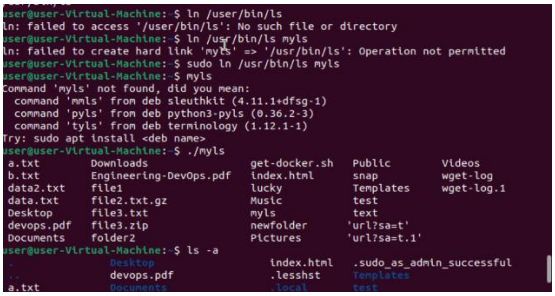
Creating soft link - ln -s original_file softlink_name - ln -s file.txt soft_link.txt

**hard link -** is an exact **copy** of the original file, but it shares the **same inode number**.

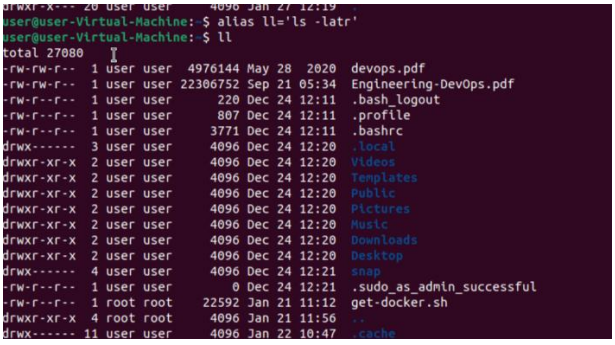
**Key Features:**

- Points to the **same data blocks** as the original file.
- Even if the **original file is deleted**, the hard link still works.
- Cannot link to files on **different partitions**.
- Cannot link to **directories**.

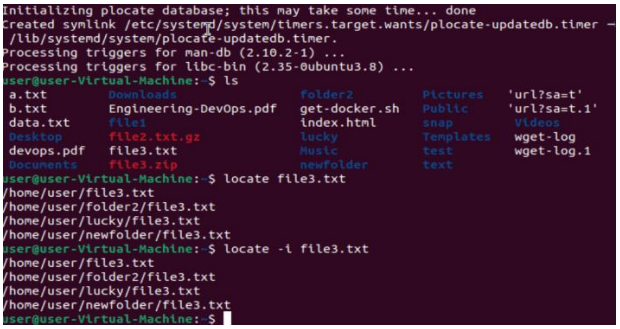
Creating hard link - ln original_file hardlink_name



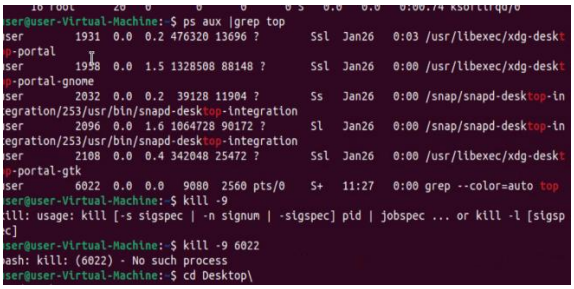

alias



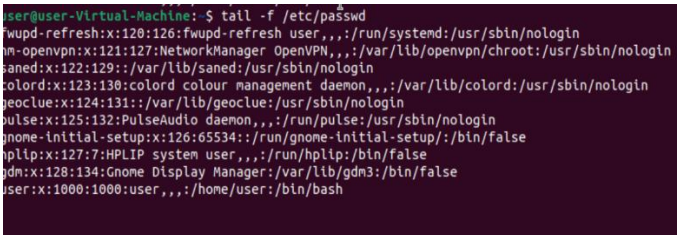

locate



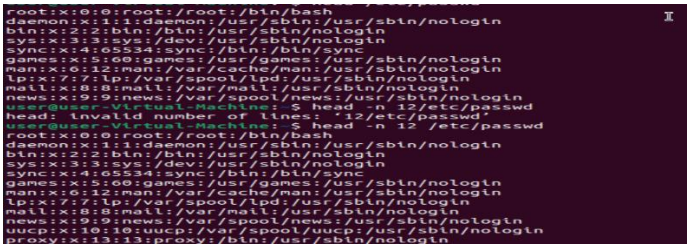

ps




tail

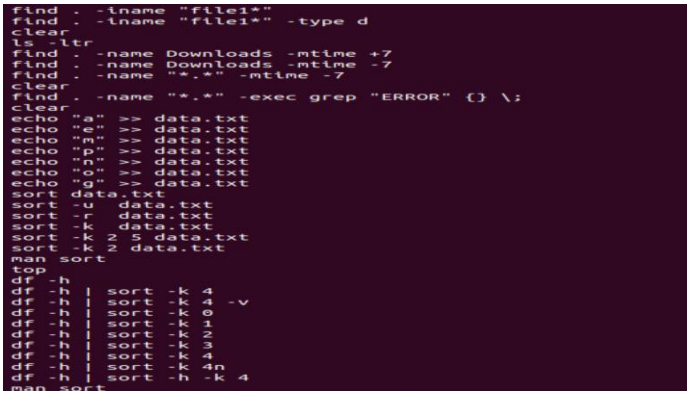



head




cut


Find,sort,df-h



```
find . -iname "file1*"
find . -iname "flle1*" -type d
clear
ls -ltr
find . -name Downloads -mtime +7
find . -name Downloads -mtime -7
find . -name "*.*" -mtime -7
clear
find . -name "*.*" -exec grep "ERROR" {} \;
clear
echo "a" >> data.txt
echo "e" >> data.txt
echo "m" >> data.txt
echo "p" >> data.txt
echo "n" >> data.txt
echo "o" >> data.txt
echo "g" >> data.txt
sort data.txt
sort -u  data.txt
sort -r  data.txt
sort -k  data.txt
sort -k 2 5 data.txt
sort -k 2 data.txt
man sort
top
df -h
df -h | sort -k 4
df -h | sort -k 4 -v
df -h | sort -k 0
df -h | sort -k 1
df -h | sort -k 2
df -h | sort -k 3
df -h | sort -k 4
df -h | sort -k 4n
df -h | sort -h -k 4
man sort
```