# LINUX

## Operating System (OS)

Is a software Program that manages a computer's resources. It acts as an interface between the user and the computer's hardware

## OS work

- The OS allocates resources like memory, CPU, and storage among other programs.
- It manages application programs
- It handles input and output devices
- It manages network connections
- It handles errors

## Type of OS

- **Multitasking OS:** Allows Multiple Programs to run simultaneously.
- **Real-time Operating System (RTOS):** Switches between task quickly to give the impression that multiple programs are running at once.
- **Linux OS:** Open-source OS that is free and customizable.
- **macOS:** Apple's desktop OS for Mac computers.
- **IOS:** Apple's OS for iphone, ipad and other apple mobile devices



## OS stored

The OS is stored on the hard disk, but it is copied into RAM when the computer starts up to speed up the process.

## Linux Architecture

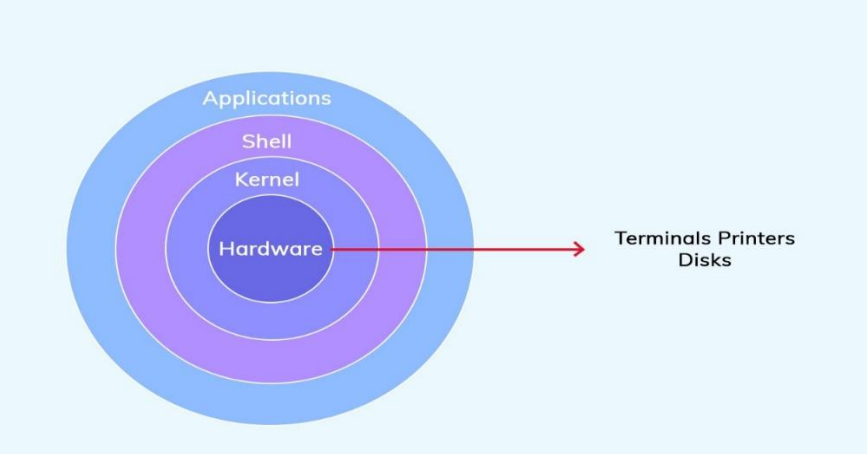

**Fig 1: Linux Architecture**

**What is Linux?**

- Linux is an open-source kernel developed by Linus Torvalds.
- It manages hardware and software communication in computers.
- Uses API (Application Programming Interface) to interact with users.

**Features of Linux**

- **Multitasking** – Runs multiple processes simultaneously.
- **User Management** – Supports multiple users with restricted access.
- **Multiple Desktop Environments** – Includes GNOME, KDE, Cinnamon, etc.
- **Open Source** – Anyone can modify and contribute.
- **Hierarchical File System** – Uses a tree structure for efficient file management.
- **Multithreading** – Can execute multiple tasks in the same program.

**Why Study Linux Architecture?**

- Ensures efficient and secure system usage.
- Helps in customization and contributing to Linux development.



**Linux Architecture (Components)**

1. **Applications & Utilities** – Software like browsers, games, and editors.
2. **Shell** – Command interpreter for user interaction (**Bash** is the default).
3. **API (Libraries)** – Acts as an interface for programs to interact with the OS.
4. **Kernel** – The core component that controls hardware and system operations.
    - **Monolithic Kernel** – Handles all services (e.g., Linux).
    - **Microkernel** – Minimal functions, uses external modules.
    - **Hybrid Kernel** – A mix of monolithic and microkernel (e.g., Windows, Mac).
5. **Hardware** – Includes CPU, RAM, storage, and other peripherals.
6. **Bootloader** – Loads the operating system when the computer starts.

**System Calls in Linux**

- Allows user applications to request OS services.
- Examples: **read()**, **write()**, **open()**.
- Executes critical operations in **Kernel Mode**.

- **Linux vs Windows: Key Differences**

| Feature | Linux 🐧 | Windows 🖥️ |
|---|---|---|
| **Source Code** | Open-source & customizable | Closed-source (Proprietary) |
| **Cost** | Free (Most distros) | Paid (License required) |
| **Security** | More secure, fewer viruses | More vulnerable to malware |
| **Performance** | Lightweight, efficient | Resource-heavy, slower over time |
| **User Control** | Full control over OS | Limited customization |
| **Updates** | Manual & user-controlled | Automatic (sometimes forced) |
| **Software Support** | Mostly open-source, some proprietary apps | Extensive software & gaming support |
| **Command Line** | Powerful & widely used (Bash, Zsh) | Limited use (PowerShell, CMD) |
| **File System** | Ext4, XFS, Btrfs | NTFS, FAT32 |
| **User Base** | Developers, IT professionals | General users, businesses, gamers |

**Conclusion**: **Linux** is best for developers & power users, while **Windows** is user-friendly and better for gaming & everyday tasks.

### 🖥️ Linux Commands Cheat Sheet



| Category | Command | Description | Example |
|---|---|---|---|
| **1.Hardware Information** | lscpu | Display CPU information | lscpu |
| | lsblk | Show block devices (disks) | lsblk |
| | free -h | Show free and used memory | free -h |
| | sudo dmidecode | Get detailed BIOS hardware info | sudo dmidecode |
| | df -h | Check disk space | df -h |
| | uptime | Show system uptime | uptime |
| **2. Searching Files** | find [path] -name [pattern] | Find files by name | find /home -name "file.txt" |
| | grep [pattern] [file] | Search a file for a pattern | grep "error" /var/log/syslog |
| | locate [filename] | Find files quickly using a database | locate file.txt |

| Category | Command | Description | Example |
|---|---|---|---|
| | which [command] | Find location of a command's binary | which python |
| **3.File Management** | mkdir [dir] | Create a new directory | mkdir new_folder |
| | rm [file] | Remove a file | rm old_file.txt |
| | cp [src] [dest] | Copy a file | cp file1.txt backup/ |
| | mv [src] [dest] | Move or rename a file | mv oldname.txt newname.txt |
| | touch [file] | Create a new, empty file | touch newfile.txt |
| | stat [file] | Show detailed info about a file | stat file.txt |
| | head [file] | View first 10 lines of a file | head file.txt |
| | tail [file] | View last 10 lines of a file | tail file.txt |
| | cat [file] | Display file content | cat file.txt |
| | less [file] | View file one page at a time | less file.txt |
| **4.Directory Navigation** | ls | List files and directories | ls -l |
| | cd [dir] | Change directory | cd Documents |
| | pwd | Show current directory | pwd |
| | tree | Display directories as a tree | tree |
| **5.File Compression** | tar cf [archive.tar] [file/dir] | Archive files | tar cf backup.tar Documents |
| | gzip [file] | Compress a file | gzip log.txt |
| | tar xf [archive.tar] | Extract tar archive | tar xf backup.tar |
| | unzip [file.zip] | Extract zip archive | unzip archive.zip |
| **6. File Transfer** | scp [src] [user]@[host]:[dest] | Secure file copy to remote server | scp file.txt user@server:/home |
| | wget [URL] | Download a file from a URL | wget http://example.com/file.zip |
| | rsync [src] [dest] | Synchronize directories | rsync -av /src/ /dest/ |

| Category | Command | Description | Example |
|---|---|---|---|
| **7.User Management** | whoami | Show current user | whoami |
| | passwd | Change password | passwd |
| | sudo useradd [user] | Add a new user | sudo useradd newuser |
| | sudo userdel [user] | Delete a user | sudo userdel olduser |
| | groups [user] | Show user's groups | groups amogh |
| | id [user] | Show UID, GID, and groups of user | id amogh |
| **8.Process Management** | ps | List running processes | ps aux |
| | top | Show live system processes | top |
| | htop (if installed) | Interactive process viewer | htop |
| | kill [PID] | Kill a process by PID | kill 1234 |
| | killall [name] | Kill processes by name | killall firefox |
| | bg | Resume a stopped job in the background | bg |
| | fg | Bring a background job to the foreground | fg |
| **9.System Monitoring** | df -h | Check disk space | df -h |
| | du -h [dir] | Show size of files and directories | du -h /home |
| | uptime | Show how long the system has been running | uptime |
| | vmstat | Show system performance stats | vmstat |
| **10. Networking** | ping [host] | Ping a host | ping google.com |
| | ip addr show | Show network interfaces | ip addr show |
| | netstat -tuln | Show listening ports | netstat -tuln |
| | curl [URL] | Fetch content from a URL | curl http://example.com |

| Category | Command | Description | Example |
|---|---|---|---|
| **11.File Permissions** | chmod [mode] [file] | Change file permissions | chmod 777 script.sh |
| | chown [user]:[group] [file] | Change file owner | chown user:user file.txt |
| | umask | Show default file permissions | umask |
| **12.Package Management** | sudo apt install [package] | Install a package (Debian/Ubuntu) | sudo apt install vim |
| | sudo yum install [package] | Install a package (Red Hat/CentOS) | sudo yum install nano |
| | apt-cache search [keyword] | Search for packages | apt-cache search git |
| | dpkg -l | List installed packages (Debian) | dpkg -l |
| **13.Keyboard Shortcuts** | Ctrl + C | Kill the running process | |
| | Ctrl + Z | Suspend the current process | |
| | Ctrl + R | Search command history | |
| | Ctrl + L | Clear terminal screen | |



**Linux Directory Structure**

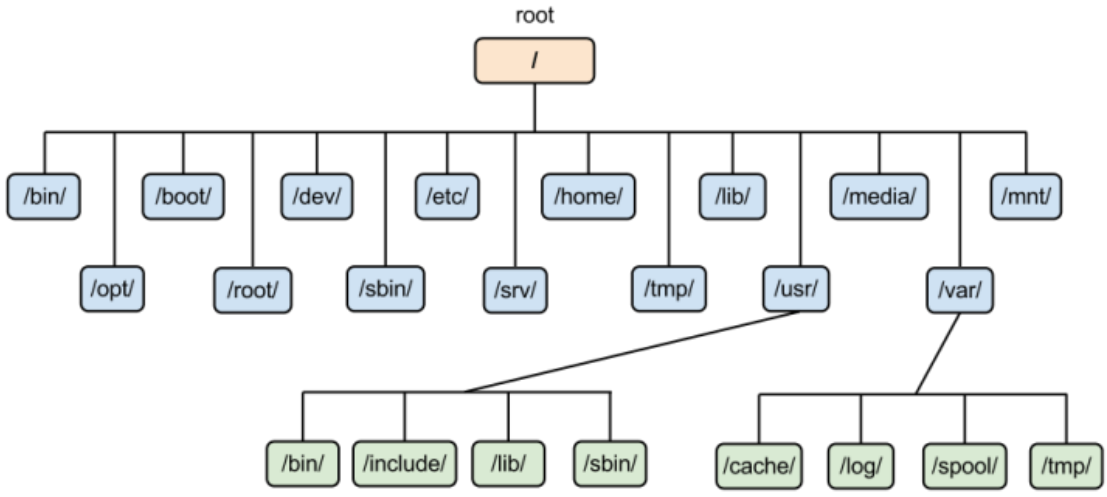


**Fig 1.1 Linux Directory Hierarchy**

Linux has a tree-like folder structure, starting from **/ (root)**. Each folder has a specific role:

| Directory | What It Does |
|---|---|
| **/ (Root)** | The starting point of everything in Linux. |
| **/bin** | Stores basic commands like ls, cp, rm, etc. (Used by all users). |
| **/sbin** | System commands like shutdown, fdisk (Used by admin/root). |
| **/home** | Stores personal files for users (e.g., /home/amogh). |
| **/root** | Home folder for the root (admin) user. |
| **/etc** | Stores system settings and configuration files. |
| **/var** | Stores logs (/var/log), cache, and temporary data. |
| **/tmp** | Temporary files (automatically deleted on reboot). |
| **/usr** | Stores software programs and libraries (/usr/bin for apps). |
| **/dev** | Contains system devices like hard drives (/dev/sda). |
| **/proc** | Virtual folder with system info (/proc/cpuinfo for CPU details). |
| **/sys** | Provides information about system hardware. |
| **/opt** | For installing extra software manually. |
| **/mnt** | Used to mount external drives (manual). |
| **/media** | Auto-mounts USB drives, CDs, etc. |
| **/boot** | Stores boot files (Linux kernel, bootloader). |
| **/lib & /lib64** | Stores shared system libraries needed to run programs. |



**Simply put:**

- **User files?** → /home
- **System settings?** → /etc
- **Installed software?** → /usr
- **Device files?** → /dev
- **Logs?** → /var/log
- **Boot-related stuff?** → /boot

**File Permissions –**

**Understanding File Permissions**

Each file has 3 types of permissions for 3 types of users:

| User Type | Description |
|---|---|
| **Owner (user)** | The user who created the file. |
| **Group** | Users in the same group as the owner. |
| **Others** | Everyone else on the system. |

**Permission Types**

| Permission | File Meaning | Directory Meaning |
|---|---|---|
| **r (Read)** | Can view the content of a file | Can list files inside the directory (ls) |
| **w (Write)** | Can modify or delete a file | Can create, rename, or delete files inside the directory |
| **x (Execute)** | Can run the file (if it's a script) | Can access (cd) the directory |

**Numeric Permission Values (Octal Mode)**

| Permission | Number |
|---|---|
| r (read) | 4 |
| w (write) | 2 |
| x (execute) | 1 |



**File Default Permissions**

| User Type | Permissions (Symbolic) | Permissions (Octal) | Breakdown |
|---|---|---|---|
| **Root User** | rw-r--r-- | 644 | User → rw- (6), Group → r-- (4), Others → r-- (4) |
| **Local User** | rw-rw-r-- | 664 | User → rw- (6), Group → rw- (6), Others → r-- (4) |

---

**Directory Default Permissions**

| User Type | Permissions (Symbolic) | Permissions (Octal) | Breakdown |
|---|---|---|---|
| **Root User** | rwxr-xr-x | 755 | User → rwx (7), Group → r-x (5), Others → r-x (5) |

| User Type | Permissions (Symbolic) | Permissions (Octal) | Breakdown |
|---|---|---|---|
| **Local User** | rwxrwxr-x | 775 | User → rwx (7), Group → rwx (7), Others → r-x (5) |

**Key Takeaways:**

- Root user files default to 644 **(rw-r--r--)**
- Local user files default to 664 **(rw-rw-r--)**
- Root user directories default to 755 **(rwxr-xr-x)**
- Local user directories default to 775 **(rwxrwxr-x)**

**Full File & Directory Permissions (Before umask)**

| User Type | Default File Permission | Default Directory Permission |
|---|---|---|
| **Root User (UID 0)** | 666 (rw-rw-rw-) | 777 (rwxrwxrwx) |
| **Local User (UID 1000+)** | 666 (rw-rw-rw-) | 777 (rwxrwxrwx) |

**Umask**

umask sets the default permissions for newly created files and directories by removing permissions from the default (666 for files, 777 for directories).



**Common umask values:**

- 022 → Files: 644 (rw-r--r--), Directories: 755 (rwxr-xr-x)
- 027 → Files: 640 (rw-r-----), Directories: 750 (rwxr-x---)
- 077 → Files: 600 (rw-------), Directories: 700 (rwx------)
- Umask for the root user is 0022
- Umask for the local user is 0002

- Root user **(#)** has full control over all files.
- Local users **($)** can only modify their own files.
- Permissions **(rwx)** control who can read, write, or execute files.
- Use chmod to change permissions & **chown** to change ownership.
- Be careful with sudo—it can override all restrictions!

**Here's a table with examples for each useradd option:**

| Option | Description | Example |
|---|---|---|
| **-b, --base-dir** | Base directory for the home directory of the new account | useradd -b /mnt/users newuser |
| **-c, --comment** | GECOS field (user info) | useradd -c "John Doe, Developer" johndoe |

| Option | Description | Example |
|---|---|---|
| **-d, --home-dir** | Specify home directory | useradd -d /home/custom johndoe |
| **-D, --defaults** | Show/change default useradd settings | useradd -D |
| **-e, --expiredate** | Set account expiration date | useradd -e 2025-12-31 johndoe |
| **-f, --inactive** | Set password inactivity period | useradd -f 10 johndoe (disable account 10 days after password expires) |
| **-g, --gid** | Set primary group ID | useradd -g developers johndoe |
| **-G, --groups** | Add user to multiple groups | useradd -G sudo,devops johndoe |
| **-h, --help** | Display help message | useradd -h |
| **-k, --skel** | Use custom skeleton directory | useradd -k /etc/custom_skel johndoe |
| **-K, --key** | Override default settings | useradd -K PASS_MAX_DAYS=90 johndoe |
| **-l, --no-log-init** | Do not add to lastlog/faillog | useradd -l johndoe |
| **-m, --create-home** | Create home directory | useradd -m johndoe |
| **-M, --no-create-home** | Do not create home directory | useradd -M johndoe |
| **-N, --no-user-group** | Do not create a group with the same name as the user | useradd -N johndoe |
| **-o, --non-unique** | Allow duplicate UIDs | useradd -o -u 1001 johndoe |
| **-p, --password** | Set encrypted password | useradd -p $(openssl passwd -crypt "password") johndoe |
| **-r, --system** | Create a system account | useradd -r systemuser |
| **-R, --root** | Set chroot directory | useradd -R /mnt/chroot johndoe |
| **-P, --prefix** | Set prefix directory for /etc/* files | useradd -P /custom/etc johndoe |
| **-s, --shell** | Set login shell | useradd -s /bin/zsh johndoe |
| **-u, --uid** | Set user ID | useradd -u 1050 johndoe |
| **-U, --user-group** | Create group with same name as user | useradd -U johndoe |
| **-Z, --selinux-user** | Set SELinux user mapping | useradd -Z staff_u johndoe |

**1. groupmod (Modify a Group)**

**Used to modify an existing group in Linux.**

| Option | Description | Example |
|---|---|---|
| **-g, --gid** | Change the group ID | groupmod -g 1050 developers (Changes GID of "developers" to 1050) |
| **-n, --new-name** | Rename a group | groupmod -n devops developers (Renames "developers" group to "devops") |

**2. usermod (Modify a User)**

**Used to modify an existing user account.**

| Option | Description | Example |
|---|---|---|
| **-c, --comment** | Change the user's comment (GECOS field) | usermod -c "Senior Developer" johndoe |
| **-d, --home** | Change the user's home directory | usermod -d /home/newhome johndoe |
| **-e, --expiredate** | Set account expiration date | usermod -e 2025-12-31 johndoe |
| **-g, --gid** | Change primary group | usermod -g devops johndoe |
| **-G, --groups** | Add user to supplementary groups | usermod -G sudo,docker johndoe |
| **-a, --append** | Append new groups instead of replacing | usermod -aG docker johndoe |
| **-l, --login** | Change username | usermod -l newjohndoe johndoe |
| **-s, --shell** | Change default shell | usermod -s /bin/zsh johndoe |
| **-L, --lock** | Lock user account | usermod -L johndoe |
| **-U, --unlock** | Unlock user account | usermod -U johndoe |

**3. userdel (Delete a User)**

**Used to delete a user account.**

| Option | Description | Example |
|---|---|---|
| **-r, --remove** | Remove user and home directory | userdel -r johndoe |
| **-f, --force** | Force remove user even if logged in | userdel -f johndoe |

**What is a Shell in Linux?**

A shell in Linux is a command-line interface (CLI) that allows users to interact with the operating system by executing commands. It acts as an intermediary between the user and the kernel.

**Types of Shells in Linux**

There are multiple shells available in Linux, each with different features and functionalities. Below is a comparison of the most common ones:

| Shell | Full Name | Path | Features & Differences |
|---|---|---|---|
| **Bash** | Bourne Again Shell | /bin/bash | Default shell in most Linux distributions, supports scripting, command history, and auto-completion. |
| **Sh** | Bourne Shell | /bin/sh | One of the oldest shells, lacks interactive features like command history and tab completion. |
| **Zsh** | Z Shell | /bin/zsh | Similar to Bash but with better customization, auto-correction, and powerful scripting capabilities. |
| **Ksh** | Korn Shell | /bin/ksh | Faster than Bash, supports scripting, and has advanced programming features. |
| **Tcsh** | TENEX C Shell | /bin/tcsh | An improved version of C Shell (csh), offers better scripting and command-line editing. |
| **Csh** | C Shell | /bin/csh | Syntax similar to the C programming language, useful for C programmers but less common now. |
| **Fish** | Friendly Interactive Shell | /usr/bin/fish | User-friendly shell with autosuggestions, syntax highlighting, and simplified scripting. |
| **Dash** | Debian Almquist Shell | /bin/dash | Lightweight and faster alternative to Bash, used for system scripts in Debian-based systems. |



**Difference Between Popular Shells**

**1. Bash vs. Zsh**

| Feature | Bash | Zsh |
|---|---|---|
| **Default on most Linux** | ✅ Yes | ❌ No |
| **Auto-completion** | ✅ Yes | ✅ Yes (Better) |
| **Auto-correction** | ❌ No | ✅ Yes |
| **Plugin Support** | ❌ Limited | ✅ Extensive (Oh My Zsh) |
| **Syntax Highlighting** | ❌ No | ✅ Yes |
| **Performance** | Medium | Faster than Bash |

**2. Bash vs. Fish**

| Feature | Bash | Fish |
|---|---|---|
| **POSIX Compatibility** | ✅ Yes | ❌ No |
| **User-Friendly** | ❌ No | ✅ Yes |
| **Syntax Highlighting** | ❌ No | ✅ Yes |
| **Auto-suggestions** | ❌ No | ✅ Yes |

**How to Check and Change Your Shell in Linux**

| Command | Description | Example Output |
|---|---|---|
| echo $SHELL | Check the current shell | /bin/bash |
| cat /etc/shells | List all available shells | /bin/bash, /bin/zsh, /bin/fish, etc. |
| chsh -s /path/to/shell | Change the default shell | chsh -s /bin/zsh |
| sudo chsh -s /bin/zsh username | Change shell for another user | (Requires sudo) |
| which bash | Find shell path | /bin/bash |
| which zsh | Find Zsh path | /bin/zsh |



**Short Notes**

**Shell**: A command-line interface for interacting with the Linux OS.

**Popular Shells**: Bash, Zsh, Fish, Ksh, Csh, Dash, Tcsh.

**Check Current Shell**: Use echo $SHELL.

**List Installed Shells**: Use cat /etc/shells.

**Change Shell**: Use chsh -s /path/to/shell (Log out & log in for changes).

**Install New Shell (if not installed)**:

| Shell | Debian/Ubuntu | RHEL/CentOS | Arch Linux |
|---|---|---|---|
| **Zsh** | sudo apt install zsh | sudo yum install zsh | sudo pacman -S zsh |
| **Fish** | sudo apt install fish | sudo yum install fish | sudo pacman -S fish |
| **Ksh** | sudo apt install ksh | sudo yum install ksh | sudo pacman -S ksh |
| **Tcsh** | sudo apt install tcsh | sudo yum install tcsh | sudo pacman -S tcsh |
| **Dash** | sudo apt install dash | sudo yum install dash | sudo pacman -S dash |

**Short Notes**

**Install a Shell:** Use the package manager for your distro.

**Verify Installation:** Run which shell_name (e.g., which zsh).

**Change Default Shell:**chsh -s /bin/zsh  # Replace with your shell path

**List Available Shells:** cat /etc/shells.

**Apply Changes:** Log out & log back in.

**Echo Command in Linux**

The echo command is used to display text or variables in the terminal. It is commonly used in shell scripts and command-line operations.

**Syntax:**

echo [options] [string]

**Examples:**

echo "Hello, World!"  **# Prints Hello, World!**

echo $HOME  **# Displays the value of the HOME variable**

echo -e "Line1\nLine2"  **# Enables interpretation of escape sequences**



**Creating a File Using Echo**

**With Override (will overwrite existing file):**

To overwrite a file with new content, use the > operator.

**Without Override (append to file):**

To append text to an existing file, use the >> operator.

| Method | Syntax | Example |
|---|---|---|
| **With Override** | echo "text" > filename | echo "Hello, World!" > file.txt |
| **Without Override** | echo "text" >> filename | echo "New text" >> file.txt |

**Examples:**

- **Override File:**

echo "This is new content" > file.txt  # Overrides content of file.txt

- **Without Override (Append):**

echo "Additional content" >> file.txt  # Appends to file.txt

**systemctl Services in Linux**

systemctl is a command-line utility used to manage systemd services in Linux. It can start, stop, enable, disable, restart, and check the status of services.

**Common Commands:**

systemctl start service_name  # Start a service

systemctl stop service_name  # Stop a service

systemctl restart service_name  # Restart a service

systemctl status service_name  # Check service status

systemctl enable service_name  # Enable service at boot

systemctl disable service_name  # Disable service from boot

**Using . and * in Linux**

. (Dot): Represents the current directory.

**Example:**

cd .  # Stays in the current directory

* (Asterisk): Wildcard for matching multiple files.

**Example:**

ls *.txt  # Lists all .txt files in the current directory



# User and Group Management in Linux

Linux is a multi-user system that allows multiple users to log in and use system resources. Users and groups help in managing permissions and access control effectively.

**1. User Management**

Users in Linux can be regular users, system users, or superusers (root).

**1.1 Types of Users in Linux**

1. Root User (Superuser): Has full control over the system.
2. System Users: Used by system processes and services.
3. Regular Users: Created for human users.

**1.2 Files That Store User Information**

- /etc/passwd → Stores user account information.
- /etc/shadow → Stores user passwords in an encrypted format.
- /etc/group → Stores group details.

**1.3 Creating a New User**

**Basic User Creation**

sudo useradd amogh

- Adds a new user but does not create a home directory.

**Creating a User with a Home Directory**

sudo useradd -m amogh

- The -m option ensures a home directory (/home/amogh) is created.

**Create a User with a Specific Shell**

sudo useradd -s /bin/bash amogh

- Assigns Bash as the default shell.

**Set Password for the User**

sudo passwd amogh

- Prompts to enter a password.



**View User Details**

cat /etc/passwd | grep amogh

**1.4 Modifying User Accounts**

**Change a Username**

sudo usermod -l newname oldname

**Change Home Directory**

sudo usermod -d /new/home/dir amogh

**Change Default Shell**

sudo usermod -s /bin/zsh amogh

**Lock and Unlock User Accounts**

- Lock User:
- sudo passwd -l amogh
- Unlock User:
- sudo passwd -u Amogh

**1.5 Deleting Users**

Delete User Without Removing Home Directory

sudo userdel amogh

Delete User and Remove Home Directory

sudo userdel -r Amogh

**2. Group Management**

A group is a collection of users who share the same permissions.

**2.1 Types of Groups**

1. Primary Group: The default group assigned to a user.
2. Supplementary Group: Additional groups a user belongs to. (Secondary Group)

**2.2 Creating a Group**

sudo groupadd devops

**2.3 Adding a User to a Group**

sudo usermod -aG devops amogh

- The -aG option adds the user without removing from existing groups.

**2.4 View Groups of a User**



groups amogh

id amogh

**2.5 Change a User's Primary Group**

sudo usermod -g devops amogh

**2.6 Remove a User from a Group**

sudo gpasswd -d amogh devops

**2.7 Delete a Group**

sudo groupdel devops

**3. Managing Superuser and Sudo Access**

The sudo command allows regular users to execute commands as root.

**3.1 Add a User to the Sudo Group**

sudo usermod -aG sudo amogh

- Grants sudo access to amogh.

**3.2 Edit Sudo Permissions**

sudo visudo

- Add this line for password-less sudo access:
- amogh ALL=(ALL) NOPASSWD:ALL

**4. Switching Between Users**

**4.1 Switch to Another User**

su - amogh

- The - ensures the user's environment is loaded.

**4.2 Run Commands as Another User**

sudo -u amogh ls /home/Amogh

**5. Viewing User and Group Information**

**5.1 List All Users**

cut -d: -f1 /etc/passwd

**5.2 List All Groups**

cut -d: -f1 /etc/group

**5.3 Find User ID (UID) and Group ID (GID)**



id amogh

- Example Output:
- uid=1001(amogh) gid=1001(amogh) groups=1001(amogh),27(sudo)

**6. Managing Password Policies**

**6.1 Set Password Expiry for a User**

sudo chage -M 30 amogh

- Forces password change every 30 days.

**6.2 Force Immediate Password Change**

sudo passwd -e amogh

**7. Real-World Scenarios**

**Scenario 1: Create a User with Sudo Privileges**

sudo useradd -m -s /bin/bash amogh

sudo passwd amogh

sudo usermod -aG sudo amogh

**Scenario 2: Restrict a User to a Specific Group**

sudo usermod -g developers amogh

sudo usermod -aG docker amogh

**Scenario 3: Lock an Inactive User**

sudo usermod --expiredate 1 amogh

- This disables the account.

**Conclusion**

- Linux user and group management is essential for security and access control.
- You now know how to create, modify, and manage users and groups effectively.
- Using sudo, permissions, and best practices ensures a secure Linux system.

**File Structure (/etc/shadow Format)**

Each line represents a user account and consists of multiple fields separated by :

arduino

username:password:lastchange:min:max:warn:inactive:expire:reserved

| Field Number | Field Name | Description |
|---|---|---|
| 1 | **Username** | The login name of the user. |
| 2 | **Password** | The hashed password (or !/* if disabled). |
| 3 | **Last Password Change** | Days since last password change (from Unix epoch). |
| 4 | **Min Days** | Minimum days before changing the password. |
| 5 | **Max Days** | Maximum days a password is valid before requiring a change. |
| 6 | **Warning Days** | Days before expiry when the user is warned. |
| 7 | **Inactive Days** | Days after expiry before disabling the account. |
| 8 | **Account Expiry** | Absolute date when the account is disabled. |
| 9 | **Reserved** | Reserved for future use. |



🔐 **Example Entry**

perl

CopyEdit

amogh:$6$gJH84KL/$sdfhJkd...:19345:0:99999:7:::

- amogh → Username
- $6$gJH84KL/$sdfhJkd... → SHA-512 encrypted password
- 19345 → Last password change date
- 0 → No minimum days required to change the password
- 99999 → Password never expires
- 7 → User gets a warning 7 days before expiry

### 🔎 Security & Permissions

- /etc/shadow is **readable only by root**:

ls -l /etc/shadow

**Output:**

-r-------- 1 root root 1234 Mar 08 12:34 /etc/shadow

- To read it, you must be root:

sudo cat /etc/shadow

### 🔧 Modifying /etc/shadow



- Change password expiration settings:

sudo chage -M 90 -W 7 amogh  # Password expires in 90 days, warns 7 days before

- Lock user account:

sudo passwd -l amogh

- Unlock user account:

sudo passwd -u amogh

### ⚠️ Warning

- Do **not** edit /etc/shadow manually unless necessary.
- If you corrupt it, users may **lose access** to their accounts.
- Always **use passwd, chage, or usermod commands** to modify it safely.

**Vim in Linux: A Comprehensive Guide**

Vim (Vi IMproved) is a powerful, highly customizable text editor used in Linux and UNIX systems. It is an enhanced version of vi, featuring syntax highlighting, multiple buffers, extensive plugin support, and much more.

**1. Installing Vim in Linux**

**Check if Vim is Installed**

Before installing, check whether Vim is already installed by running:

bash

vim --version

If Vim is installed, you will see version details. If not, install it using the appropriate package manager for your distribution.

**Installation Commands**

- **Debian/Ubuntu**

sudo apt update

sudo apt install vim -y

- **RHEL/CentOS/Fedora**



sudo yum install vim -y  # For older versions

sudo dnf install vim -y  # For Fedora or newer RHEL/CentOS

- **Arch Linux**

sudo pacman -S vim

- **macOS (via Homebrew)**

brew install vim

**2. Starting Vim**

To open a file in Vim, use:

vim filename

If the file does not exist, Vim will create a new one.

- **Opening multiple files**

vim file1 file2 file3

    - Use :n to switch to the next file
    - Use :prev to go back

### 3. Vim Modes

Vim operates in multiple modes:

| Mode | Description |
|---|---|
| **Normal** | Default mode, used for navigation and commands |
| **Insert** | Used for text editing (like a regular text editor) |
| **Visual** | Used for selecting text for copying, cutting, etc. |
| **Command** | Used for executing commands (saving, quitting, searching) |

**Switching Between Modes**

- **Insert Mode:** Press i, a, o, etc.
- **Normal Mode:** Press Esc (from any mode)
- **Visual Mode:** Press v (for character selection) or V (for line selection)
- **Command Mode:** Press : in Normal mode

### 4. Basic File Operations

| Command | Description |
|---|---|
| :q | Quit Vim (only if no unsaved changes) |
| :q! | Quit without saving |
| :w | Save the file |
| :wq or ZZ | Save and exit |
| :x | Save and exit (only if changes were made) |
| :w filename | Save file as filename |



### 5. Editing Text in Vim

**Insert Mode Commands**

- 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

**Deleting Text**

- x → Delete a character
- dd → Delete the current line
- D → Delete from cursor to end of line
- d$ → Delete from cursor to end of line
- d0 → Delete from cursor to start of line

**Copy, Paste, and Cut**

- yy → Copy (yank) the current line
- y$ → Copy to end of the line
- p → Paste after cursor
- P → Paste before cursor
- dd → Cut the current line

**Undo and Redo**

- u → Undo last change
- Ctrl + r → Redo



**6. Moving and Navigating in Vim**

**Arrow Key Alternatives**

- h → Left
- l → Right
- j → Down
- k → Up

**Jumping Within a File**

| Command | Action |
|---|---|
| 0 | Move to the beginning of the line |
| ^ | Move to the first non-whitespace character |
| $ | Move to the end of the line |
| gg | Jump to the beginning of the file |
| G | Jump to the end of the file |
| :n | Jump to line n |

**7. Searching and Replacing**

**Search Commands**

- /word → Search forward for "word"
- ?word → Search backward for "word"
- n → Move to the next match
- N → Move to the previous match

**Find and Replace**

- :%s/old/new/g → Replace all occurrences of "old" with "new" in the file
- :s/old/new/ → Replace only in the current line

**8. Working with Multiple Files**

**Open and Switch Between Files**

- :e filename → Open another file
- :bn → Next buffer
- :bp → Previous buffer
- :bd → Close a buffer



**Split Window Editing**

- :split filename → Open filename in a horizontal split
- :vsplit filename → Open filename in a vertical split
- Ctrl + w + w → Switch between splits

**9. Customizing Vim**

Edit Vim's configuration file:

vim ~/.vimrc

**Example .vimrc Configuration**

vim

CopyEdit

```
syntax on       " Enable syntax highlighting
set number      " Show line numbers
set tabstop=4   " Set tab size to 4 spaces
set shiftwidth=4 " Auto-indent with 4 spaces
```

```
set expandtab   " Convert tabs to spaces
set autoindent " Maintain indentation on new lines
set hlsearch    " Highlight search results
set incsearch   " Incremental search
```

**10. Exiting Vim (If Stuck)**

If you accidentally enter Vim and can't exit, try:

1. Press Esc
2. Type :q! (quit without saving) or :wq (save and quit)
3. Press Enter

**11. Vim Plugins**

Vim supports plugins via a plugin manager like **vim-plug**.

**Installing vim-plug**

```
curl -fLo ~/.vim/autoload/plug.vim --create-dirs \
https://raw.githubusercontent.com/junegunn/vim-plug/master/plug.vim
```



**Example: Adding Plugins**

Edit ~/.vimrc and add:

```
vim
call plug#begin()
Plug 'preservim/nerdtree'   " File explorer
Plug 'vim-airline/vim-airline' " Status bar
call plug#end()
```

Then run :PlugInstall in Vim to install plugins.

# Crontab

crontab is a command-line utility used in Unix-like operating systems to schedule jobs (commands or scripts) to run automatically at specified intervals. The jobs are defined in a file called the cron table (or crontab).

**Basic Crontab Syntax:**

A crontab file consists of a series of lines, each representing a scheduled job. Each line follows this format:

* * * * * /path/to/command

- - - - -

| | | | |

| | | | +--- Day of the week (0 - 7) (Sunday = 0 or 7)

| | | +----- Month (1 - 12)

| | +------- Day of the month (1 - 31)

| +--------- Hour (0 - 23)

+----------- Minute (0 - 59)

**Example:**

30 08 * * 1 /home/user/script.sh

**This example means "run script.sh at 08:30 AM every Monday."**

**How to Use crontab:**



**Edit the crontab file:**

Use the command crontab -e to open the crontab file in your default text editor. You can then add jobs according to the syntax shown above.

**List the crontab:**

Use crontab -l to list the current user's scheduled cron jobs.

**Remove the crontab:**

Use crontab -r to remove the crontab file, which deletes all scheduled cron jobs.

View or edit a specific user's crontab (if you have sufficient permissions):

crontab -u username -e will allow you to edit the crontab for a specific user (if you're an administrator).

**Special Time Strings:**

**@reboot: Run the command once after the system starts.**

**@yearly: Equivalent to 0 0 1 1 *.**

**@monthly: Equivalent to 0 0 1 * *.**

**@weekly: Equivalent to 0 0 * * 0.**

**@daily: Equivalent to 0 0 * * *.**

**@hourly: Equivalent to 0 * * * *.**

**Example of Special Time Strings:**

@daily /home/user/daily_backup.sh

This will run the daily_backup.sh script once every day.

## Compression Method in Linux

In Linux, compression and decompression refer to the processes of reducing the size of a file (or multiple files) and expanding a compressed file, respectively. These processes are essential for saving disk space and transferring large files efficiently.

Here are some common tools and commands used for compression and decompression in Linux:

**1. tar (Tape Archive)**



Compression with tar: You can use tar to combine multiple files into a single archive. The tar command is often combined with compression tools like gzip, bzip2, and xz.

**Compressing with tar:**

```
tar -czf archive.tar.gz file1 file2 file3
# -c: Create a new archive
# -z: Compress the archive using gzip
# -f: Specify the name of the archive
# -j: Compress the archive using bzip2 (tar -cjvf archive.tar.bz2 filename)
# -J: Compress the archive using xz
```

**Decompressing with tar:**

```
tar -xzf archive.tar.gz
```

# -x: Extract the archive

# -z: Decompress the archive using gzip

# -f: Specify the name of the archive

# -j: Decompress the archive using bzip2 (tar -xjvf archive.tar.bz2 filename)

# -J: Decompress the archive using xz

**2. gzip (GNU Zip)**

gzip is a popular compression tool for single files. It is often used with tar.

**Compressing with gzip**:

**gzip file**

# This will create file.gz and remove the original file.

**Decompressing with gzip:**



**gunzip file.gz**

# This will extract the original file and remove the .gz extension.

**3. bzip2 (Burrows-Wheeler Block Sort)**

bzip2 is a compression utility that provides better compression than gzip, but it is slower.

**Compressing with bzip2:**

**bzip2 file**

# This will create file.bz2 and remove the original file.

**Decompressing with bzip2:**

bunzip2 file.bz2

# This will extract the original file and remove the .bz2 extension.

**4. xz (LZMA2 compression)**

xz is a compression utility that provides even higher compression rates than gzip and bzip2 but is slower.

Compressing with xz:

**xz file**

# This will create file.xz and remove the original file.

**Decompressing with xz:**

unxz file.xz

# This will extract the original file and remove the .xz extension.

**5. zip and unzip**

zip is commonly used to create compressed archives that can be opened on both Linux and Windows systems.

Compressing with zip:



zip archive.zip file1 file2 file3

# Creates archive.zip containing file1, file2, and file3.

**Decompressing with unzip:**

unzip archive.zip

# Extracts the contents of archive.zip.

**6. 7z (7-Zip)**

7z is a file archiver that supports high compression ratios and various formats.

**Compressing with 7z:**

7z a archive.7z file1 file2 file3

# 'a' stands for 'add', creating archive.7z with file1, file2, and file3.

**Decompressing with 7z:**

7z x archive.7z

# Extracts the contents of archive.7z.

**Summary of Commands:**

tar: Combines files and compresses them (commonly with gzip, bzip2, or xz).

gzip: Compresses individual files using the gzip format.

bzip2: Compresses individual files with better compression ratios than gzip.

xz: Compresses files with very high compression ratios, slower than gzip and bzip2.

zip: Commonly used to compress files into a .zip archive.

7z: A powerful file archiver with high compression ratios, supporting multiple formats.



## Grep in Linux (Global Regular Expression Print)

grep is a powerful command-line tool in Linux used to search for specific patterns in files or output. It stands for "global regular expression print," and it allows you to search for text using regular expressions.

Basic Syntax:

grep [options] pattern [file...]

pattern: The text or regular expression you're searching for.

file: One or more files to search within. If no files are specified, grep reads from standard input.

**Commonly Used Options:**

-i: Ignore case (case-insensitive search).

-r or -R: Recursively search directories.

-v: Invert the match, showing lines that do not contain the pattern.

-l: Show only the filenames of files that match the pattern.

-n: Show line numbers with the matching lines.

-w: Match whole words only (i.e., the pattern must be a complete word).

-c: Count the number of matching lines.

-H: Show the filename with the matching line.

-o: Show only the matched parts of the line.

**Examples:**

**Search for a string in a file:**

grep "hello" filename.txt

**Search for a pattern case-insensitively:**

grep -i "hello" filename.txt

**Search recursively in all .txt files in a directory:**

grep -r "hello" *.txt

**Show line numbers along with matching lines:**

grep -n "hello" filename.txt

**Count the number of lines that match a pattern:**

grep -c "hello" filename.txt



**Search for lines that do not contain the pattern:**

grep -v "hello" filename.txt

**Show only filenames of files that contain the pattern:**

grep -l "hello" *.txt

**Search for whole words only:**

grep -w "hello" filename.txt

**Search for a pattern in multiple files:**

grep "hello" file1.txt file2.txt

**Show only the matched parts of a line:**

grep -o "hello" filename.txt

You can combine options as needed to refine your search. For example, to search recursively and ignore case, you could use:

grep -ri "hello" *

## Find in linux

The **find** command in Linux has numerous options that allow you to search for files and directories efficiently. Below is a comprehensive list of its options categorized based on their functionality.

### 1. Basic Options

| Option | Description | Example |
|---|---|---|
| . | Search in the current directory | find . -name "file.txt" |
| / | Search in the entire system | find / -name "file.txt" |
| -name | Search for files with a specific name (case-sensitive) | find /home -name "report.pdf" |
| -iname | Search for files with a specific name (case-insensitive) | find /home -iname "report.pdf" |
| -type f | Search for files only | find /home -type f |
| -type d | Search for directories only | find /home -type d |

### 2. File Size-Based Search

| Option | Description | Example |
|---|---|---|
| -size +100M | Find files larger than 100MB | find / -size +100M |
| -size -500K | Find files smaller than 500KB | find / -size -500K |
| -size 1G | Find files of exactly 1GB | find / -size 1G |

### 3. Date & Time-Based Search

| Option | Description | Example |
|---|---|---|
| -mtime -7 | Find files modified in the last 7 days | find /var/log -mtime -7 |
| -mtime +30 | Find files not modified in the last 30 days | find /var/log -mtime +30 |
| -atime -3 | Find files accessed in the last 3 days | find /home -atime -3 |
| -cmin -60 | Find files changed in the last 60 minutes | find /tmp -cmin -60 |

### 4. Permission-Based Search

| Option | Description | Example |
|---|---|---|
| -perm 777 | Find files with **777** (full) permissions | find /home -perm 777 |

| Option | Description | Example |
|---|---|---|
| -perm 644 | Find files with **644** permissions | find /home -perm 644 |
| -perm -4000 | Find files with **SUID** permission set | find /usr/bin -perm -4000 |

### 5. Ownership-Based Search

| Option | Description | Example |
|---|---|---|
| -user amogh | Find files owned by user **amogh** | find /home -user amogh |
| -group developers | Find files owned by group **developers** | find /home -group developers |

### 6. Depth Control

| Option | Description | Example |
|---|---|---|
| -maxdepth 2 | Limit search to 2 directory levels | find /home -maxdepth 2 -name "*.txt" |
| -mindepth 3 | Ignore first 2 levels and search deeper | find /home -mindepth 3 -name "*.txt" |



### 7. Empty Files and Directories

| Option | Description | Example |
|---|---|---|
| -empty | Find empty files and directories | find /tmp -empty |
| -type f -empty | Find empty files only | find /home -type f -empty |

### 8. Executing Commands on Found Files

| Option | Description | Example |
|---|---|---|
| -exec | Execute a command on found files | find /home -name "*.log" -exec rm {} \; |
| -ok | Ask before executing a command | find /home -name "*.log" -ok rm {} \; |

- **Example: Find all .txt files and list details**

  find /home -name "*.txt" -exec ls -l {} \;

- **Example: Find and move all .jpg files to /backup**

  find /home -name "*.jpg" -exec mv {} /backup/ \;

- **Example: Find and change file permissions to 644**

```
find /home -name "*.sh" -exec chmod 644 {} \;
```

**9. Deleting Files**

| Option | Description | Example |
|---|---|---|
| -delete | Delete found files | find /tmp -name "*.tmp" -delete |
| -type f -delete | Delete only files, not directories | find /tmp -type f -name "*.log" -delete |

⚠ **Warning**: Use -delete with caution as it permanently removes files.

**10. Logical Operators in find**

| Operator | Description | Example |
|---|---|---|
| -and | Combine two conditions (default) | find /home -name "*.txt" -and -size +1M |
| -or | Match either condition | find /home -name "*.txt" -or -name "*.pdf" |
| -not | Negate a condition | find /home -not -name "*.log" |



**11. Finding Symbolic Links**

| Option | Description | Example |
|---|---|---|
| -type l | Find symbolic links | find /home -type l |

**12. Finding Files by Inode Number**

| Option | Description | Example |
|---|---|---|
| -inum 12345 | Find file by inode number | find / -inum 12345 |

Find the inode number of a file:

ls -i filename.txt

**Search by File Type (-type)**

f: regular file

d: directory

l: symbolic link

p: named pipe (FIFO)

s: socket

b: block special file

c: character special file

**Summary**

The find command is an extremely powerful tool in Linux.

You can search based on:

- ✅ File name
- ✅ Size
- ✅ Date modified
- ✅ File type (directory, file, symbolic link)
- ✅ User ownership
- ✅ Permissions

1. ifconfig



Purpose: Displays or configures network interfaces.

Usage:

ifconfig – Show all network interfaces.

ifconfig eth0 – Show details for the interface eth0.

ifconfig eth0 up – Activate the interface.

ifconfig eth0 down – Deactivate the interface.

2. ip

Purpose: A more modern and flexible tool for managing networking.

Usage:

ip addr – Show IP addresses assigned to all network interfaces.

ip link – Show network interface details.

ip link set eth0 up – Bring up the interface eth0.

ip link set eth0 down – Bring down the interface eth0.

ip route – Show the current routing table.

ip addr add 192.168.1.100/24 dev eth0 – Assign an IP address to eth0.

3. ping

Purpose: Check the connectivity to a remote host.

Usage:

ping 8.8.8.8 – Ping the Google DNS server.

ping www.example.com – Ping a domain name.

ping -c 4 192.168.1.1 – Ping the IP address 192.168.1.1, 4 times.

4. traceroute

Purpose: Trace the route packets take to reach a network host.

Usage:

traceroute www.example.com – Trace the route to www.example.com.



5. netstat

Purpose: Displays network connections, routing tables, interface statistics, etc.

Usage:

netstat -tuln – Show all listening ports and their status.

netstat -an – Show all connections and listening ports.

netstat -r – Show the routing table.

netstat -i – Display network interfaces statistics.

6. ss

Purpose: Show socket statistics, a more modern alternative to netstat.

Usage:

ss -tuln – Show listening ports.

ss -s – Show summary statistics.

7. dig

Purpose: Query DNS servers to resolve domain names.

Usage:

dig www.example.com – Query the DNS for www.example.com.

dig @8.8.8.8 www.example.com – Query Google's DNS for www.example.com.

8. nslookup

Purpose: Query DNS servers (older tool compared to dig).

Usage:

nslookup www.example.com – Query the DNS for www.example.com.

9. route



Purpose: View or manipulate the IP routing table.

Usage:

route -n – Show the routing table in numeric format.

route add default gw 192.168.1.1 – Add a default gateway.

10. iwconfig

Purpose: Configure wireless network interfaces.

Usage:

iwconfig – Display wireless interface information.

iwconfig wlan0 essid "MyWiFi" – Connect wlan0 to a specific Wi-Fi network.

11. hostname

Purpose: Show or set the system's hostname.

Usage:

hostname – Show the current hostname.

hostname newhostname – Change the hostname to newhostname.

12. curl

Purpose: Transfer data from or to a server (supports HTTP, FTP, and more).

Usage:

curl http://example.com – Fetch a webpage.

curl -I http://example.com – Fetch the HTTP headers of a webpage.

13. wget

Purpose: Non-interactive download tool.

Usage:

wget http://example.com/file.tar.gz – Download a file from a URL.

wget -r http://example.com/ – Download a whole website recursively.



14. nmap

Purpose: Network exploration and security auditing tool.

Usage:

nmap 192.168.1.1 – Scan a single host.

nmap -p 1-65535 192.168.1.1 – Scan a range of ports on a host.

15. tcpdump

Purpose: Capture network traffic.

Usage:

tcpdump – Start capturing packets on the default interface.

tcpdump -i eth0 – Capture traffic on eth0.

tcpdump -w capture.pcap – Save captured traffic to a file.

16. ip link show

Purpose: Displays the state of network interfaces.

Usage:

ip link show – Display details for all interfaces.

17. ethtool

Purpose: Query or control network device driver and hardware settings.

Usage:

ethtool eth0 – Display information about the eth0 interface.

ethtool -s eth0 speed 100 duplex full – Set the speed and duplex mode of the interface.

18. systemctl restart network

Purpose: Restart the network service (for most Linux distros).



Usage:

systemctl restart network – Restart the networking service.

19. ip maddr

Purpose: Display multicast group memberships.

Usage:

ip maddr – Show multicast addresses.