

Linux

For DevOps Engineers and Cloud Engineers



# **Table of Contents:**

# **Fundamentals of Linux**

## **Operating System (OS):**

An **Operating System (OS)** is system software that acts as an intermediary between computer hardware and the user. It manages hardware resources and provides a set of services for application software.

🧠 **What Does an Operating System Do?**

At a high level, an OS performs five key functions:

1. **Process Management** 🧩
   * Manages processes in the system: creating, scheduling, and terminating.
   * Ensures that multiple programs can run concurrently without interfering.
2. **Memory Management** 🧠
   * Keeps track of each byte in a computer’s memory and manages allocation and deallocation.
   * Handles **virtual memory**, where programs can use more memory than physically available.
3. **File System Management** 📁
   * Organizes and manages data on storage devices like hard drives or SSDs.
   * Provides an interface for reading/writing files and managing directories.
4. **Device Management** 🔌
   * Controls hardware devices via device drivers.
   * Manages communication between software and peripherals like printers, keyboards, etc.
5. **User Interface** 🖥️
   * Offers either a Command-Line Interface (CLI) or a Graphical User Interface (GUI) to interact with the system.

**💻 Examples of Operating Systems:**

* **Windows** (by Microsoft)
* **macOS** (by Apple)
* **Linux** (open source, many distributions like Ubuntu, Fedora)
* **Android** (based on Linux, for mobile devices)
* **iOS** (by Apple, for iPhones/iPads)

| **Period** | **Focus** | **Key OSs** |
| --- | --- | --- |
| 1940s–50s | Manual operation | – |
| 1950s–60s | Batch processing | GM-NAA I/O, early IBM OS |
| 1960s–70s | Time-sharing, multiprogramming | Multics, CTSS |
| 1970s | Portability, UNIX | UNIX, VMS, BSD |
| 1980s | Personal computing, GUIs | MS-DOS, Mac OS, Windows |
| 1990s | Networking, open source | Windows 95, Linux |
| 2000s–Today | Mobile, cloud, virtualization | Android, iOS, Linux |

🕰️ **History of Operating Systems:**

🐧 **Why Linux Over Windows?**

**1. Cost-Effectiveness**

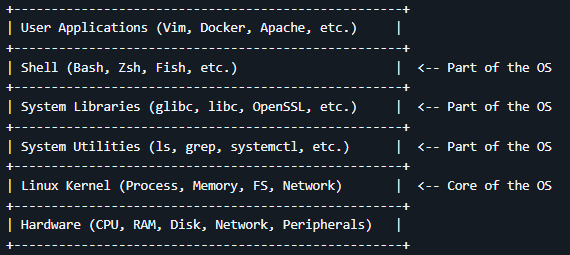
* **Free and Open Source**: Linux does not require expensive licensing fees, making it a cost-effective choice for companies.
* **Lower Maintenance Costs**: Linux is stable and requires minimal maintenance, reducing operational expenses.

**2. Performance and Efficiency**

* **Better Resource Utilization**: Linux is lightweight and consumes fewer system resources compared to Windows.
* **High Scalability**: Linux efficiently scales from small embedded systems to enterprise data centers without performance degradation.

**3. Security and Reliability**

* **Less Vulnerable to Malware**: Linux has strong user privilege separation, making it more secure against viruses and malware.
* **Frequent and Transparent Updates**: Regular security patches ensure system stability without requiring frequent reboots.
* **High Stability**: Linux systems can run for years without crashes, ensuring better uptime and reliability.

**🐧 Core Components of a Linux Machine: **

**(a) Hardware Layer**

**🔹** The physical components of the computer (CPU, RAM, disk, network interfaces, etc.).

🔹 The OS interacts with hardware using device drivers.

**(b) Kernel (Core of Linux OS)**

**🔹** The Linux Kernel is responsible for directly managing system resources, including:

**Process Management** – Schedules processes and handles multitasking.

**Memory Management** – Allocates and deallocates RAM efficiently.

**Device Drivers** – Acts as an interface between software and hardware.

**File System Management** – Manages how data is stored and retrieved.

**Network Management** – Handles communication between systems.

**(c) System Call Interface**

**🔹** A bridge between user space and kernel space.

**🔹** Allow user programs to request services from the kernel (like reading a file or starting a process).

🧩 Examples: open (), read (), fork (), exec ()

**(d) Shell (Command Line Interface - CLI)**

**🔹** A command interpreter that allows users to interact with the kernel.

🔹 Examples: Bash, Zsh, Fish, Dash, Ksh.

🔹 Converts user commands into system calls for the kernel.

**(e) User Applications**

**🔹** End-user programs like web browsers, text editors, DevOps tools, etc.

🔹 Applications interact with the OS using system calls via the shell or GUI.

**🧠 Fun Fact:**

Everything in Linux is treated as a **file** — including hardware devices!

## **🐧 Linux Distributions**

A Linux distribution (often called a distro) is a complete operating system built using the Linux kernel along with:

* System software
* User interface (GUI or CLI)
* Package Manager
* Pre-installed tools and applications

Each distro is tailored for different users — beginners, developers, security experts, or servers.

Here are some popular Linux distributions:

**Ubuntu** – One of the most beginner-friendly distros, widely used for personal and server use. It has great community support.

**CentOS (discontinued, replaced by AlmaLinux/Rocky Linux)** – Previously a popular choice for servers, based on Red Hat Enterprise Linux (RHEL).

**Debian** – A very stable and reliable distro, often used as a base for other distros like Ubuntu.

**Fedora** – A cutting-edge distro that introduces new features before they reach RHEL.

**Arch Linux** – A lightweight, rolling-release distro for advanced users who like customization.

**Kali Linux** – Designed for cybersecurity and penetration testing.

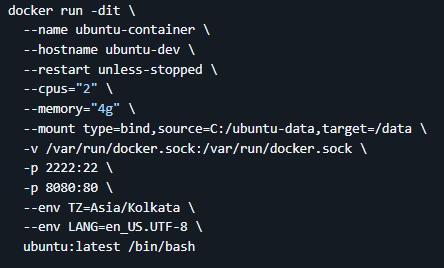
**Alpine Linux** – A lightweight, security-focused distro often used in containers.

**🖥️ Setting Up a Linux Environment on Windows and macOS:**

There are multiple ways to setup a Linux environment on a Windows or Mac machines such as cloud vm, wsl2, virtualbox, Hyperkit e.t.c., However, what I would recommend is using a container as a Linux environment.

Just install Docker desktop, run the below command and create Linux container of any distribution without worrying about the cost and connectivity issues.

**Docker Command to Run Ubuntu Linux Container (Persistent & Long-Term)**

****

**Explanation of Each Parameter: A screenshot of a computer program

AI-generated content may be incorrect.**

# **Package Managers in Linux**

**What is a Package Manager?**

A package manager is a tool that automates the process of installing, updating, configuring, and removing software in a Linux system. It ensures that software and its dependencies are managed efficiently.

🔍 **How Does a Package Manager Work?**

1. **Repositories (Repos):**
   * A package manager fetches software from official repositories (online storage of packages).
   * Example: Ubuntu gets packages from archive.ubuntu.com.
2. **Installing Software:**
   * When you install the software, the package manager:

✅ Downloads the package from the repository.

✅ Resolves dependencies (installs additional required software).

✅ Installs and configures the software automatically.

1. **Updating Software:**
   * A single command updates all installed packages to the latest version.
2. **Removing Software:**
   * The package manager also removes software cleanly without leaving unnecessary files.

📦 **Popular Package Managers in Linux**

A screenshot of a computer

AI-generated content may be incorrect.

🌍 **How Package Managers Fetch Software from Repositories**

A repository is a server that stores software packages. When a package manager installs software:

1. It checks the repository list (e.g., /etc/apt/sources.list in Ubuntu).
2. It downloads the package and its dependences.
3. It installs and configures the software automatically.

📁 **Example of an Ubuntu Repository Entry**

Types: deb

URIs: http://ports.ubuntu.com/ubuntu-ports/

Suites: noble noble-updates noble-backports noble-security

Components: main universe restricted multiverse

Signed-By: /usr/share/keyrings/ubuntu-archive-keyring.gpg

🔄 **Why Should You Run apt update After Installing Ubuntu?**

When you install Ubuntu, the packages included in the ISO image might be outdated. Running:



✅ Updates the package list from repositories.

Then, to install the latest versions of packages, run:



🛠 **Essential Package Manager Commands**

**APT (Debian, Ubuntu)**

A screenshot of a computer

AI-generated content may be incorrect.

**DNF (Fedora, RHEL, CentOS)**

A screenshot of a computer screen

AI-generated content may be incorrect.

**Pacman (Arch Linux)**

A screenshot of a computer screen

AI-generated content may be incorrect.

**Zypper (OpenSUSE)**

A black background with white text

AI-generated content may be incorrect.

**🚀 Best Practices for Using Package Managers:**

✅ Always update your package list before installing software:



✅ Use **autoremove** to clean up unused dependencies:



✅ Enable automatic security updates (Ubuntu):



# **📁 Linux Directory Structure**

In Linux, everything is organized in a tree-like hierarchy, starting from the root directory **/**.

Each folder has a specific purpose, and understanding this structure helps you navigate, troubleshoot, and manage the system effectively.

🔗 **Symbolic Links (Less Significant):**

| **Directory** | **Linked** **To** | **Description** |
| --- | --- | --- |
| /sbin | /usr/sbin | System binaries for administrative commands (e.g., reboot, shutdown, ifconfig) |
| /bin | /usr/bin | Essential user binaries (e.g., ls, cp, mv, bash) |
| /lib | /usr/lib | Shared libraries and kernel modules |

🛠️ **Important System Directories:**

| **Directory** | **Description** |
| --- | --- |
| /boot | Stores files needed to boot the system (e.g., kernel, GRUB configs) |
| /usr | Contains most user-installed applications and libraries |
| /var | Stores logs, caches, and frequently changing data |
| /etc | Contains system configuration files |

**👤 User & Application-Specific Directories:**

| **Directory** | **Description** |
| --- | --- |
| /home | Default location for user home directories (e.g., /home/chandu) |
| /opt | Used for installing optional or third-party software |
| /srv | Holds data for system services like web servers (rarely used in containers) |
| /root | Home directory for the root (superuser) |

🔄 **Temporary & Volatile Directories:**

| **Directory** | **Description** |
| --- | --- |
| /tmp | Temporary files; (cleared on reboot) |
| /run | Stores runtime process information (e.g., PID files, sockets) |
| /proc | Virtual filesystem for process/system info (e.g., /proc/cpuinfo) |
| /sys | Virtual filesystem with hardware/kernel info (e.g., /sys/class/) |
| /dev | Contains device files (e.g., /dev/null, /dev/sda). |

| **Directory** | **Description** |
| --- | --- |
| /mnt | Temporary mount point for external filesystems |
| /media | Mount point for removable devices like USBs and CDs |
| /data | Custom mount point (commonly used in WSL for Windows volumes like C:) |

💾 **Mount Points:**

**👥 User Management in Linux**

Linux is a multi-user operating system, meaning multiple users can use the system at the same time or at different times. User management ensures each user has the right permissions, resources, and isolation.

**👤 Types of Users in Linux:**

| **User Type** | **Description** |
| --- | --- |
| Root User | The superuser with full system access. Username: root. Can execute all commands. |
| Normal Users | Created by the administrator or during OS installation. Limited permissions. |
| System Users | Created by the system for services (e.g., www-data, mysql). Typically, don’t log in. |

| **File** | **Purpose** |
| --- | --- |
| /etc/passwd | Stores user account info (username, UID, home directory, shell) |
| /etc/shadow | Stores encrypted passwords and password aging info |
| /etc/group | Contains group information |
| /etc/sudoers | Configures who can use sudo and how |

**📁 Important Files for User Management:**

**Note:** Use visudo to edit /etc/sudoers safely.

**➕ Creating Users in Linux:**

**To create a new user in Linux, use:**

* useradd Command (For most Linux distributions)

****

**Note:** This creates a user without a home directory.

* To create a user with a home directory:



* To specify a shell**:**

****

* **adduser** Command (For Debian-based systems):

****

This is an interactive command that asks for a password and additional details.

**🔐 Set or Change User Password:**

To set or change a user’s password**:**

****

**Enforcing Password Policies:**

* Password expiration**:** Set password expiry days

****

* Lock a user account

****

* Unlocking a user account

****

**🔄 Modify a User:**

**Modify an existing user with usermod:**

* Change the username**:**

****

* Change the home directory**:**

****

* Change the default shell**:**

****

**➖Delete a User:**

* To remove a user but keep their home directory**:**

****

* To remove a user and their home directory**:**

****

**👥 Group Management:**

Groups help assign shared permissions**.**

* Create a Group**:**

****

* Add User to Group**:**

****

* View Groups of a User**:**

****

* Changing Primary Group**:**

****

**Sudo Access and Privilege Escalation:**

**Adding a User to Sudo Group:**

* On Debian-based systems**:**

****

* On RHEL-based systems**:**

****

**🔐Granting Specific Commands with Sudo:**

Edit the sudoers file**:**

****

Then add**:**

****

# **📁 File Management in Linux**

Linux treats everything as a file, including text files, directories, devices, and even hardware!

Learning file management is essential for navigating, creating, editing, and organizing files and directories in a Linux environment.

**📌 Basic File Terminology:**

| **Term** | **Description** |
| --- | --- |
| File | A collection of data (e.g., text, images, scripts). |
| Directory | A folder that contains files or other directories. |
| Path | Location of a file or directory. Can be absolute or relative. |

**📂 File & Directory Navigation:**

| **Command** | **Description** |
| --- | --- |
| pwd | Show current working directory |
| ls | List of all files and directories |
| cd | Change directory |
| cd .. | Go one level up |
| cd / | Go to the root directory |
| cd ~ | Go to home directory |

**🗃️ File and Directory Operations:**

**📁 Create:**

**A black and white screen with white text

AI-generated content may be incorrect.**

**✏️ Rename / Move:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**❌ Delete:**

**A screen shot of a computer

AI-generated content may be incorrect.**

**📄 Copy:**

**A black and white screen with white text

AI-generated content may be incorrect.**

**🔍 View File Contents and Editing:**

| **Command** | **Description** |
| --- | --- |
| cat file.txt | Displays file content. |
| tac file.txt | Displays file content in reverse order. |
| head -n 10 file.txt | Displays the first 10 lines of a file. |
| tail -n 10 file.txt | Displays the last 10 lines of a file. |
| less file.txt | Opens a file for viewing with scrolling support. |
| more file.txt | Like less but only moves forward. |
| nano file.txt | Opens a simple text editor. |
| vi file.txt | Opens a powerful text editor. |
| echo 'Hello' > file.txt | Writes text to a file, overwriting existing content. |
| echo 'Hello' >> file.txt | Appends text to a file without overwriting. |

# **✍️ VI Editor Shortcuts in Linux**

The vi editor (or its improved version vim) is a powerful text editor in Linux, commonly used to edit configuration files and scripts directly from the terminal.

It works in three modes:

| **Mode** | **Description** |
| --- | --- |
| **Normal mode** | Default mode when vi starts — used for navigation and commands |
| **Insert mode** | Used for typing/editing text |
| **Command mode** | Used for saving, quitting, searching, etc. |

**🚀 Launching VI:**

**A black and white background with white text

AI-generated content may be incorrect.**

* Open the file in normal mode**.**
* If the file doesn’t exist, it will be created upon saving.

| **Key** | **Action** |
| --- | --- |
| i | Enter insert mode before the cursor |
| I | Enter insert mode at the beginning of the line |
| a | Enter insert mode after the cursor |
| A | Enter insert mode at the end of the line |
| o | Open a new line below |
| O | Open a new line above |
| Esc | Exit insert mode and return to normal mode |

**🧭 Mode Switching:**

| **Command** | **Action** |
| --- | --- |
| : w | Save (write) file |
| : q | Quit |
| : wq or ZZ | Save and quit |
| : q! | Quit without saving |
| :x | Save and exit (same as: wq) |
| : e filename | Open another file in vi |
| : set nu | Show line numbers |
| : set nonu | Hide line numbers |

**💾 Basic Command Mode Shortcuts (: commands)**

| **Command** | **Action** |
| --- | --- |
| yy or Y | Copy (yank) current line |
| p | Paste after the cursor |
| P | Paste before the cursor |
| dd | Delete (cut) current line |
| x | Delete character under cursor |
| u | Undo last change |
| Ctrl + r | Redo |
| r | Replace a single character |
| R | Overwrite mode (until Esc) |

**📝 Text Editing in Normal Mode:**

| **Key** | **Action** |
| --- | --- |
| h | Move left |
| l | Move right |
| j | Move down |
| k | Move up |
| 0 | Move to the beginning of line |
| ^ | First non-blank of line |
| $ | End of line |
| w | Jump forward one word |
| b | Jump back one word |
| G | Go to the end of file |
| gg | Go to the beginning of file |
| : n | Go to line number n |

**🧭 Navigation Shortcuts:**

| **Command** | **Action** |
| --- | --- |
| /text | Search forward for "text" |
| ?text | Search backward for "text" |
| n | Repeat last search |
| :%s/old/new/g | Replace all "old" with "new" in entire file |
| :s/old/new/g | Replace in current line |

**🔍 Search and Replace:**

# **🔐 SSH (Secure Shell)**

**✅ What is SSH?**

SSH stands for Secure Shell. It's a protocol used to securely connect to a remote computer over a network — typically used to access servers, Linux systems, and cloud instances.

Think of SSH as a secure way to control another computer using the terminal just like you're sitting in front of it!

**🛡️ Why is SSH Important?**

* **Encrypted Communication** – All data sent between you and the server is encrypted (safe from eavesdropping).
* **Remote Access –** Control remote systems from anywhere**.**
* **File Transfer** – Send and receive files securely via SCP/SFTP.
* **Automation** – Useful for running scripts, updates, and deployment tasks on servers.

**🔐 How SSH Works**

SSH uses **key-based** or **password-based** authentication.

1. You run an SSH command.
2. SSH connects to the remote machine's SSH server (usually on **port 22**).
3. You log in using a **username + password** or **SSH key**.
4. Once connected, you can run commands, manage files, or even tunnel other connections securely.

**🧑‍💻 Basic SSH Command:**

**A close up of a black and white screen

AI-generated content may be incorrect.**

**🔹 Example:**

**A black and white screen with white text

AI-generated content may be incorrect.**

* **ubuntu:** username on the remote machine.
* **192.168.1.10:** IP address or domain name of the remote machine.

**📁 Common SSH Use Cases:**

**A screenshot of a computer

AI-generated content may be incorrect.**

# **📦 SCP (Secure Copy Protocol) in Linux**

**✅ What is SCP?**

**SCP** stands for **Secure Copy Protocol**. It is a command-line tool used to copy files or directories between two systems over SSH.

* It encrypts files during transfer (unlike cp or ftp).
* Works **locally → remote**, **remote → local**, or even **remote → remote.**

📌 **Basic Syntax:**



* source: file or folder you want to copy.
* destination: where you want to copy it (can be local or remote).
* Both can be in the format: user@host:/path.

**📂 Common Use Case:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**⚙️ Useful Options:**

| **Option** | **Description** |
| --- | --- |
| -r | Recursively copy entire directories |
| -P | Specify port number if not using default (22) |
| -i | Use a specific identity file (SSH key) |
| -C | Enable compression during transfer |
| -v | Verbose output (helpful for debugging) |

# **🔐 File Permissions Management in Linux**

**Introduction to File Permissions:**

Linux file permissions determine who can read, write, or execute files and directories. Each file and directory have three levels of permission:

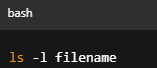
**👥 Types of Users:**

* **Owner (User):** The creator of the file.
* **Group**: Users belonging to the assigned group.
* **Others**: All other users on the system.

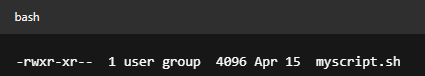
Permissions are represented as:

* **Read (r or 4)** – View file contents.
* **Write (w or 2)** – Modify file contents.
* **Execute (x or 1)** – Run scripts or programs.

**📄 Viewing File Permissions:**

****

**Example Output:**

****

**Breakdown:**

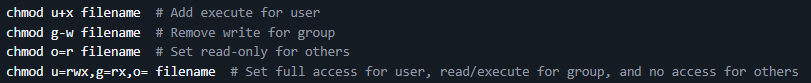
* - = regular file (d = directory)
* rwx = Owner can read, write, execute
* r-x = Group can read, execute
* r-- = Others can only read

**✏️ Changing Permissions with chmod:**

**Using Symbolic Mode**

Modify permissions using symbols:

* Add (+), remove (-), or set (=) permissions.

Examples:

**Using Numeric (Octal) Mode**

Each permission has a value:

* Read (4), Write (2), Execute (1).

Examples:



**👑 Changing Ownership with chown:**

Modify file owner and group**:**

****

Recursively change ownership**:**

****

**🔁 Changing Group Ownership with chgrp:**

****

**📁 Special Permission Bits (Advanced):**

**SetUID (s on user execute bit):**

Allows users to run a file with the file owner's permission.

****

**Example**: /usr/bin/passwd allows users to change their passwords**.**

**SetGID (s on group execute bit):**

**Files:** Users run the file with the group's permissions**.**

**Directories:** Files created inside inherit the group**.**

****

**Sticky Bit (t on others execute bit):** Used directories to allow only the owner to delete their files**.**

****

**Example:**/tmp directory**.**

**Default Permissions:**umask

**umask** defines default permissions for new files and directories. Check current umask:

****

**Set a new umask:**

****

# **⚙️ Process Management in Linux**

**What is a Process?**

A process is a running instance of a program. Every time you execute a command; Linux creates a process to handle it.

Each process has:

* PID (Process ID) – Unique identifier
* PPID (Parent Process ID) – ID of the parent process
* UID (User ID) – Who owns the process
* Resource info – CPU, memory usage, etc.

**Index of Commands Covered**:

**Viewing Processes:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Managing Processes:**

**A screenshot of a black background with white text

AI-generated content may be incorrect.**

**Background & Foreground Processes:**

**A black background with white text

AI-generated content may be incorrect.**

**Monitoring System Processes:**

**A black background with white text

AI-generated content may be incorrect.**

| Priority Value | Meaning |
| --- | --- |
| -20 | Highest priority |
| 0 | Default priority |
| +19 | Lowest priority |

**Note:** Only the root user can set negative nice values.

**Daemon Process Management:**

**A screenshot of a computer

AI-generated content may be incorrect.**

# **📊 Linux System Monitoring**

System Monitoring in Linux is about keeping an eye on the system's health — such as CPU usage, memory, disk, network, and running processes — to troubleshoot issues or optimize performance.

**🔍 Why Monitor a Linux System?**

* Detect high CPU or memory usage
* Identify misbehaving processes
* Monitor system uptime and load
* Ensure system is stable, secure, and performant
* Perform resource planning for scaling

**Index of Commands Covered:**

**🧠 CPU and Memory Monitoring:**

**🔹 top – Real-Time System Monitoring**

**A black and white screen with white text

AI-generated content may be incorrect.**

* Shows a live view of processes and system resource usage (CPU, memory, etc.)
* Press q to quit, k to kill a process, P to sort by CPU usage.

**🔹 htop – Interactive Process Viewer (Requires Installation)**

**A black and white screen with white text

AI-generated content may be incorrect.**

* A colorful, user-friendly alternative to top.
* Allows scrolling, sorting, and killing processes with keyboard shortcuts.
* To install:

A black and white screen

AI-generated content may be incorrect.

🔹 **vmstat – System Performance Statistics:**

**A black and white screen

AI-generated content may be incorrect.**

* Reports on memory, processes, CPU, I/O, and more.
* 1 5 = run every 1 second, 5 times.

**🔹 free -m – Memory Usage in MB**:

**A black and white screen with white text

AI-generated content may be incorrect.**

* Shows total, used, and free memory (RAM and swap) in megabytes.

🔹 **free -h – Human-Readable Memory Usage:**

**A black and white screen with white text

AI-generated content may be incorrect.**

* Same as free -m but output is easier to read (e.g., 2.3G, 512M).

💾 **Disk Monitoring:**

**🔹 df -h – Disk Space Usage:**

**A black and white text

AI-generated content may be incorrect.**

* Displays total and available disk space for mounted file systems.
* -h = human-readable sizes (e.g., GB/MB).

**🔹 du -sh \* – Directory Size Usage:**

**A black and white text

AI-generated content may be incorrect.**

* Shows size of each item (file/folder) in the current directory.
* -s: summary; -h: human-readable

**🔹 iostat – CPU and Disk I/O Statistics:**

**A black and white text

AI-generated content may be incorrect.**

* Displays CPU load and disk input/output stats.
* May require:

A black and white screen

AI-generated content may be incorrect.

🌐 **Network Monitoring:**

**🔹 ifconfig – Show Network Interfaces (Deprecated):**

**A black and white screen with white text

AI-generated content may be incorrect.**

* Shows IP addresses and interface status.
* Now deprecated in favor of ip a.

**🔹 ip a – Show Network Interface Details:**

****

* Modern replacement for ifconfig.
* Shows interface names, IP addresses, MAC addresses, and more.

**🔹 netstat -tulnp – Active Connections & Ports:**

**A screenshot of a computer

AI-generated content may be incorrect.**

* -t: TCP
* -u: UDP
* -l: Listening ports
* -n: Show numeric addresses
* -p: Show process IDs and names

**Note**: netstat may need to be installed with net-tools.

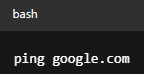
**🔹 ss -tulnp – Faster Alternative to netstat:**

**A black and white screen with white text

AI-generated content may be incorrect.**

* Performs the same job as Netstat but faster and with more options.
* Pre-installed on most modern distros.

**🔹 ping hostname – Test Network Connectivity**

****

* Send ICMP packets to test if the host is reachable.
* Use Ctrl+C to stop.

**🔹 traceroute hostname – Trace Network Path**

**A black and white screen with white text

AI-generated content may be incorrect.**

* Shows each hop between your system and the target server.
* It is useful to diagnose where network delays occur.

🔹 If not installed:



**🔹 nslookup domain – DNS Lookup Info:**

**A black and white screen with white text

AI-generated content may be incorrect.**

* Shows IP address of a domain and DNS server response.
* Good for testing DNS resolution issues.

📝 **Log Monitoring:**

**🔹 tail -f /var/log/syslog – Live Log Monitoring:**

**A black and white text

AI-generated content may be incorrect.**

* Continuously shows new lines added to the system log.
* Use CTRL + C to stop.

**🔹 journalctl -f – Live Logs for Systemd Systems:**

**A black and white screen with white text

AI-generated content may be incorrect.**

* Monitors system logs in real-time.
* Works only on systemd-based systems like Ubuntu, Fedora, etc.

**🔹 dmesg | tail – View Kernel Logs:**

**A black and white text

AI-generated content may be incorrect.**

* Displays the latest kernel messages (hardware, boot info, driver messages).
* Useful for checking hardware-related errors.

# **🌐 Networking in Linux**

Linux provides a powerful set of tools and commands to manage, monitor, and troubleshoot network configurations and connectivity.

**Key Concepts:**

| Concept | Description |
| --- | --- |
| IP Address | Unique identifier for a device on a network |
| MAC Address | Hardware address of the network interface |
| Interface | Network device (e.g., eth0, wlan0, lo) |
| Hostname | Name of the Linux system on the network |
| DNS | Resolves domain names to IP addresses |

**Networking Commands**:

1. ping google.com – Checks connectivity to a remote server.
2. ifconfig – Displays network interfaces (deprecated, use ip).
3. ip a – Shows IP addresses of network interfaces.
4. netstat -tulnp – Displays open network connections.
5. curl https://example.com – Fetches a webpage's content.
6. wget https://example.com/file.zip – Downloads a file from the internet.

# **💽 Disk and Storage Management in Linux**

**Introduction to Disk and Storage Management:**

Disk management in Linux involves monitoring, formatting, partitioning, and mounting storage devices. It helps you understand how disk space is allocated and used, and how to manage additional drives or volumes.

| **Task** | **Description** |
| --- | --- |
| View disk usage | Check how much space is used and available |
| Partition disks | Divide a disk into logical sections |
| Format disks | Prepare partitions with a file system |
| Mount/unmount | Attach/detach disks to the system |
| Monitor I/O | Analyze disk read/write activity |

**📦 Common Disk Management Tasks:**

**📊 Disk Usage Monitoring Commands:**

**🔹 df – Disk Free Space:**

****

* Shows how much disk space is used and available on mounted filesystems.
* -h = human-readable (e.g., GB/MB)

**🔹** **du – Disk Usage per Directory:**

****

* Shows the size of a directory and its contents.
* -s: summary, -h: human-readable

**🔹 lsblk – List Block Devices:**

****

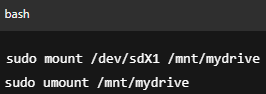
* Lists all attached block devices (like disks and partitions).
* Does not show mount points by default unless -f is used.

**🔹 blkid – Show UUIDs and File System Types:**

****

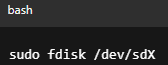
* Displays information about devices, UUIDs, and file system types.

**🔹 mount / umount – Mount or Unmount File Systems:**

****

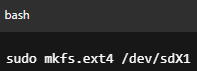
* mount: attaches a device to the system
* umount: detaches the device safely

**🔹 fdisk – Partition a Disk (Interactive):**

****

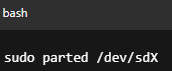
* Used to create, delete, or modify partitions on MBR disks.

**🔹 mkfs – Create File System:**

****

* Formats a partition with the ext4 file system (or other: xfs, ntfs, etc.)

**🔹 parted – Advanced Partitioning Tool:**

****

* Useful for managing large drives, GPT, resizing, scripting.

**🔹 df -T – Check File System Type:**

****

* Shows the type of file system used on mounted partitions (e.g., ext4, xfs, etc.)

**Logical Volume Management (LVM)**

* pvcreate /dev/sdX – Create a physical volume
* vgcreate vg\_name /dev/sdX – Create a volume group
* lvcreate -L 10G -n lv\_name vg\_name – Create a logical volume
* mkfs.ext4 /dev/vg\_name/lv\_name – Format an LVM partition
* mount /dev/vg\_name/lv\_name /mnt – Mount an LVM partition

**Swap Management**

* mkswap /dev/sdX – Create a swap partition
* swapon /dev/sdX – Enable swap space
* swapoff /dev/sdX – Disable swap space