**What is Linux?**

**Linux** is a **free and open-source operating system** based on **Unix**. It acts as an interface between hardware and software, allowing programs to run, manage resources, and communicate with hardware devices.

It is widely used in servers, embedded systems, desktops, mobile devices (like Android), and cloud infrastructure.

**🧩 Components of Linux Operating System**

Linux is made up of several **key components**, each responsible for different aspects of system functionality:

**1. Kernel (Core of Linux)**

* **Function:** Manages system resources (CPU, memory, I/O, etc.).
* Acts as a **bridge between hardware and software**.
* Types:
  + **Monolithic Kernel**: Everything runs in one space (Linux uses this).
* Manages:
  + Process scheduling
  + Memory management
  + Device drivers
  + System calls

**2. System Libraries**

* Special functions/programs used by applications to interact with the kernel.
* **glibc** is the most common system library in Linux.
* Examples:
  + libc.so: C standard library
  + libm.so: Math library

**3. System Utilities**

* Essential tools and commands used to manage the system.
* Examples:
  + ls, cp, mv, ps, top, kill, chmod, df, du, etc.
* These allow users and admins to interact with the system effectively.

**4. Shell (Command Line Interface)**

* Acts as an **interface between user and kernel**.
* Users type commands, which the shell interprets and sends to the kernel.
* Types:
  + **Bash (Bourne Again SHell)** – most common
  + **Zsh, Ksh, Tcsh, Fish**, etc.

**5. Hardware Layer**

* Physical components: CPU, RAM, HDD/SSD, network cards, etc.
* Managed entirely by the **kernel** through device drivers.

**6. User Applications**

* Programs run by users: text editors, browsers, compilers, media players, etc.
* Examples:
  + vim, firefox, gcc, gimp, docker, etc.

**Optional: Init System**

* Responsible for bootstrapping the user space and managing services/daemons.
* Examples:
  + **Systemd**, **SysVinit**, **Upstart**

**🔄 Linux Architecture Diagram (Textual)**

sql

CopyEdit

+--------------------------+

| User Applications |

+--------------------------+

| Shell / CLI / GUI |

+--------------------------+

| System Utilities |

+--------------------------+

| System Libraries |

+--------------------------+

| Kernel |

| - Process Mgmt |

| - Memory Mgmt |

| - Device Drivers |

| - Filesystem Mgmt |

| - Network Stack |

+--------------------------+

| Hardware Layer |

+--------------------------+

**Linux Boot Process (Kernel Involvement - Intermediate Level)**

The Linux boot process consists of several stages that culminate in a running system. Below are the **key stages** with a focus on the **kernel and process handling**:

**🧱 1. BIOS/UEFI Stage**

* **BIOS (Basic Input/Output System)** or **UEFI** initializes hardware.
* Loads the **bootloader** (like GRUB) from the **MBR** (Master Boot Record) or EFI partition.

**🧹 2. Bootloader Stage (e.g., GRUB)**

* GRUB locates and loads the **Linux kernel (vmlinuz)** and **initial RAM disk (initrd or initramfs)** into memory.
* Passes control to the kernel with parameters (e.g., root filesystem path).

**🧠 3. Kernel Initialization**

Once the bootloader loads the kernel, here's what happens:

**a. Decompression and Setup**

* vmlinuz is a compressed kernel image → it decompresses itself in memory.
* Kernel sets up **protected mode**, memory management, and hardware abstraction.

**b. Initramfs or Initrd Execution**

* Temporary root filesystem loaded into RAM.
* Kernel mounts initramfs and executes /init inside it (a shell script or binary).
* Responsible for:
  + Loading necessary drivers/modules.
  + Mounting the actual root filesystem (e.g., /dev/sda2).

**🚀 4. Switch to Real Root Filesystem**

* Kernel executes a switch from initramfs to the **real root filesystem**.
* Unmounts initramfs, mounts real /.

**🔁 5. Starting init Process (PID 1)**

* The kernel executes the first user-space process: init.
* This is the **parent of all user-space processes**.

Depending on the Linux distro, init could be:

* **System V init** (/sbin/init)
* **Systemd** (/lib/systemd/systemd)
* **Upstart** (older Ubuntu)

**📋 6. Init/Systemd Process Responsibilities**

* **Mounts essential filesystems** like /proc, /sys, /dev.
* **Spawns services/daemons** as defined in runlevels or systemd targets.
* Starts getty on terminals for user login.
* Starts **networking, cron, dbus**, etc.

**⚙️ 7. Kernel Process Management**

* Kernel always has several **internal processes**, usually kernel threads:
  + kthreadd: Kernel thread manager.
  + ksoftirqd: Handles soft IRQs.
  + kworker: Manages background work queues.
  + kswapd: Handles memory swapping.
  + rcu\_sched: Read-copy-update synchronization.
* You can see them in top, ps, or /proc.

## 🧩 ****Types of Linux Commands****

Linux commands can be grouped based on their **functionality**:

### 1. 📁 ****File and Directory Commands****

Used for file creation, viewing, copying, moving, deleting, and permissions.

| **Command** |  | **Description** |
| --- | --- | --- |
| ls |  | List files and directories |
| cd |  | Change directory |
| pwd |  | Show current directory path |
| touch |  | Create empty file |
| mkdir |  | Create directory |
| cp |  | Copy file or directory |
| mv |  | Move or rename |
| rm |  | Remove file or directory |
| chmod |  | Change file permissions |

### 2. 🧠 ****Process Management Commands****

Used to view, manage, or kill running processes.

| **Command** | **Description** |
| --- | --- |
| ps | View running processes |
| top | Dynamic real-time process view |
| kill | Kill a process by PID |
| killall | Kill processes by name |

### 3. 👤 ****User Management Commands****

Manage users, groups, and permissions.

| **Command** | **Description** |
| --- | --- |
| adduser | Add a new user |
| passwd | Change user password |
| whoami | Show current user |
| Su | Switch user |
| Sudo | Execute command as root |
| Id | Show user ID and group info |
| groupadd | Create a new group |

### 4. 🌐 ****Networking Commands****

Used for network configuration, monitoring, and troubleshooting.

| **Command** | **Description** |
| --- | --- |
| ifconfig / ip | Show/assign IP addresses |
| Ping | Test connectivity |
| Netstat | Show network connections |
| Ss | Detailed network info |
| Curl | Transfer data from or to a server |
| Wget | Download files from the internet |
| hostname | Display system hostname |

### 5. 🛠️ ****System Information Commands****

Display system hardware and OS details.

| **Command** | **Description** |
| --- | --- |
| uname -a | Kernel and OS info |
| df -h | Disk space usage |
| free -m | Memory usage |
| uptime | System uptime |
| who | Logged in users |

### 6. 📦 ****Package Management Commands****

Install, update, and remove software packages.

| **Package Manager** | **Commands** |
| --- | --- |
| Debian/Ubuntu (apt) | apt install, apt update, apt remove |
| Red Hat/CentOS (yum or dnf) | yum install, dnf update |
| Arch Linux (pacman) | pacman -S, pacman –R |

### 7. 🐚 ****Shell/Bash Commands****

Special built-in commands for scripting and shell features.

| **Command** | **Description** |
| --- | --- |
| echo | Print text |
| read | Read input |
| export | Set environment variable |
| history | Show command history |

### 8. 🧪 ****Text Processing Commands****

Used to process and manipulate text data.

| **Command** | **Description** |
| --- | --- |
| Cat | View file content |
| Grep | Search text |
| Awk | Pattern scanning and processing |
| Sed | Stream editor (search & replace) |
| Cut | Extract fields |
| Sort | Sort lines |

## ****1. Crontab – Automate Repetitive Tasks****

### ****What is Crontab?****

Crontab (cron table) is used to schedule jobs (commands/scripts) to run at specific times or intervals.

### ****Basic Syntax of Crontab:****

\* \* \* \* \* command-to-execute

| | | | |

| | | | └── Day of the week (0 - 7) [Sunday = 0 or 7]

| | | └──── Month (1 - 12)

| | └────── Day of month (1 - 31)

| └──────── Hour (0 - 23)

└────────── Minute (0 - 59)

### ****Step-by-Step Example:****

1. **Open Crontab Editor:**

crontab -e

1. **Schedule a Script at 2 AM Daily:**

0 2 \* \* \* /home/user/backup.sh

1. **Common Examples:**
   * Every minute:

\* \* \* \* \* echo "Hi" >> /tmp/test.txt

* + Every Sunday at 5 PM:

0 17 \* \* 0 /path/to/script.sh

1. **View Current Cron Jobs:**

crontab -l

1. **Delete All Cron Jobs:**

crontab -r

## ****2. SSH Keys and Connecting to Multiple Servers****

### ****Goal:****

SSH (Secure Shell) allows remote login. Using SSH keys avoids password prompts.

### ****Step-by-Step:****

1. **Generate SSH Key on Local Machine:**

ssh-keygen

Press Enter 3 times to accept defaults. It generates:

* + Public key: ~/.ssh/id\_rsa.pub
  + Private key: ~/.ssh/id\_rsa

1. **Copy Public Key to Remote Servers:**

ssh-copy-id user@server1\_ip

ssh-copy-id user@server2\_ip

1. **Login Without Password:**

ssh user@server1\_ip

ssh user@server2\_ip

1. **Configure ~/.ssh/config (optional):**

Host server1

HostName 192.168.1.10

User ubuntu

Host server2

HostName 192.168.1.11

User ubuntu

Now login using:

ssh server1

ssh server2

## ****3. Partition Creation and Mounting****

### ****Step-by-Step:****

1. **List Available Disks:**

lsblk

sudo fdisk -l

1. **Partition New Disk (/dev/sdb):**

sudo fdisk /dev/sdb

Inside fdisk:

* + Press n (new partition)
  + Press p (primary)
  + Accept defaults (Enter)
  + Press w (write and exit)

1. **Format Partition (e.g., /dev/sdb1):**

sudo mkfs.ext4 /dev/sdb1

1. **Create Mount Directory and Mount:**

sudo mkdir /mnt/data

sudo mount /dev/sdb1 /mnt/data

1. **Make Mount Persistent (After Reboot):**

sudo blkid /dev/sdb1

Copy the UUID and add to /etc/fstab:

UUID=xxxx-xxxx /mnt/data ext4 defaults 0 2

## ****4. Rebooting Steps in Linux****

### ****Immediate Reboot:****

sudo reboot

### ****Reboot After 5 Minutes:****

sudo shutdown -r +5

### ****Schedule Reboot at Specific Time:****

sudo shutdown -r 23:30

### ****Cancel Scheduled Reboot:****

sudo shutdown -c

### ****Check Uptime and Last Reboot:****

uptime

last reboot

## ****5. File Transfer Using SCP****

### ****Transfer Local File to Remote Server:****

scp file.txt user@remote\_ip:/home/user/

### ****Transfer Directory:****

scp -r /local/dir user@remote\_ip:/home/user/

### ****Transfer From Remote to Local:****

scp user@remote\_ip:/home/user/file.txt /local/path/

### ****Use Different SSH Port (e.g., 2222):****

scp -P 2222 file.txt user@remote\_ip:/path/

## ****6. Linux File System Structure (FHS)****

| **Path** | **Description** |
| --- | --- |
| / | Root of all directories |
| /bin | Basic user commands (ls, cp, mv) |
| /sbin | System commands (shutdown, reboot) |
| /etc | Configuration files (passwd, ssh/sshd\_config) |
| /home | Users’ home directories |
| /root | Root user’s home directory |
| /var | Logs and spool data (e.g., /var/log/) |
| /usr | User-installed programs |
| /tmp | Temporary files |
| /dev | Device files (hard disk, USB) |
| /boot | Bootloader, kernel, initrd |
| /mnt | Temporary mount point |
| /proc | Virtual filesystem (process info) |

## ****7. Rsync – Advanced File Sync Tool****

### ****Sync Local Folder to Remote Server:****

rsync -avz /local/dir/ user@remote\_ip:/remote/dir/

### ****Sync With Progress:****

rsync -avz --progress /local/dir/ user@remote\_ip:/remote/dir/

### ****Dry Run (Simulate):****

rsync -avz --dry-run /src/ /dest/

### ****Sync Remote to Local:****

rsync -avz user@remote\_ip:/remote/dir/ /local/dir/

### ****Exclude Files/Folders:****

rsync -avz --exclude 'temp/' /src/ /dest/

### ****What is**** rsync ****in Linux?****

**rsync (Remote Sync)** is a powerful command-line tool in Linux used for **synchronizing files and directories** between two locations — either on the same system or across remote systems over SSH.

It is commonly used for **backups, mirroring files, and fast incremental file transfers**.

### ****Key Features of rsync:****

* **Synchronizes only changed parts** of files (very efficient).
* **Works locally or remotely** (via SSH).
* Supports **compression** and **encryption**.
* Supports **resume**, **exclude/include rules**, and **preserves file permissions**, timestamps, etc.

### ****Basic Syntax:****

rsync [options] source destination

### ****Common Examples:****

1. **Sync a local folder to another folder:**

rsync -av /home/user/docs/ /backup/docs/

1. **Sync files to a remote server:**

rsync -avz /home/user/docs/ user@192.168.1.10:/home/user/backup/

1. **Sync from remote to local:**

rsync -avz user@192.168.1.10:/home/user/backup/ /home/user/docs/

1. **Exclude files or folders:**

rsync -av --exclude '\*.log' /src/ /dest/

1. **Dry run (simulate without actual copying):**

rsync -av --dry-run /src/ /dest/

### ****Key rsync Options:****

| **Option** | **Meaning** |
| --- | --- |
| -a | Archive mode (preserves permissions, timestamps, symbolic links) |
| -v | Verbose (show progress) |
| -z | Compress file data during transfer |
| --delete | Delete files in destination that don’t exist in source |
| --progress | Show progress of each file transfer |

## ****What is**** find ****in Linux?****

The find command is used to **search for files and directories** in a directory hierarchy based on various conditions like name, size, type, date modified, and more.

## ****Basic Syntax:****

find [path] [options] [expression]

## ****Useful Examples:****

### ****1. Find a file by name:****

find /home/user -name "file.txt"

* Searches /home/user for a file named exactly file.txt.

### ****2. Case-insensitive search:****

find /var/log -iname "syslog"

### ****3. Find all**** .log ****files:****

find / -type f -name "\*.log"

### ****4. Find directories named “config”:****

find /etc -type d -name "config"

## ****Filter by Time and Size****

### ****5. Find files modified in last 7 days:****

find /home -type f -mtime -7

### ****6. Find files modified more than 30 days ago:****

find /home -type f -mtime +30

### ****7. Find files larger than 100MB:****

find / -type f -size +100M

### ****8. Find empty files or directories:****

find /tmp -empty

## ****Run Actions on Found Files****

### ****9. Delete all**** .tmp ****files:****

find /tmp -type f -name "\*.tmp" -delete

### ****10. Execute command on each file:****

find /var/log -type f -name "\*.log" -exec rm {} \;

* {} = current file, \; = end of command

## ****Bonus: Combine with**** grep

### ****11. Find files containing a keyword:****

find /var/log -type f -exec grep -i "error" {} \;

## ****Important Flags Recap:****

| **Flag** | **Meaning** |
| --- | --- |
| -name | Match filename (case-sensitive) |
| -iname | Match filename (case-insensitive) |
| -type f | Search for files |
| -type d | Search for directories |
| -size +100M | File size greater than 100 MB |
| -mtime -7 | Modified in last 7 days |
| -delete | Delete matched files |
| -exec | Execute command on matched files |

**Linux Commands**

| **Topic** | **Description** | **Example Command(s)** |
| --- | --- | --- |
| **Crontab** | Schedule tasks to run automatically at specified times or intervals | crontab -e 0 2 \* \* \* /home/user/backup.sh crontab -l |
| **SSH & SSH Keys** | Secure remote login and passwordless access using key-based authentication | ssh-keygen ssh-copy-id user@server ssh user@server |
| **Mount & Partitioning** | Create, format, and mount new disks or partitions | fdisk /dev/sdb mkfs.ext4 /dev/sdb1 mount /dev/sdb1 /mnt/data |
| **Rebooting in Linux** | Safely restart the Linux system with various timing options | sudo reboot shutdown -r +5 shutdown -c |
| **SCP (Secure Copy)** | Transfer files securely between local and remote systems over SSH | scp file.txt user@server:/path/ scp -r dir/ user@server:/path/ |
| **Linux File System Structure** | Understand major system directories and their purposes | /bin, /etc, /home, /var, /root, /dev, /proc, /mnt, /tmp, /boot |
| **Rsync** | Efficiently sync files/folders locally or remotely with speed and control | rsync -avz /src/ user@server:/dest/ rsync -av --delete --progress /src/ /dest/ |
| **Find Command** | Search for files/folders by name, size, type, time, etc. | find / -name "\*.log" find /home -mtime -7 find /tmp -type f -delete |