Linux Notes

What is Linux?

Linux is a free, open-source, and Unix-like operating system used to run computers, servers, and even smartphones.

It is based on the Linux kernel, created by Linus Torvalds in 1991.

The name "Linux" comes from combining **Linus + Unix**.

. UNIX Overview

- Unix is a powerful, **multiuser, multitasking** operating system.
- Originally developed in **1970s at AT&T Bell Labs**.
- It is the **foundation of many modern operating systems**.

Operating System (OS)

- An **Operating System** is a software that acts as a **bridge between the user and the computer hardware**.
- It manages hardware resources (CPU, memory, storage, devices) and provides services for running applications.
- OS converts **high-level instructions** (what we type or click) into **low-level instructions** (that hardware understands).

👉 Example: When you click "Play" on a music player, the OS tells the CPU to process the audio file, manages memory, and sends the sound output to speakers.

Types of Interfaces

1. **CLI (Command Line Interface)**
    - User interacts by typing commands.
    - Example: Linux terminal (ls, cd, pwd).
    - Suitable for advanced users and automation.
2. **GUI (Graphical User Interface)**
    - User interacts using icons, windows, and menus.
    - Example: Windows Desktop, macOS Finder.

Linux Notes

- Easier for beginners.

▪ Types of OS

1. Desktop OS

- Used for **personal or office computers**.
- Examples: **Windows 10/11, Ubuntu Desktop, Kali Linux, macOS**.
- Main purpose: Running applications like browsers, MS Office, games, etc.

👉 Example: You use Windows 11 on your laptop to browse the internet, do projects, and watch movies.

---

2. Server OS

- Designed for **managing servers** (large-scale computing systems).
- More stable, supports multiple users, security features, and resource sharing.
- Examples: **CentOS, Red Hat Enterprise Linux, Ubuntu Server, Windows Server**.
- Used in data centers and hosting services.

👉 Example: An **Ubuntu Server** can host a website or database that thousands of users access daily.

▪ Hypervisor

- A **Hypervisor** is software that allows multiple **virtual machines (VMs)** to run on a single physical machine.
- Each VM acts like a separate computer with its own OS.
- Types:
  - **Type 1 (Bare Metal):** Runs directly on hardware (e.g., VMware ESXi, Microsoft Hyper-V).
  - **Type 2 (Hosted):** Runs on top of another OS (e.g., Oracle VirtualBox, VMware Workstation).

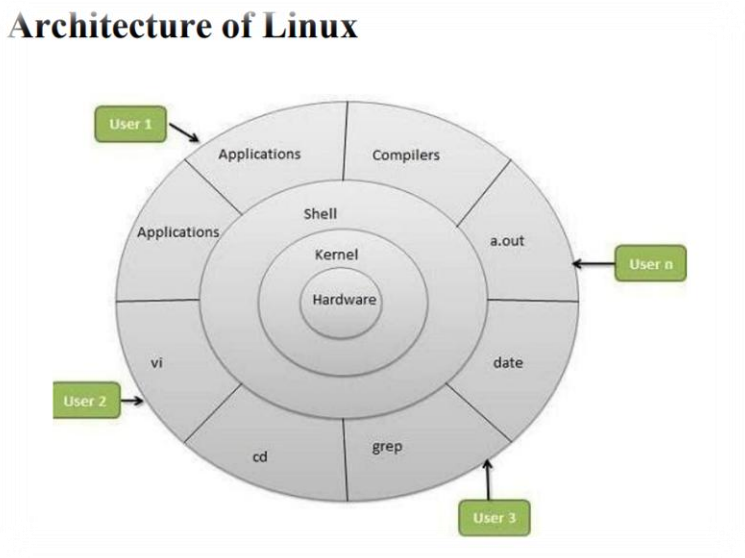
👉 Example: If you have 1 laptop with Windows OS, you can use VirtualBox (Type 2 hypervisor) to create multiple virtual machines like Ubuntu and Red Hat Linux.

Linux Notes

Principles of Linux:

1. **Open Source**
    - Anyone can view, modify, and distribute the code.
    - Promotes community contribution.
2. **Everything is a File**
    - In Linux, hardware devices, programs, and data are all treated like files.
3. **Small, Simple Tools**
    - Linux has many small programs that each do one job well (e.g., ls, cp, grep).
4. **Chain Programs Together**
    - You can combine simple commands using pipes (|) to perform complex tasks.
5. **Security and Permissions**
    - Users have limited access unless they are "root" (admin).
    - File permissions control who can read, write, or execute files.
6. **Multitasking and Multiuser**
    - Linux can run many programs at once and supports multiple users.
7. **Portability**
    - Linux can run on different types of devices — from PCs to servers to mobile phones.

Linux Notes



Shell

- Interface between user and OS.
- Executes user commands (default: bash).
- Stores and interprets commands.

Kernel

- Handles system calls and resources.
- Manages hardware like CPU, memory, I/O.

Hardware

- Actual physical devices (RAM, CPU, peripherals).

Application

- User-level programs (Word, Email, Browser).

- ### What is VPN?

  - **VPN (Virtual Private Network)** is like a **secure tunnel** between your computer (or phone) and the internet.
  - Instead of connecting directly to websites, your data first goes to a **VPN server**.
  - This VPN server then forwards your request to the internet.

Benefits of VPN:

Linux Notes

1. **Encryption** – Your data is scrambled so that hackers cannot read it.
2. **Hide IP Address** – Your real location is hidden; the website sees the VPN server's IP instead.
3. **Bypass restrictions** – You can access blocked websites or apps in your country (like unblocking Netflix US from India).
4. **Security** – Protects your data when using public Wi-Fi (like in airports or cafes).

👉 **Example:** If you connect to a VPN server in the US while you are in India, websites will think you are browsing from the US, not India.

Who are Developers?

- A **developer** is a person who **creates software** (programs, apps, games, or systems).
- Their main job is to **design, build, and maintain** computer applications.

◇ **Types of Developers:**

1. **System Developers** – Build the **foundation software** like operating systems (Linux, Windows).
2. **Application Developers** – Build apps like WhatsApp, Instagram, or MS Word.
3. **Game Developers** – Create games (PUBG, Minecraft, etc.).
4. **Web Developers** – Build websites and web applications.
5. **Mobile Developers** – Build Android/iOS apps.

👉 **Example:**

- A **web developer** builds Flipkart's website.
- A **mobile developer** creates the Flipkart app.
- A **system developer** builds the Android OS that runs the app.

▪ **Stages of Developers:**

1. **Single User, Single Tasking**
   - **What it means:**
     - Only **one user** can use the system.

- Only **one task/program** runs at a time.

- **Example:**
  - **MS-DOS (Disk Operating System)**.
  - If you are running a program (like playing music), you cannot open another task (like editing a text file) at the same time.

2. **Single User, Multi-Tasking**

- **What it means:**
  - Only **one user** can use the system at a time.
  - But the user can run **multiple tasks/programs simultaneously**.
- **Example:**
  - **Windows 10, macOS**.
  - You can listen to music , browse the internet , and write a document at the same time.

3. **Multi-User, Multi-Tasking**

- **What it means:**
  - **Multiple users** can use the system **at the same time**.
  - Each user can run **multiple tasks simultaneously**.
- **Example:**
  - **Linux, Unix, Mainframes, Windows Server**.
  - On a Linux server:
    - One user can run a database.
    - Another user can host a website.
    - Another user can run backups.
  - All at the same time, without disturbing each other.

Linux Notes

- Difference Between Windows, Unix & Linux

WINDOWS	UNIX	LINUX
Propritary based	Propritary based	Community based
Costly	Costly	Free of cost
Less secure	Highly secure	Highly secure
Closed source	Closed source	Open source
Heavy hardware	Less hardware	Less hardware
Non – portable	Portable	Portable
Easy to use (GUI)	Hard to use (CLI)	Hard to use (CLI)
80% desktops use windows	90% approx. Server use UNIX	90% approx. Server use LINUX

- Virtual Machine creation steps:

1. Download Vbox according to your system need

   VirtualBox

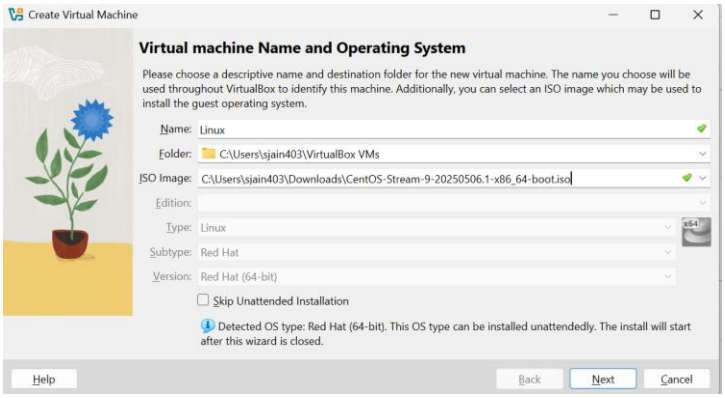
2. Inastall Vbox

3. Now Download ISO Image for centos 9

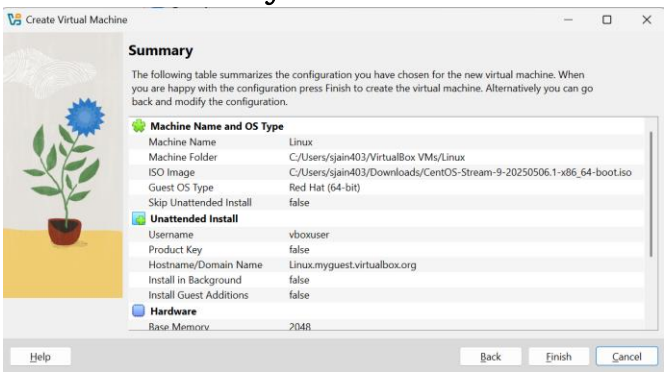
   CentOs9

4. Open Vbox and click on New and select downloaded ISO image

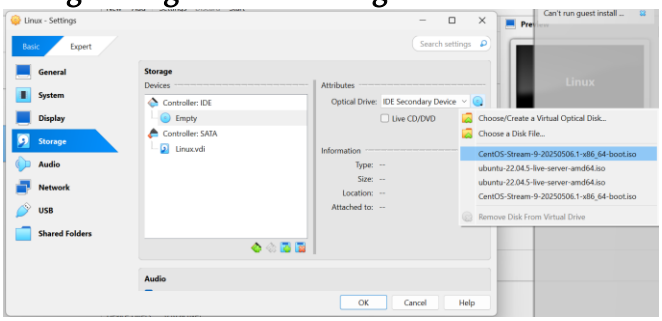
Linux Notes



Click next and click on finish

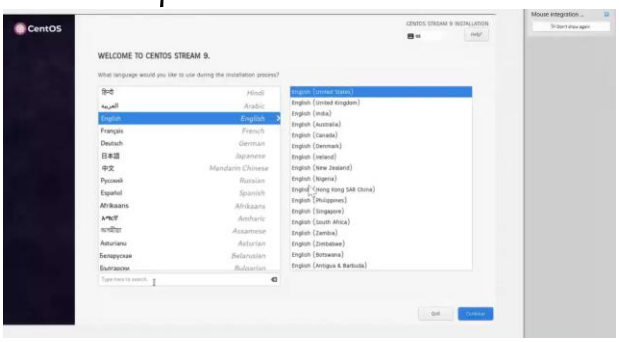


5. Setting->Storage->select ISO Image



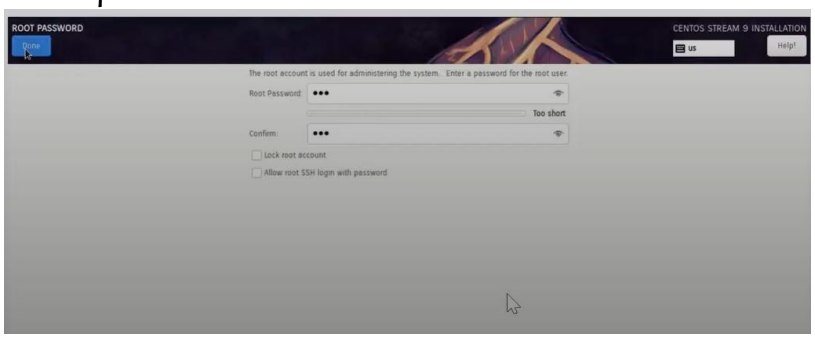
Now you are good to start machine

6. Select below options

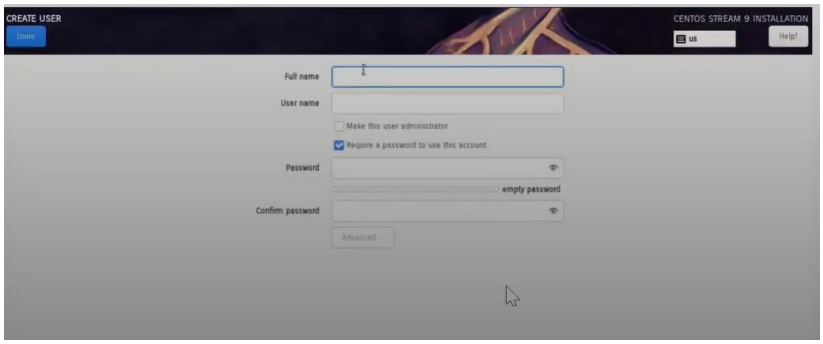


Linux Notes

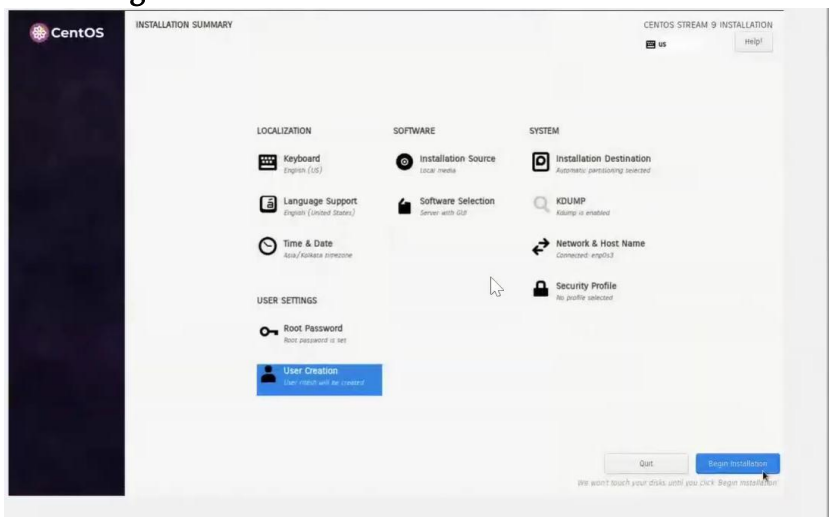
Set root password



Create user



Click on begin installation



After Completing installation, you need to reboot system

Basic Linux Commands

Command	Description
tty	Shows terminal number

Linux Notes

who	Displays logged-in users
whoami	Shows current username
w	Shows who is logged in & what they are doing
wh<Tab><Tab>	Lists similar commands
clear / Ctrl+L	Clears screen only
logout / Ctrl+D	Log out of terminal
lspci	Show PCI devices
lsusb	Show USB devices
Ctrl+Alt+F3 / Ctrl+Alt+F1	GUI ⟶ CLI and back
dmidecode	Hardware info (root only)
uname	Shows kernel information
man	Opens manual
uname / uname -a / -r / -v	Kernel info
ls --help	Command help
cd ..	Go back to previous dir
useradd username	Create new user
passwd username	Set password for user
hostnamectl	Show system info
free -h	Show RAM info
info	Similar to man
whatis	One-line manual description
which <command>	Show path of command

Prompt Structure

Linux Notes

[root@server~]#

- root ⟶ current user
- @server ⟶ hostname
- ~ ⟶ present directory
- # = root, $ = Local user

Calendar and Date Commands

Command	Description
cal	Current month calendar
cal <year>	Full year calendar
cal <month> <year>	Specific month/year calendar
cal -3	Previous, current, and next month
cal -j	Julian days of current month
cal <year> -j	Julian days of year
date	Current date/time
date -s "DD-MM-YYYY HH:MM:SS"	Set date/time
reboot / init 6	Restart system
poweroff / init 0	Shutdown system

Shutdown Commands

Command	Description
shutdown	Shutdowns system after 1 minute
shutdown -h 10	Shutdown after 10 min with broadcast message
shutdown -c	Cancel scheduled shutdown and notify users
shutdown now	Immediate shutdown
shutdown -h	Immediate shutdown

Linux Notes

logout / exit	Logout from current user session
Ctrl + D	Logout from terminal

More Linux Commands

Command	Description
mkdir	Make new directory
touch	Create empty file
cd	Change directory
ls	List contents
ll	Long list format
mv	Move or rename files
pwd	Print current working directory
cp	Copy files or directories
echo	Display text or variable value
grep	Search/filter text
;	Combine multiple commands on one line
rm	Remove files
rmdir	Remove empty directories
du -sh	Show file size
df -h	Show storage usage
top	View real-time processes
history	Show command history
rm -rvf	Force delete directories/files recursively
bash	Start a new shell

Linux Notes

hostname <name>	Set temporary hostname
echo $SHELL	Show current shell
lscpu	Show CPU info
man lscpu	Manual for lscpu
su	Switch to root user
arch	Show system architecture

- **Most used Linux distros currently in IT industry.**
  - RPM based:- RHEL & Centos
  - Debian based :- Ubuntu Server.

- **How to create files?**

1. **touch Command:** Used to create an empty file or update the timestamp of an existing file.

   Examples: touch file1.txt # Creates file1.txt

   touch a{1..10} # Creates multiple files(10 files) at once

   touch existing.txt # Updates the timestamp (modification time)

2. **cat Command**: Used to display the contents of a file, combine files, or even create a new file.

   Redirection Symbols:

Symbol	Meaning
>	Redirect output (overwrite)
>>	Redirect output (append)
2>	Redirect error messages (overwrite)
2>>	Redirect error messages (append)
&>	Redirect both output and error (overwrite)
&>>	Redirect both output and error (append)

Example:

cat -n filename # Displays the file with line numbers.

cat file1 2> errorsfile # If file1 doesn't exist or errors occur, errors are saved in errorsfile (old errors are overwritten).

cat file1 2>> errorsfile # Saves new errors at the end of errorsfile (old errors are kept).

cat file1 &> file2 # Redirects both output and errors into file2, replacing old content.

cat file1 &>> file2 #Appends both output and error into file2, without deleting existing data.

3. **vim Editor:** A powerful text editor used for writing and editing code or text inside the terminal.

1. Normal Mode (Default Mode)

- **Common Normal Mode Commands:**

h ← move left

l → move right

k ↑ move up

j ↓ move down

gg → goes to up (first line)

G → goes to down

dd → delete a line

n dd → delete number of lines

dw → delete word

n dw → delete number of words

yy → copy (yank) a line

n yy → copy number of lines

yw → copy words

n yw → copy number of words

cc → cut a line and go to insert mode

n cc → cut number of line

Linux Notes

p ⟶ paste after the cursor

u ⟶ undo

Ctrl + r ⟶ redo

2. Insert Mode

Use this mode to **start typing text**, like any regular editor.

◇ **Enter Insert Mode from Normal Mode:**

- i ⟶ Insert before the cursor (left Side)
- I ⟶ Insert at beginning of line
- a ⟶ Append after cursor (Right Side)
- A ⟶ Append at end of line
- o ⟶ Open a new line below
- O ⟶ Open a new line above
- r ⟶ Replace single alphabet/character/digit
- R⟶ Replace multiple alphabet/words

📌 Press Esc to go **back to Normal Mode**.

3. Executive Mode:

- Save files
- Quit Vim
- Search and replace
- Set settings

◇ **Enter by pressing: in Normal Mode**

(You'll see : at the bottom)

◇ **Common Commands:**

:w ⟶ Save

:q ⟶ Quit

:wq ⟶ Save and quit **OR** :x ⟶ Save and quit

Linux Notes

:q! ⟶ Quit without saving

:set nu ⟶ Show line numbers(temporary)

:set nonu ⟶ Hide line numbers

4. Visual Mode

Used to **select text** to copy, delete, or format.

◇ **Enter from Normal Mode:**

- v ⟶ Select character-wise
- V ⟶ Select whole lines
- Ctrl + v ⟶ Block (column) selection

After selecting, you can:

- Press **d** to delete
- Press **y** to copy (yank)
- Press **p** to paste

▪ Copy, Move and Rename Commands:

1. **Copy Command — cp** Used to copy files or folders.

cp source destination

Examples:

cp file1.txt file2.txt # Copies file1.txt ⟶ new file2.txt

cp file.txt /home/user/ # Copies file.txt to another folder

2. Move Command — mv Used to **move or rename** files or folders.

mv source destination

Examples:

mv file1.txt /home/user/ #Move files to different folder

mv folder1 /home/user/ #Move a Folder

mv file1.txt file2.txt #Renames file1.txt to file2.txt

3. Rename File or Folder There is no separate rename command (except in some systems). You rename by using *mv*

Examples: *mv oldname.txt newname.txt*

mv oldfolder/ newfolder/

■ Read Operation in Vim & Terminal:

cat – Simple file output

cat file.txt

- ◇ Displays the whole content of the file.

cat -n filename – It used to read existing file in terminal.

less – Scrollable file viewer

less file.txt

- ◆ You can scroll with:
  - Up/Down Arrow
  - Space for next page
  - q to quit

more – divided into pages

- *more file.txt*
- Similar to less, shows file **page by page.** Data show in % (per page how much data contain from all data which file have)

head Command -- Shows the first few lines of a file (default is 10 lines).

Examples: *head file.txt*

head -20 file.txt

tail Command -- Shows the last few lines of a file (default is 10 lines).

Examples: *tail file.txt*

tail -3 file.txt

■ User Management:

Linux Notes

What is a User in Linux?

A **user** is someone who uses the system. Just like Windows has different login accounts, Linux has users like:

- **root (superuser/admin) -** Full command access (UID 0)
- **system users (created by the system for special services) -** No login shell (UID 1 to 999)
- **Local users (like you and me)** - has limited permission (UID 1000 to 60000+)

Difference between useradd & adduser Linux commands

useradd command	adduser command
It is a low-level native binary (compiled).	It is a high-level **Perl script** that uses useradd in the background.
Does **not** provide an interactive prompt.	More interactive & user-friendly.
It is a **soft link** to adduser.	adduser is not present in all Linux distributions.
Creates a /home directory with the -m option for new users.	Creates the /home directory by default.
Adds new user to multiple groups at the same time.	Needs separate command to add user to multiple groups.
Does not ask for extra details.	Asks for details like full name, roll number, work info, etc.

/etc/passwd File Format Example

sakshi:x:1000:1000:/home/sakshi:/bin/bash

5 Dependencies of a User

1. UID – User ID
2. GID – Group ID
3. Skeleton File – From /etc/skel (default user files)
4. Home Directory – /home/username
5. Mail Directory – /var/mail/username

User Management Commands

adduser username

passwd Rahul # To set a password

Task	Command Example
Change UID	usermod -u 1005 shiv
Change GID	usermod -g 1001 shiv
Add comment	usermod -c "linux user" shiv
Change home dir	usermod -d /home/shree shiv
Change shell	usermod -s /bin/bash shiv
Lock user	usermod -L radha
Unlock user	usermod -U radha

userdel – Delete a User

This command is used to **remove a user** from the system.

Example: userdel Ramesh

userdel -r Ramesh # -r = remove home directory and mail files too(Remove with dependencies)

Skeleton Files

The **skeleton directory** in Linux stores **default files & folders** that are copied to a new user's home directory during account creation. Path: /etc/skel/

Examples of skeleton files:

1. **.bash_profile**
    - Hidden script file with custom configurations.
    - Executes automatically in **bash interactive login shells**.
2. **.bashrc**
    - Hidden script file for **terminal session configuration**.

- Executes in **both interactive & non-interactive shells**.

3. **.bash_logout**
   - Runs when a login shell exits.
   - Cleans up user environment after logout.
   - Stored in user's home directory ($HOME/).

▪ Shadow File:

File Location: /etc/shadow

What It Contains

Each line in /etc/shadow contains **one user's** info in this format:

username:encrypted_password:last_change:min:max:warn:inactive:expire

Example:

radha:6sdfgB89kL....:19510:0:99999:7:::

Fields:

Field	Meaning
1	Username (radha)
2	Encrypted password (6...)
3	Last password change (days since 1970)
4	Minimum days before password change
5	Maximum days password is valid
6	Warning days before expiry
7	Inactive days after expiry
8	Account expiry date

Only **root user** can read this file for security reasons.

cat /etc/shadow

Linux Notes

Useful Shadow File Commands:

View Password Expiry Info-> chage -l radha

Set Password Expiry to 60 Days -> chage -M 60 radha

Set Warning Before Expiry to 7 Days -> chage -W 7 radha

Set Account Expiry Date - > chage -E 2025-12-31 radha

Lock a User Account-> passwd -l radha

Unlock a User Account-> passwd -u radha

▪ Group Management:

What is Group Management in Linux?

In Linux, a **group** is a collection of users. Groups are used to **manage permissions** (like read, write, execute) for **multiple users together.**

Difference between Primary Group & Secondary Group

Primary Group	Secondary Group
Created when the user is created.	Other existing groups in the system.
A user can have only **one** primary group.	A user can have multiple secondary groups.
Can be changed using usermod.	Can be added/removed with usermod or useradd.
Listed in /etc/passwd file.	Listed in /etc/group file.
Every user **must** belong to a primary group.	A user can belong to **up to 15** secondary groups.
Defines default group for new files created by user.	Defines extra groups for resource sharing.

▪ **Group Info Files:**

File	Purpose
/etc/group	Stores group info
/etc/gshadow	Stores encrypted group passwords

Linux Notes

- Group Management Commands:
  - **groupadd – Create a New Group**

    Create a Group: groupadd developers

    View All Groups: cat /etc/group
  - **gpasswd – Manage Group Membership & Password**

    Set a Group Password: gpasswd developers

    Add a User to a Group: gpasswd -a rahul developers # Adds user rahul to the developers group.

    Remove a User from a Group: gpasswd -d rahul developers

    Add a Multiple user to a group: gpasswd -M Sakshi, Gunjan, Sanika devlopers

    Add Admin to a group: gpasswd -A Mentore developers

    Remove password: gpasswd -r devloper

    groupmod – Modify an Existing Group

    Rename a Group: groupmod -n devteam developers

    Change Group ID (GID): groupmod -g 1050 devteam
  - **groupdel – Delete a Group**

    groupdel devteam

Why Hide Passwords and Group Info?

In early UNIX systems, user passwords were stored **directly** in the /etc/passwd file — which is readable by everyone! ❌

To improve security, Linux now stores encrypted passwords in a **separate file** (/etc/shadow) that **only root can read**. ☑

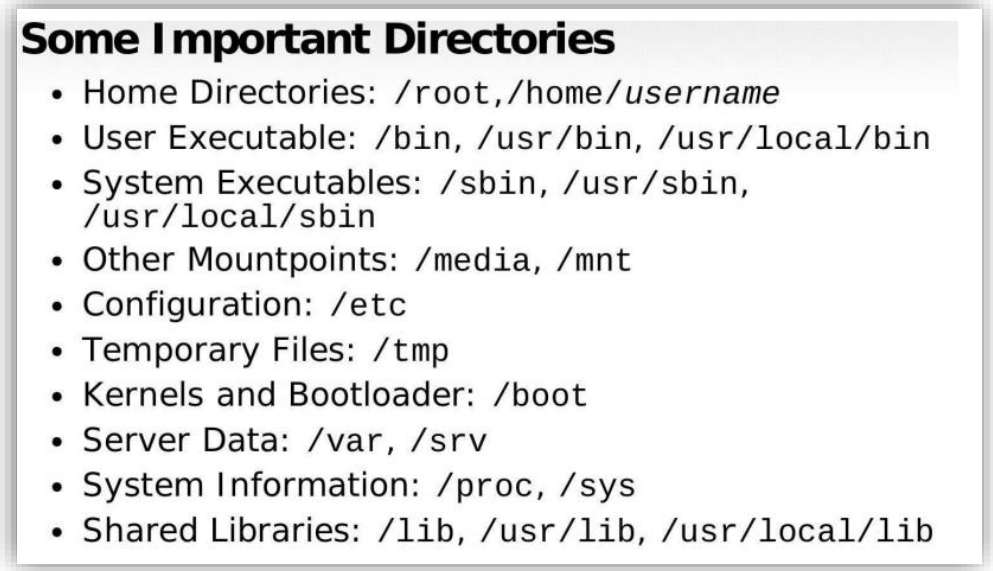
Same goes for group passwords (/etc/group → /etc/gshadow)

Linux Notes

Commands to Hide & Unhide

Purpose	Command	What it Does
Move password to /etc/shadow	**pwconv**	Hides encrypted passwords for security
Move password back to /etc/passwd	**pwunconv**	Unhides — puts passwords back in /etc/passwd ❌ (Not recommended)
Move group password to /etc/gshadow	**grpconv**	Hides group passwords securely
Move group password back to /etc/group	**grpunconv**	Unhides group passwords (back to /etc/group) ❌

▪ Directories in Linux OS



1. / (Root)

Top-most directory in Linux.

Everything starts from here.

Like *C:* drive in Windows.

2. /bin (Binary)

Linux Notes

Contains essential Linux commands used by all users (e.g., ls, cp, mv).

These commands are needed even in single-user mode.

3. /sbin (System Binary)

Contains system commands used by root only (e.g., shutdown, reboot).

Used for system maintenance.

4. /usr

Stands for Unix System Resources.

Contains installed software and libraries for users.

/usr/bin: user commands

/usr/lib: libraries

/usr/share: documentation, icons, etc.

5. /usr/local

For locally installed software (not from Linux distro).

Keeps custom tools separate from system ones.

6. /boot

Contains files needed for booting Linux.

Like the GRUB bootloader and kernel (vmlinuz).

7. /etc

Contains configuration files (not programs).

E.g., /etc/passwd, /etc/hostname, /etc/network/interfaces.

8. /dev (Device)

Contains files for hardware devices.

E.g., /dev/sda (hard disk), /dev/usb.

9. /proc

Contains virtual files that show real-time system info.

Example: /proc/cpuinfo, /proc/meminfo.

10. /sys

Like /proc, but for hardware configuration.

Virtual files used by kernel.

11. /var

Stands for variable files – they change frequently.

Stores logs, emails, spool files.

E.g., /var/log, /var/mail.

12. /tmp

Temporary files created by applications.

Often cleared on reboot.

13. /lib

Contains essential shared libraries (.so files).

Needed by commands in /bin and /sbin.

14. /lib64

64-bit versions of libraries.

15. /opt

Used to install optional software/packages.

Mostly 3rd party apps (e.g., Google Chrome, VMware).

16. /media

Used for auto-mounted removable media (CDs, USBs).

E.g., /media/usb.

17. /mnt

Used to manually mount drives.

Temporary mounting point.

18. /home

Contains home directories of users.

Example: /home/rahul, /home/sakshi.

19. /root

This is the home directory of root user.

Different from / root of the file system.

- ## Permission in Linux
  - ### Types of files:
    - #### 1. User-Defined Files

      These are files created by users for storing data, scripts, etc.

      ##### 1.1 Regular File

- Definition: Normal files that store text, images, programs, etc.
- Subtype:
  - Text file – .txt, .sh, .py
  - Binary file – compiled programs like /bin/ls
- Command to identify: ls -l
- Symbol: - (dash at the beginning)
- Example: myfile.txt, script.sh

1.2 Directory File

- Definition: Special file that holds other files/folders.
- Command to identify: ls -l
- Symbol: d
- Example: /home, /etc, /usr

1.3 Symbolic Link (Soft Link)

- Definition: Shortcut to another file or directory.
- Created using: ln -s source target

Linux Notes

- Symbol: l
- Properties:
  - Has different inode than original
  - If original is deleted, link becomes broken
- Example: ln -s file.txt link.txt

1.4 Hard Link

- Definition: Another name for the same file (same inode).
- Created using: ln source target
- Symbol: - (like regular file)
- Properties:
  - Same inode
  - Still works even if the original file is deleted
- Example: ln file1 file2

◇ 2. System-Defined Files

These are special files created by the system to manage hardware, processes, and inter-process communication.

2.1 Block Device File

- Definition: Used for devices that store data in blocks (e.g., hard drives, USBs).
- Symbol: b
- Example: /dev/sda, /dev/sdb1
- Check with: ls -l /dev/sda

2.2 Character Device File

- Definition: Used for devices that transfer data one character at a time.
- Symbol: c
- Example: /dev/tty (terminal), /dev/random

2.3 Socket File

- Definition: Used for process-to-process communication over the network.
- Symbol: s
- Example: /var/run/docker.sock, /tmp/mysocket

2.4 Pipe File (FIFO)

- Definition: Used for communication between processes, First-In-First-Out.
- Symbol: p
- Created using: mkfifo
- Example: /tmp/myfifo

Structure of a File Fieldes

ls -ltr filename

-rw-r--r-- 1 user group size date/time filename

Field	Meaning
-rw-r--r--	**Permissions**
1	**Number of links** (Hard links to the file)
user	**Owner** of the file
group	**Group** to which the file belongs
1284	**File size** in bytes
Jul 3 10:00	**Last modification date and time**
file.txt	**File name**

Linux Notes

Permissions in Linux

Linux gives 3 types of users certain **permissions** to files and directories:

User Type	Symbol	Example Permissions
Root	u	rwx (read, write, execute)
Group	g	r-x (read, execute only)
Others	o	r-x (read, execute only)

How to Give Permissions to Users

Each permission has a number (used for symbolic & numerical representation):

Permission	Symbol	Value
Read	r	4
Write	w	2
Execute	x	1

Example: rwx = 4 + 2 + 1 = **7**

Commands That Use These Permissions

These commands need or are affected by permissions:

- r (read) ⟶ Used by commands like: cat, vim, less, more, tail, head
- w (write) ⟶ Used by: touch, mkdir, create, edit, delete files
- x (execute) ⟶ Run or enter: ./file.sh, cd folder

Symbols Used to Give Permissions

Symbol	Meaning
+	Add permission
-	Remove permission

Symbol	Meaning
=	Assign specific permission

Symbols for Users

Symbol	Meaning
u	User (owner/root)
g	Group
o	Others
a	All (u + g + o)

Examples using chmod (symbolic method)

```
chmod ugo+rwx /vaishnavi/     # Give all permissions to all users
chmod ugo-x /vaishnavi/       # Remove execute permission from all
chmod u-rwx /vaishnavi/       # Remove all permissions from user
chmod ugo=x /vaishnavi/       # Give only execute permission to all
```

Numerical Method to Give Permissions

Each permission is written as a **number (octal method)**.

For Directory:

rwx rwx rwx = 777

This gives **all permissions** to:

- user (7)
- group (7)
- others (7)

For File:

rw- rw- rw- = 666

This gives **read and write** to:

- user (6)
- group (6)
- others (6)

Example:

chmod 777 dell # Gives full permission to everyone on file/folder 'dell'

▪ File change type

1. umask (User Mask)

What is it?

umask is a **default permission subtractor**. It decides **what permissions will *not* be given** when a new file or folder is created.

- Files are created with default max permission: **666** (rw-rw-rw-)
- Directories get: **777** (rwxrwxrwx)
- umask subtracts values from these defaults.

Example:

umask 022

touch file1

Explanation:

- 666 - 022 = 644 ⟶ rw-r--r-- So, file1 will have **read/write for owner**, and **read-only for group & others**

☑ **Check current umask:** umask

☑ **Set temporarily:** umask 027

2. Special Permissions

Special permissions are used to:

- **Allow non-root users to execute programs with temporary root privileges (SUID).**
- **Ensure group ownership inheritance in shared directories (SGID).**
- **Protect files in public directories from being deleted by others (Sticky Bit).**

Linux Notes

Linux has **three special permissions**:

Type	Symbol	Applies To	Purpose
SUID	s	Files	Run file as file owner
SGID	s	Files/Dirs	Run as group / new files get group of dir
Sticky Bit	t	Directories	Only owner can delete own files

SUID (Set User ID)

When a file has **SUID**, any user who runs that file will **temporarily gain the file owner's permissions**.

Example:

ls -l /usr/bin/passwd

chmod u+s filename

chmod u-s filename #remove SUID

SGID (Set Group ID)

There are **two uses**:

1. On a **file**: The file runs with the permissions of the **group owner**.
2. On a **directory**: Any new file/folder created inside **inherits the group** of that directory.

Example:

```
[root@vbox ~]# groupadd practice3
[root@vbox ~]# gpasswd -M Raksha,Mentore,A1 practice3
[root@vbox ~]# mkdir /Task3
[root@vbox ~]# chgrp practice3 /Task3
[root@vbox ~]# ls -ltr -d /Task3
drwxr-xr-x. 2 root practice3 6 Jul  2 19:04 /Task3
[root@vbox ~]# chmod 770 /Task3
[root@vbox ~]# ls -ltr -d /Task3
drwxrwx---. 2 root practice3 6 Jul  2 19:04 /Task3
[root@vbox ~]# su - Raksha
[Raksha@vbox ~]$ cd /Task3
[Raksha@vbox Task3]$ touch Rakshafile
[Raksha@vbox Task3]$ ll
total 0
-rw-r--r--. 1 Raksha Raksha 0 Jul  2 19:06 Rakshafile
[Raksha@vbox Task3]$
logout
[root@vbox ~]# su - Mentore
[Mentore@vbox ~]$ cd /Task3
[Mentore@vbox Task3]$ cd /Task3
[Mentore@vbox Task3]$ touch Mentorefile
[Mentore@vbox Task3]$ ll
total 0
-rw-r--r--. 1 Mentore Mentore 0 Jul  2 19:08 Mentorefile
-rw-r--r--. 1 Raksha  Raksha  0 Jul  2 19:06 Rakshafile
[Mentore@vbox Task3]$ touch Mentore2
[Mentore@vbox Task3]$ ll
total 0
-rw-r--r--. 1 Mentore Mentore 0 Jul  2 19:08 Mentore2
-rw-r--r--. 1 Mentore Mentore 0 Jul  2 19:08 Mentorefile
-rw-r--r--. 1 Raksha  Raksha  0 Jul  2 19:06 Rakshafile
[Mentore@vbox Task3]$ rm -rvf Rakshafile
removed 'Rakshafile'
```

Sticky Bit

It allows only the file owner or root to delete or rename files inside a directory.

Example:

chmod o+t file/dir name

3. ACL (Access Control List)

ACLs is used to assign special permission to **specific users or groups** *to read, write, execute a file or directory without changing the base ownership and permission .*

☑ **Why Use ACL?**

- *When you want to give a* **specific user or group** *access* **without changing ownership**.
- *Useful for* **shared environments** *where traditional permissions are not sufficient.*

Command	Description
setfacl -m u:username:perm file	Add/modify user ACL

Linux Notes

Command	Description
setfacl -m g:groupname:perm file	Add/modify group ACL
setfacl -x u:username file	Remove ACL for user
setfacl -b filename	Remove all ACL entries
getfacl filename	View current ACL entries

Example:

touch file1

setfacl -m u:john:r file1

getfacl file1

4. **Sudo (Superuser do):**

   It allows a **regular (non-root) user** to run **commands with root (administrator) privileges** — but **only when allowed**.

Why Do We Use sudo?

- To perform **admin-level tasks** like installing software, editing system files, restarting services, etc.
- Prevents giving full-time root access to users.
- Helps **log and control** who used administrative power.

Who Can Use sudo?

Users listed in the /etc/sudoers file or members of the sudo group.

▪ Job Scheduling and Automation in Linux

📌 What is Job Scheduling?

Job scheduling is a system process that allows tasks (commands or scripts) to be **automatically executed** at a specific **date and time**, either **once** or **repeatedly**.

It is **widely used for automation** of repetitive tasks such as:

- Backing up files
- Sending emails

- Running updates
- System maintenance

Types of Job Scheduling in Linux

Type	Description	Tool/Command Used
A	**Non-periodic** (one-time)	at
B	**Periodic** (repeating)	cron / crontab
C	**Used for missed tasks**	anacron

1) at Command (One-Time Scheduling)

Schedules a **single task** to run at a specific **future time**.

Required Service:

- The at command relies on the **atd** service.

Syntax: at HH:MM

Example:

at 20:53

> mkdir /demo

> touch /demo/file

> touch /file

> [Ctrl + D]

Supporting Commands

Command	Description
atq	List all scheduled at jobs
atrm <jobID>	Remove/cancel a scheduled job
date	View current system date & time

2) cron Command (Recurring jobs)

Used to schedule commands or scripts **repeatedly** at fixed times/dates.

Linux Notes

Why use cron?

- Automates repetitive tasks
- Runs jobs without user interaction

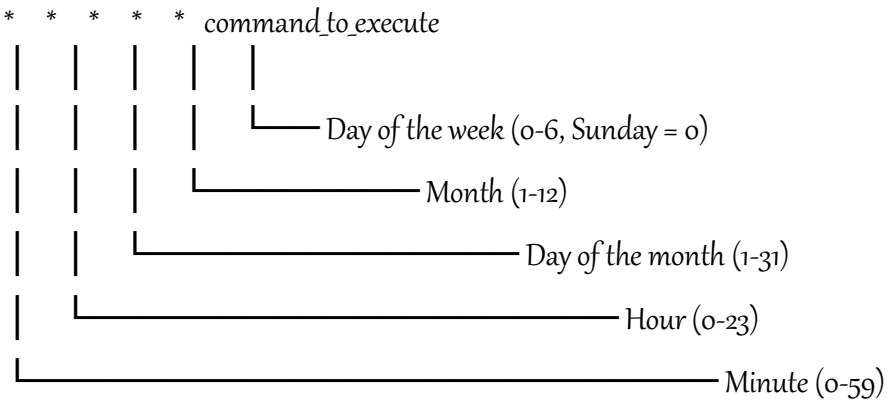
How does Cron Work?

- Cron uses a **daemon** (background service) called cron.
- Cron reads configurations from a **file called crontab** (cron table).
- **Service name**: crond

Check cron status:

systemctl status cron

Crontab Format (5 Fields + Command)



Crontab Commands

Command	Description
crontab -e	Edit crontab entries
crontab -l	List user's current cron jobs
crontab -r	Remove user's crontab

Example Cron Jobs

Run script every day at 6 AM

0 6 * * * /home/user/backup.sh

Run job every Monday at 12 PM

0 12 * * 1 /home/user/report.sh

Run every 10 minutes

*/10 * * * * /home/user/logrotate.sh

3) **anacron (For non-continuous systems)**- used in windows

Archiving and Compression

1. What is Archiving?

- **Archiving** is the process of combining multiple files into one.
- Common tool: tar (Tape Archive).
- Archiving **does not reduce file size**, just packs files together.

2. What is Compression?

- **Compression** reduces the size of files to save storage. This is specifically use for backup, storage, file transfer.
- Common compression tools:
  - gzip
  - bzip2
  - xz

TAR Command – Archiving: tar <options> <archive-file-name> <source-file>

Option	Description
-c	Create archive
-v	Verbose (shows progress)
-f	File name of archive
-r	Append files to archive
-t	List archive content
-x	Extract from archive

Linux Notes

Examples:

tar -cvf etc.file.tar /etc # Create archive

tar -rvf etc.file.tar file.txt # Add file to archive

tar -tvf etc.file.tar # View archive contents

tar -xvf etc.file.tar # Extract archive

Compression Types

Compression	Extension	Option	Unzip Command
gzip	.tar.gz	-z	gunzip
bzip2	.tar.bz2	-j	bunzip2
xz	.tar.xz	-J	unxz

Examples of Archive + Compression

tar -czvf etc.tar.gz /etc # gzip compression

tar -cjvf etc.tar.bz2 /etc # bzip2 compression

du -sh etc.tar # Check archive size

gzip etc.tar # Compress with gzip ⟶ creates etc.tar.gz

du -sh etc.tar.gz # Check compressed size

bzip2 etc.tar.gz # Further compress ⟶ etc.tar.gz.bz2

bunzip2 etc.tar.gz.bz2 # Decompress using bzip2

▪ Package Management (Software)

Package management is a **method of installing, removing, updating, and keeping track of software packages** from specific repositories in the Linux system.

✅ **Purpose:**

- Automates software installation
- Ensures version control

- Manages software dependencies
- Provides easy access to security patches and updates

What is a Repository?

A **repository** is a **storage location** where software packages are kept. These packages can be installed, updated, or removed using tools like yum, dnf, apt, etc.

What is Metadata?

Metadata is the **information about packages** available in the repository. It helps the system know:

- Package **name**
- **Version**
- **Dependencies**
- **Description**

Red Hat-based Package Management Tools

1. RPM (RedHat Package Manager)

- It is a **low-level** package manager with **basic functionalities**.
- Can install a **single package** at one time.
- **Does not handle dependencies** automatically.
- Does **not support automatic upgrades**.
- Used with .rpm packages.

Common RPM Commands:

Command	Description
rpm -q <pkg>	Query installed package
rpm -ivh <package.rpm>	Install a package (i = install, v = verbose, h = hash)
rpm -evh <pkg>	Remove a package
rpm -qa	List all installed packages
rpm -qi <pkg>	Get package information

Command	Description
rpm-qip <pkg>	Detail information of download packages
rpm -ql <pkg>	List files in a package
rpm -qd <pkg>	View documentation
rpm -qc <pkg>	Show config files

2. YUM (Yellowdog Updater Modified): Yum is primary package management tool for red hat. Yum performs dependency resolution when installing, updating, removing software packages. Yum can mange packages for install repositories in the system or from rpm package.

- It is a high-level front-end for RPM.
- Automatically resolves dependencies.
- Can install multiple packages at once.
- Allows automatic upgrades to latest versions.
- Uses online or local repositories to fetch packages.

Command	Description
yum install <pkg>	Install a package
yumdownloader <pkg>	Download package
yum remove <pkg>	Remove a package without dependencies
yum list all	List all available packages
yum list installed | grep <pkg>	Show package install or not
yum provides <pkg>	Find package that owns a file
yum upgrade <pkg>	Upgrade and replace a package
yum update <pkg>	Update while keeping old version rollback ready
yum autoremove <pkg>	**Remove package with dependencies**
yum repolist	**Show available repositories**

Linux Notes

Difference Between RPM and YUM

Feature	RPM	YUM
Type	Low-level tool	High-level tool
Dependency Mgmt	No automatic dependencies	Resolves dependencies automatically
Usage	One package at a time	Multiple packages and batch operations
Repositories	Not used	Uses online or local repos
Upgrade	Manual only	Can auto-upgrade packages

Debian-based Package Management Tools

1. DPKG (Debian Package)

- Low-level tool to manage .deb files.
- Does not resolve dependencies.
- Mainly used for **local installation**.

```
sudo dpkg -i <package.deb>    # Install .deb package
sudo dpkg -r <package>        # Remove package
sudo dpkg -l                  # List installed packages
```

2. APT (Advanced Package Tool)

- High-level tool used in Ubuntu/Debian.
- Resolves **dependencies automatically**.
- Uses online repositories (like YUM).

```
sudo apt install <pkg>        # Install
sudo apt remove <pkg>         # Remove
sudo apt update               # Update repo metadata
sudo apt upgrade              # Upgrade installed packages
```

How to Create a YUM Local Repository (Basic Steps)

Step 1: Prepare Directory and Download Packages

mkdir /matfly

cd /matfly

yumdownloader tree

yumdownloader httpd

yumdownloader mysql

Step 2: Create Repo Metadata

createrepo /matfly

cd /matfly

ls

repodata

Step 3: Create .repo File

cd /etc/yum.repos.d/

vim matfly.repo

```
[matfly-01]      #ID
name=matfly
baseurl=file:///matfly
enabled=1
gpgcheck=0
```

Step 4: Clean and List

yum clean all

yum repolist

Difference Between wget and curl

Feature	wget	curl
Purpose	Download files via HTTP, FTP	Transfer data using multiple protocols
Resume Support	Yes	Yes (with options)
Upload Support	No	Yes

Linux Notes

Feature	wget	curl
Syntax Simpler?	Yes	Slightly more complex
Output Format	Saves to file directly	Outputs to terminal by default
Example	wget http://file.com/a.txt	curl -O http://file.com/a.txt

▪ Linux Partition & Storage Management

Storage Types in Linux

1. **DAS (Direct Attach Storage)**
    - Devices directly connected to the server (e.g., USB, HDD, SSD).
2. **NAS (Network Attach Storage)**
    - Devices connected via a network. Acts like a shared folder.
3. **SAN (Storage Area Network)**
    - High-speed network storage (used in cloud like AWS, Azure, Google Cloud).

What is a Partition?

- A partition divides a hard drive into isolated sections.
- Each section behaves like its **own hard drive**.

Types of Partitions:

❖ **Primary Partition**

- Directly attached to OS.
- Can **boot the OS**.
- **Max 4 primary partitions** allowed.
- You can only **create 3 primary** if planning to add extended/logical ones.

❖ **Extended Partition**

- Container that holds logical partitions.
- **Only 1 extended** partition can be created.

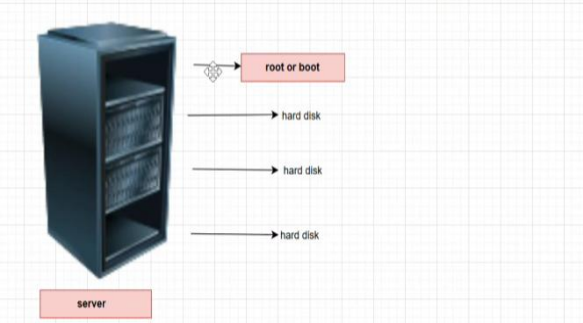
Linux Notes

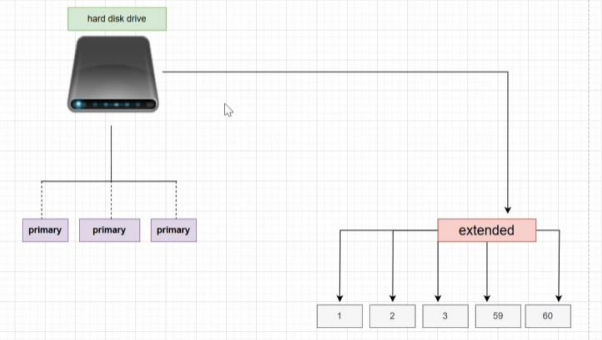
- Must be created to allow logical partitions.

❖ **Logical Partition**

- Exists **inside** the extended partition.
- **Max 60 logical partitions** allowed.

Example: Primary (3) + Extended (1) + Logical (60) = **64 partitions total**





What is a File System?

- A file system is a **method/structure** used by OS to **store, organize, access, and manage data** on a storage device (like SSD or HDD).

Types of File Systems:

OS	File System Types
Windows	FAT, NTFS(new technology file system)
Linux	ext2, ext3, ext4, xfs

inode

- An inode stores file info like modification date, time, permissions.

journaling

- Helps the system repair itself after improper shutdown or crash.

Partitioning in Linux Using fdisk

Step 1: Add New Hard Disk

1. Shut down CentOS machine.
2. Go to **Settings > Storage > Click on SATA + symbol > Add Hard Disk > OK**.

Step 2: Create Partition

fdisk /dev/sdb

Options in fdisk:

- n: create new partition
- d: delete partition
- e: extended partition
- l: logical partition
- w: write and save

Use lsblk to view partitions as:

- /dev/sdb1
- /dev/sdb2

Add File System

To format the partition and prepare it for use:

mkfs.ext4 /dev/sdb1

mkfs.ext4 /dev/sdb2

Use mkfs or mkfs -t to specify type (like ext4, xfs, etc.)

Mounting in Linux - mounting means **connecting a storage device (like hard disk, USB, CD, partition, etc.)** to a **folder in your file system** so that you can access and use it.

mount /dev/sdb1 /mnt

Other Useful Commands:

- mount -a ⟶ refresh mount file.
- du -hT ⟶ shows mount space.
- umount /mnt ⟶ unmount the file system.

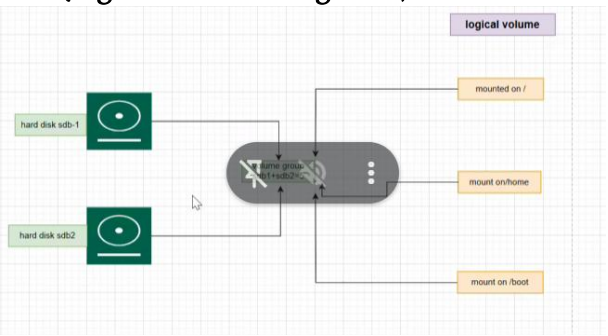
Permanent Mount

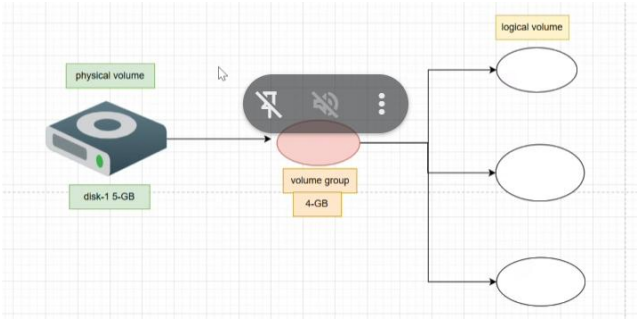
vim /etc/fstab ⟶ **/dev/sdb2 /mnt ext4 defaults 0 0**

Linux Notes

This ensures mounting persists after reboot.

LVM (Logical Volume Management)





What is LVM?

- LVM allows us to **manage storage space** dynamically.
- You can **extend, merge, shrink** storage **without deleting existing partitions**.
- Useful when you run out of disk space.

Physical Volume (PV)

- This is the **real hard disk** (or partition).
- Example: Your computer's hard disk or a USB drive.
- 👉 Think of it like a **bucket of raw storage**.

Volume Group (VG)

- A **collection of many PVs (disks/partitions)** combined together.
- It joins multiple small buckets (PVs) into one **big tank**.
- 👉 Example: If you have 2 hard disks of 10 GB each, VG will show them as **20 GB total storage**.

Logical Volume (LV)

- From that **big tank (VG)**, you can take out smaller **containers (LVs)**.
- Each LV acts like a **normal partition** that you can format and mount (e.g., /home, /boot, /data).
  - 👉 Example: From a 20 GB tank (VG), you can make:
    - 10 GB for /home
    - 5 GB for /boot
    - 5 GB for /data

Steps to Create LVM:

1. **Create Physical Volume (PV):**

pvcreate /dev/sdb1

pvcreate /dev/sdb2

2. **Create Volume Group (VG):**

vgcreate vg1 /dev/sdb1 /dev/sdb2

3. **Create Logical Volume (LV):**

lvcreate -L 5G -n lv1 vg1

4. **Format & Mount the LV:**

mkfs.ext4 /dev/vg1/lv1

mount /dev/vg1/lv1 /mnt

5. **Use it: Add files, etc.**

touch /mnt/file{1..5}

6. **Resize LV or VG if needed:**

lvextend -L +500M /dev/vg1/lv1

vgreduce vg1 /dev/sdc2

7. **Unmount and Remove Everything**

umount /mnt

lvremove /dev/vg1/lv1

vgremove vg1

pvremove /dev/sdc1 /dev/sdc2 /dev/sdc3

8. **Use fdisk to delete partitions**

fdisk /dev/sdc

Press d to delete each partition, then w to save

9. **Verify**

Linux Notes

lsblk

Networking

1. What is Networking?

Networking refers to the practice of connecting two or more computers or devices so that they can communicate and share resources like files, internet, printers, etc.

Example: Connecting your laptop and phone to the same Wi-Fi is a basic form of networking.

---

2. What is the Internet?

The Internet is a global network of interconnected computers that communicate using standardized protocols. It allows users to access and share information worldwide.

Simple: It's a giant web that connects millions of smaller networks.

---

3. Types of Networking:

There are different types of networks based on size and purpose:

Type	Full Form	Description
LAN	Local Area Network	Covers a small area like a home, office, or school. Fast and easy to set up.
MAN	Metropolitan Area Network	Covers a larger area like a city or a university campus.
WAN	Wide Area Network	Covers large geographical areas. The Internet is a global WAN.
PAN	Personal Area Network	Very short-range network, like connecting your phone to earbuds or smartwatch via Bluetooth.
WLAN	Wireless Local Area Network	Wireless version of LAN using Wi-Fi. Common in homes, offices, and cafes.
SAN	Storage Area Network	A high-speed network that connects servers to storage devices. Used in data centers.

Linux Notes

Type	Full Form	Description
VPN	Virtual Private Network	A secure tunnel created over public networks (like Internet) to access private networks safely.

---

4. Network Devices:

Devices used to build networks:

- **Router** – Connects different networks (e.g., home to internet)
- **Switch** – Connects multiple devices in a LAN
- **Hub** – Similar to switch but less intelligent
- **Modem** – Converts digital signals to analog and vice versa (used in internet access)
- **Access Point** – Provides wireless access to a wired network

---

5. What is NIC? RJ45(Male connector)

NIC stands for Network Interface Card. It is a hardware component inside a computer or laptop that allows it to connect to a network using a MAC (Media Access Control) address.

Example: Ethernet or Wi-Fi adapter.

---

6. What is OSI Model?

OSI stands for Open Systems Interconnection model. Developed by ISO (International Standardization Organization) in 1984, it explains how data travels from one computer to another over a network. It has 7 layers:

1. Application: Closest to the user. Interfaces with applications like browsers, email, etc.
2. Presentation: Translates, encrypts, and compresses data (e.g., formats like JPEG, PDF).
3. Session: Manages sessions (connections) between devices; starts, maintains, and ends communication.
4. Transport: Ensures reliable data delivery (TCP/UDP). Controls flow and error correction.
5. Network: Routes data between networks using IP addresses. (e.g., Routers)

6. Data Link: Deals with MAC addresses, error detection, and framing. (e.g., Switches)
7. Physical: Transmits raw bits (0s and 1s) over physical media like cables, Wi-Fi, etc.

👉 Each layer has its own function, like packaging, addressing, routing, etc.

📦 7. What is TCP?

TCP = Transmission Control Protocol. It is one of the main protocols in networking. It ensures reliable delivery of data between computers.

TCP/IP model has 4 layers (simplified from OSI):

1. Application
2. Transport
3. Network/Internet
4. Data Link & Physical (combined)

🌍 8. Difference between Public IP and Private IP:

Feature	Public IP	Private IP
Scope	Global	Local network only
Access	Accessible from Internet	Not accessible from outside
Example	8.8.8.8 (Google DNS)	192.168.0.1, 10.0.0.1
Provided By	ISP (Internet Service Provider)	Router or Admin in LAN
Uniqueness	Globally unique	Can be same in different LANs

Private IP ranges:

- Class A: 10.0.0.0 – 10.255.255.255
- Class B: 172.16.0.0 – 172.31.255.255
- Class C: 192.168.0.0 – 192.168.255.255

▪ Filter & Search Utility

Linux Notes

1. Grep Command

Full form: Global Regular Expression Print

Purpose: Searches for a specific keyword or pattern in a file and displays matching lines.

Syntax:

grep [options] "pattern" filename

Common Options:

- -i ⟶ Ignore case (uppercase/lowercase doesn't matter)

  Example: grep -i "root" file.txt
- -E ⟶ Use extended for multiple patterns.

  Example: grep -E "root|network" file.txt
- -n ⟶ Show line numbers where the match is found.

  Example: grep -n "root" file.txt
- -c ⟶ Show only the count of matching lines.

  Example: grep -c "root" file.txt
- Combine options (example: ignore case + extended Multiple Pattern):

  grep -iE "root|network" file.txt

2. Locate Command

Purpose: Finds files by name quickly using a pre-built index.

Example:

locate filename

locate filename | less # For scrolling results

Note: Database must be updated using: updated

3. Find Command

Purpose: Search for files and directories based on name, size, type, owner, etc.

Syntax:

find [path] [options] [expression]

Examples:

- Find by name:

find / -name filename

- Find by user ID:

find / -uid 1000

- Find by owner name:

  find / -user Sakshi

- Find directories only:

  find / -name sakshi -type d

Using Find with Copy Command

- Create backup directory:

mkdir /backup

- Copy files found by find:

find / -user sakshi -exec cp -rvf '{}' /backup \;

- Copy with **original ownership & permissions**:

  find / -user sakshi -exec cp -aprvf '{}' /backup \; #append and preserve

▪ Firewall

What is a Firewall?

A **firewall** is a **Linux security feature** that **monitors and controls network traffic**. It follows **rules** to allow or block specific traffic.

💡 **Purpose**: It provides **extra security** by **blocking unwanted access** and allowing only **authorized communication**.

Web Hosting Services

To host a website, you commonly use any of these 3 services:

- httpd – Apache web server

- tomcat – Java web server
- nginx – Lightweight, high-performance web server

Types of Firewalls

1. **Software Firewall**:
    - Installed on operating systems like Linux or Windows.
    - Example: firewalld in CentOS or Red Hat.
2. **Hardware Firewall**:
    - A separate physical device.
    - Placed between two networks (like between your office network and the internet).
    - Mostly used by IT/network teams for large-scale setups.

Tools for Managing Firewall in Linux:

Works in Linux distributions like CentOS, RedHat, Fedora.

To check if firewalld is installed

```
rpm -qa | grep firewalld
yum list install | grep firewalld
```

firewalld Service Management Commands

```
systemctl start firewalld     # Start firewall
systemctl enable firewalld    # Enable at boot
systemctl stop firewalld      # Stop firewall
systemctl disable firewalld   # Disable from auto start
systemctl restart firewalld   # Restart firewall
systemctl status firewalld    # Check status
```

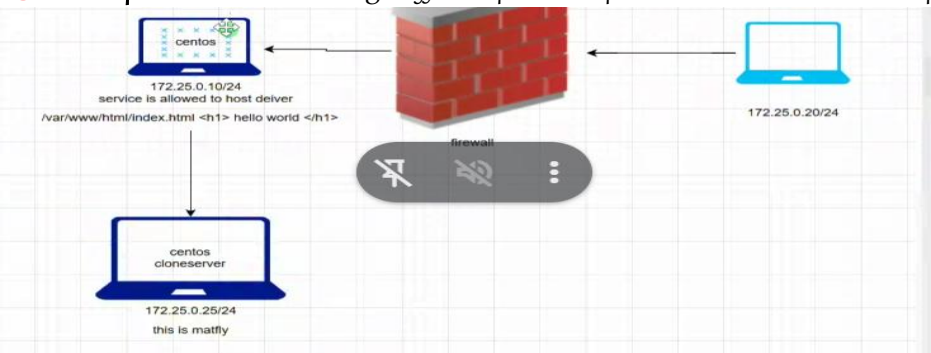
Port Forwarding (in Linux)

Definition:

Port forwarding is a **technique** to allow **external devices (like users on the internet)** to access services **inside your local/private network**.

Linux Notes

🎯 **Example**: Forward incoming traffic on public IP port 8080 to internal server port 80



IP Forwarding

IP Forwarding lets a Linux machine **act like a router**, meaning it can **forward packets between networks**.

📦 When a packet comes that's not for the system, IP forwarding **sends it to the correct destination**.

NAT & Masquerading

🔸 **What is NAT?**

NAT (Network Address Translation) translates **private IP addresses to public IP** and vice versa.

- Used when multiple devices share **one public IP**.

🔸 **What is Masquerade?**

Use to give network to other machine via IP address.

A type of NAT used in **dynamic IP** systems. It hides **private IPs** behind a **single public IP**.

▪ SSH (Secure Socket Shell)

SSH allows **secure remote login** to another system over the internet or network.

🔹 **Use:**

- Secure communication between two systems.
- Used to **access and control servers** remotely.
- Think of it like a remote control for a computer. You can be on your home computer (the client) and type commands to another computer somewhere else in the world (the server) as if you were sitting right in front of it.

- The "Secure" part is important: all the communication between the client and the server is **encrypted**, so no one can snoop on your commands or data.
- The default port for SSH is **22**.

Authentication Methods in SSH

1. **Password-based** – Enter password when connecting.
2. **Key-based** – Use **SSH keys** instead of passwords.

What is an "authorized key" in SSH?

An **authorized key** is part of a **public/private key pair** used in SSH (Secure Shell). It allows a user to connect to a remote system without typing a password.

Public Key vs Private Key

Feature	Private Key	Public Key
Ownership	Kept secret (by you only)	Can be shared with anyone
Use	Used to **decrypt** data	Used to **encrypt** data
File name	id_rsa	id_rsa.pub
Speed	Faster	Slower (used for short messages)

SCP (Secure Copy Protocol)

- Used to **copy files securely** over SSH:

scp file.txt user@IP:/home/user/

▪ What is Log Management?

Log Management is the process of collecting, storing, analyzing, and monitoring log data generated by systems, applications, devices, and networks.

Example: When you log in, when a file is accessed, when an error occurs, etc.

These logs come from:

- Applications (like Apache, cron)
- Operating System (like boot logs)
- Services (like mail, HTTP)

Types of Logs

Log File	Description
boot	Logs related to system startup
cron	Logs scheduled tasks (cron jobs)
secure	Logs login attempts & security info
maillog	Logs email activity (sendmail, etc.)
httpd	Logs web server activity (Apache)
messages	General system messages

Where Are Logs Stored?

- All logs are stored in this directory: cd /var/log

Important Services and Config Files

Item	Description
rsyslog.service	Service that manages and writes log files
Port Number	514 (used by rsyslog for log transmission)
Config File	/etc/rsyslog.conf (edit with vim)
View Boot Logs	journalctl (used to view logs from boot)

```
systemctl status rsyslog      # Check rsyslog service status
journalctl                    # View all logs since boot
vim /etc/rsyslog.conf         # Edit log config file
logger "your message here"    #create or write custom log messages
```

- **Log Management**

Linux Notes

Definition:

- Log management means collecting, storing, and managing logs generated by the system, applications, and services.
- Logs help in troubleshooting, monitoring system health, and security auditing.

Types of Logs:

- System logs (boot messages, kernel logs)
- Application logs (web server logs, database logs)
- Security logs (login attempts, firewall activity)

Log Storage Location:

- Most logs are stored in /var/log directory.

Important Files/Commands:

- **rsyslog service**: Handles log messages.
- **Port 514**: Used by rsyslog for remote logging.
- **Configuration file**: /etc/rsyslog.conf
- journalctl – View systemd logs.
- Journalctl -r : Shows recent logs.
- logger "message" – Create custom log entry.

Log Management Methods:

1. **Centralized Logging** – Collect logs from multiple machines on one server.
2. **Log Rotation** – Automatically archives, compresses, and deletes old logs to save space.
    1. **Centralized Logging** –

       Purpose: Store logs from multiple client machines in one central server.

       Steps:

       Server Machine

- Edit config:

vim /etc/rsyslog.conf

Uncomment **line 37 & 38**

Add:

$template tmplAuth,"/var/log/client/%HOSTNAME%/%PROGRAMNAME%.log"

. ?tmplAuth

- Restart service:

  systemctl restart rsyslog.service

- Create directory to store client logs:

  mkdir /var/log/client

- Firewall Setup:

  firewall-cmd --add-port=514/tcp --permanent

  firewall-cmd –reload

Client Machine

- Edit config:

  vim /etc/rsyslog.conf

  At line 82, add:

  . @@<Server-IP>:514

- Restart service:

  systemctl restart rsyslog.service

  Result: Server can see client logs inside /var/log/client.

2. **Log Rotation** –

Purpose: Automatically manage old logs by rotating, compressing, or deleting them to save space.

Config File: /etc/logrotate.conf or /etc/logrotate.d/<filename>

Example Configuration:

/root/panda/*.log {

weekly

missingok

rotate 4

compress

copytruncate

}

Options:

- weekly ⟶ Rotate logs every week
- missingok ⟶ Ignore if log file is missing
- rotate 4 ⟶ Keep 4 old log files
- compress ⟶ Compress old logs
- copytruncate ⟶ Copy content to old file and truncate original

Commands:

mkdir /root/panda

echo "this is panda" > /root/panda/file.log

logrotate -f /etc/logrotate.conf

gunzip file.log.gz # To unzip compressed log

Process Management

Definition:

- A process is a running instance of a program.
- Linux can run many processes at the same time.

Process Information:

- PID – Process ID (unique number for each process)
- TTY – Terminal from which process started
- TIME – CPU time used by process

- **CMD** – Command that started the process

Useful Commands:

- ps -e – Show all processes.
- ps -ef – Full process info.
- ps -aux – Show processes in BSD format.
- top – Real-time process monitoring (CPU, memory usage).
- jobs – Show active jobs.
- bg – Resume job in background.
- fg – Resume job in foreground.

Kill Command:

- kill <PID> – Stop a process.
- kill -9 <PID> – Forcefully stop a process.
- kill -15 <PID> – Gracefully stop a process.
- kill -2 <PID> – Stop using keyboard interrupt (Ctrl+C).

Example:

```
ps -ef | grep httpd     # Find process by name
yum install httpd       # Install Apache web server
systemctl start httpd   # Start service
kill 1234               # Stop process with PID 1234
```

nice and renice – Process Priority Control

Every process in Linux has a **priority**.

- **Lower nice value → higher priority** (runs faster, gets more CPU time).
- **Higher nice value → lower priority** (runs slower, polite to other processes).
- Range: **-20** (highest priority) to **19** (lowest priority).
- Default: **0**.

Linux Notes

nice – Start a Process with Priority

Example: Run a CPU-heavy script with low priority:

- nice -n 10 ./backup.sh #-n 10 ⟶ lower priority than normal (system will give CPU to other processes first).

renice – Change Priority of Running Process

Example: Increase priority of a running process:

ps -ef | grep backup.sh

Suppose PID = 2345

sudo renice -n -5 -p 2345

-5 ⟶ higher priority (process runs faster).

sudo needed if decreasing nice value (increasing priority).

Tip to Remember:

- nice ⟶ sets priority **when starting a process**.
- renice ⟶ changes priority **of a running process**.