9056299166 * ST - 1 * Harshpreet Singh
5 AM TO 10 PM 2210990393
HELPLINE :) god idk how to make winky face

Linux

History Linux is an open-source operating system kernel developed by Linus Torvalds in 1991. Linus released the kernel as free software under GNU (GNU's Not Unix) GPL (General Public License).

kernel's development was open source & collaborative.

Linux Foundation is a non-profit organisation that supports the growth of Linux and open source software. It provides resources, education and also hosts various events & projects.

Hardware Requirements can run on a wide range of devices, from small devices to powerful servers. Due to it's modest requirements, it is suitable for older & less powerful hardware as well. Rest specific requirements depend upon chosen distributions.

Linux Components

1. **Linux Kernel** → Core of the OS, manages hardware resources, provides essential services like process and memory management. Bridge b/w Hardware & Software.

2. **System Libraries** → Pre-compiled functions for program use, standardizes interaction with the kernel and hardware. Like GNU C library (glibc)

3. System Utilities → Essential programs for system tasks like file handling, process management, and network configuration.

4. File System → Organises and stores files, hierarchical structure under the root directory ("/"). Eg ext4, XFS, Btrfs.

5. Shell and CLI (Command-Line Interface)

Allows users to interact with the Linux system: like Bash, Zsh and others. Shell interprets user commands and provides a command-line interface for running programs and managing the system.

6. Graphical User Interface (GUI)

Provides visual environment with windows, icons, menus and mouse interaction. Eg, GNOME, KDE...

7. Device Drivers

Enable OS / kernel to communicate with hardware devices.

8. Networking Stack.

Handles network protocols, sockets, and configuration for internet and device communication.

9. Process & Memory management.

Pretty self explanatory.

10. Init System & Services

Manages system service initialization and control for essential processes at boot.

11. User Space Utilities

wide range of CLI utilities & system tools to perform process management, file manipulation and system administration like ls, cp, mv, ps, top ...

12. Package Management.

APT, yum, DNF and more are used to install software, updates and removal.

More include Security & Authorization, Error Handling & Printing services.

Linux Distributions or Distros

They are complete packages of Linux OS. They include the Linux kernel, System utilities, libraries, and often a desktop environment or window manager. Each distro is tailored to specific use cases.

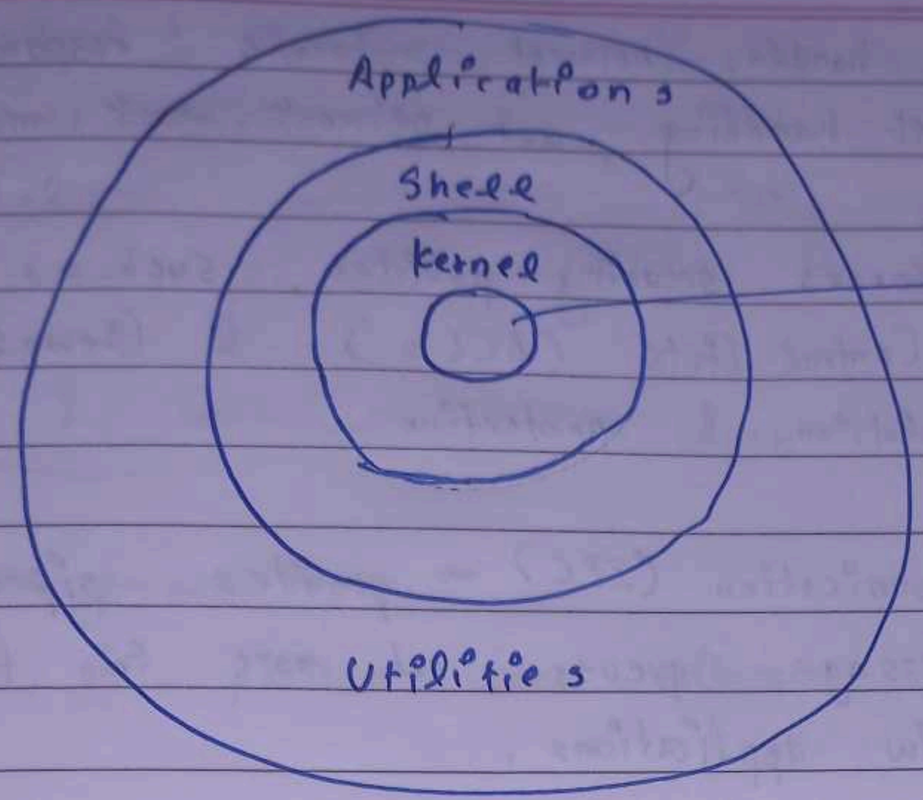
Like Ubuntu → user friendly Debian → stability & large package repo
Centos → enterprise use Fedora → Latest software
Arch Linux → Minimal for experienced users
RHEL → Enterprise grade with paid support.

Features of Linux

- Open Source
- Stability
- Security
- Multitasking
- Flexibility
- Excellent Networking

Difference b/w Linux & Unix

	Linux	Unix
	freely available to everyone (open source)	Only be used by copyrighters.
	Eg → Ubuntu, Mint, Redhat, Fedora, etc.	Eg → IBM AIX, HP-UX and Solaris.
	Anyone can use it.	Mainly for servers, workstations and mainframe.
	Free	Varying costs
GUI	Gnome & KDE	Mostly Gnome
Interface	BASH (Bourne Again Shell)	Bourne Shell
	Linux is just a kernel.	Unix is a complete package.
	Highly Secure.	Also Highly Secure.



Holy I am bad at circles.

Linux Architecture

* Kernel *

Monolithic kernel → means that kernel contains most of the OS's core functionalities within a single, unified structure.

Process Management → handles process creation, scheduling and termination maintains a process table.

Memory Management → manages physical & virtual memory, ensuring memory protection, paging and swapping.
(MMU) unit

Device Drivers → they are integrated into the kernel or provided as loadable modules. Enable communication b/w hardware & software.

File System Management → kernel provides access to the file system, which includes file I/O operations, permissions & metadata.

Networking → handles network protocols, responsible for routing, packet handling, and network stack management.

Security → enforces security policies, such as user & group permissions, Access Control Lists (ACLs) & firewall rules, Ensuring data isolation & protection.

Interprocess Communication (IPC) → provides pipes, sockets and message queues and more to facilitate collaboration b/w applications.

Architectural Differences b/w Windows & Linux.

	Windows	Linux
Kernel type	Hybrid (monolithic & microkernel)	Monolithic.
Licensing	Proprietary (Requires purchase of licenses for most versions)	Open-Source
GUI	Integrated GUI	Seperates GUI from core OS, offering various desktop environments eg. GNOME, KDE, xfce.
Software installation.	Relies on executable installers eg MSI, exe.	Uses package managers (APT, YUM, DNF) for installation.
File System	Predominantly uses NTFS.	Supports multiple file systems eg, ext4, Btrfs, XFS
User Permissions & Security models.	Uses ACLs (Access Control Lists) and user-based permission model.	Uses group permissions model

CLI

Offers Power Shell & Command prompt which are not as extensive & Linux Terminal.

Provides a robust CLI environment with a vast array of tools and scripting capabilities.

Configuration & Customizations of Linux { You can Skip reading this ! }

- Package Management
  - use apt, yum, pacman for software installation, updates & removal.
  - customize software sources & repositories to control where packages are fached from.

- Shell Customization
  - Modify .bashrc or .zshrc for command line environment
  - personalize prompts, set variables & create aliases.

- Kernel Configuration
  - Adjust parameters in /etc/sysctl.conf or /proc
  - recompile kernel or manage modules for performance or features.

- User and Group Management
  - use useradd, usermod, groupadd for accounts / groups.
  - set permissions, create home directories, manage memberships.

- Network Configuration
  - configure in /etc/network/ or /etc/sysconfig/network-scripts/
  - adjust interfaces, set IP addresses, create routing table.

- File System Customization
  - mount and manage file systems in /etc/fstab
  - customize partitions, format disks & set quotas.

BIOS → Basic Input / Output System
UEFI → Unified Extensible Firmware Interface
GRUB → GRand Unified Bootloader.

- Security Hardening
  - configure firewalls y 'iptables' 'ufw' and intrusion detection 'fail2ban'
  - regularly update system for patches.

Linux Structure & Installation.

Filesystem Hierarchy

- Linux directories include /bin, /etc, /home, /var
- /bin holds essential binaries, /etc contains configuration files, /home hosts user directories, and /var stores variables data.

Boot Process

- BIOS / UEFI loads bootloader (eg. GRUB)
- Bootloader loads Linux kernel, initializing the system and launching services.

Installation

- Boot from installation media (USB, CD / DVD)
- Select partitions, file systems, software, and create user accounts.

Package Management

- Use package managers (apt, yum, pacman) to manage software.

Configuration

- Set up user accounts, network, and system-specific setting post installation.

Updates

- Regularly use package manager to apply security patches and software updates.

Differences b/w Linux & Windows (Last time it was Architectural)

Linux	Windows
Open Source	~~Not~~ Not
Free	Costly
Case sensitive	Case-insensitive
Monolithic kernel	Hybrid kernel
More efficient	less efficient
/ → is used for seperating directories.	\ is used.
More secure.	Less secure.
3 types of user accounts • Regular • Root • Service account	4 types of user account • Administrator • Standard • Child • Guest
Root user is the super user & has all admin priviledge.	Administrator has all the admin priviledges.

Linux Boot Process

Power ON → BIOS / UEFI [POST] → Boot Device → Boot Loader [GRUB] → KERNEL → Run Levels / Systemd

BIOS and UEFI

The boot process begins with the BIOS (Basic Input/Output System) or UEFI (Unified Extensible Firmware Interface). They are responsible for initializing the hardware & loading the boot loader.

BIOS, usually, contains all the codes to gain access to the main machine devices. But after Booting is over Device drivers take over.

BIOS uses Master Boot Record (MBR) to store the metadata on and about the drive while UEFI uses GUID Partition Table (GPT)

Next, BIOS or UEFI runs the power on self test (POST). POST ensures the proper operation of the system.

After POST is passed by system we go to Boot Loader.

LILO SYSLINUX GRUB2

GRand Unified Bootloader.

Boot Loader (Mainly, GRUB)

GRUB is responsible for presenting the user with a boot menu, which allows them to select which operating system to boot into using which device.

BIOS has boot loader located in the first sector of the boot device (MBR). Takes up 512 bytes. UEFI stores all booting in an .efi file. on the EFI system partition.

** Notably, boot loader is a small program that loads the operating system. The main job of boot loader is to perform the 3 actions with kernel: locate the disk, insert into memory, and execute with the supplied options. **

Kernel * Core of Linux *

The Kernel is responsible for managing the hardware, running processes, and providing system services. After going through BIOS or UEFI, POST, and using a bootloader to initiate the kernel, the OS now controls the access to computer resources.

Now the followed procedure is:

- → Decompresses the Linux kernel in place
- → Performs Hardware Checks
- → Gain access to vital peripheral hardware
- → Run init process

Init → First process that is started by the kernel. The process is responsible for initializing the system and starting up the other system processes.

Date
Page

System d

Is a system and service manager for Linux Systems. It is responsible for managing the startup and shutdown of system services.

Performs → probes all remaining hardware
→ mount filesystems
→ initiate and terminate services
→ manage essential system processes like user logi
→ run a desktop environment

Run Levels

Is a system configuration that specifies which services should be running at a particular time. The runlevel is typically set during the boot process.

poweroff.target → run level 0 → turn off (shut down) the computer
rescue.target → run level 1 → initiate a rescue shell process
multi-user.target → run level 3 → configure the system as a non-graphical multi-user environment.
graphical.target → run level 5 → establish a graphic multiuser interface with network services.
reboot.target → run level 6 → restart the level.

Next is Login & Desktop Environment.
↓ obviously Login
↓ Provides GUI for user to interact.

Type
Function
level o f
Hardware interacti
Eg.

	Shell	Kernel
Type	User Program	System Program
Function	Provides user with interface to the OS.	Manages hardware and provides services to other programs
Level of Hardware Interaction	Low	High
Eg.	Bash, Zsh, Csh	Linux kernel, XNU, FreeBSD.

Partitioning → Process of dividing a storage device into one or more logical areas, often known as partitions. Each partition can have a different file system.

Advantages → improved performance, security, backup.

Dual booting → Process of installing 2 or more operating systems on the same computer.

Virtual Memory → is a memory management capability of an OS that uses hardware and software to allow a computer to compensate for physical memory shortages by temporarily transferring data from random access memory (RAM) to disk storage.

This enables a computer to run programs larger than its physical memory capacity by temporarily transferring data to disk storage. This is crucial for running multiple applications simultaneously.

Swap Space → portion of a hard disk or SSD used as virtual memory by the OS when physical RAM is fully utilized.

swapon -s → check swap space

use " dd " to create swap space & " mkswap " to setup swap file.

" swapon " → enable swap
" swapoff " → Disable swap.

Adding Swap Space.

1. Creating a Swap File.

- Use the 'dd' command to create an empty file
  eg, sudo dd if=/dev/zero of=/swapfile bs=1G count=4
  for a 4 GB swap file.

- Set appropriate permissions with chmod 600 /swapfile

- Use mkswap to set up the file as swap space
  eg → sudo mkswap /swapfile

2. Activating the Swap File.

- Enable the swap file w/t sudo swapon /swapfile

- Verify the swap space with swapon -s

3. Making Swap Permanent (Adding to '/etc/fstab')

- Add " /swapfile none swap sw 0 0 " to '/etc/fstab'

- This ensures the swap file is activated at boot.

Disk Partitioning

It involves dividing a physical disk into multiple ~~disk~~ local storage units called partitions to organize data and optimize disk storage.

Use 'fdisk -l' for List partitions

Use 'fdisk' command to interactively create, delete and manage partitions.

Use 'df -h' command to display information about disk space usage.

P.T.O.

File System Structure

It is a hierarchical in nature & follows a tree like structure.

The root directory ("/") is at the top of the hierarchy, all other directories are organized beneath it.

1. / (Root directory):

Top-level directory and the starting point for the system hierarchy. Contains essential system files and directories.

2. /bin (Binary):

Contains essential binary executables (commands) required for system recovery and maintenance.

eg → cat, ls, cp → essential command binaries.

Commands used by all the users of the system are located here eg ps, ls, ping, grep, cp.

3. /sbin (System / Super User Binaries):

Contains commands which are required for changing system properties. eg → adduser, reboot, shutdown.

4. /boot:

Contains files necessary for the boot process, such as kernel, initrd (initial RAM disk) and bootloader configuration.

5. /dev (Device):

Contains device files representing physical & virtual devices on the system. eg → /dev/null, /dev/usbmon0

6. /etc (Etcetera):

Contains system-wide configuration files and shell scripts used to boot and initialize system settings. This also contains startup and shutdown shell scripts used to start / stop individual programs.
eg → /etc/resolv.conf, /etc/logrotate.conf

7. /home:

Home directories for individual users are typically located here. Each user has a subdirectory containing their personal files and settings. eg → /home/harshpreet

8. /lib (library):

Contains essential shared libraries required by system binaries and kernel modules. The names are either ld* or lib*.so.*
Eg → ld-2.11.1.so, libncurses.so.5.7.
lib → architecture independent files
lib32 → for 32 bit architecture files
lib64 → for 64 bit architecture files
libx32 → for 64 bit architecture libraries but the pointer size is 32 bit.

* Linux is case insensitive * always remember while using commands except its options like -r & -f are different.

9. /media :

When you connect a removable media such as USB disk, SD card or DVD, a directory is automatically created under the /media directory for them.

10. /mnt (mount) :

A directory used as a temporary mount point for file systems.

11. /opt (Optional) :

Typically used for the installation of 3rd party software and operational packages.

12. /proc (Process) :

A virtual file system providing information about processes and system resources.

13. /root :

The home directory for the root user.

14. /run :

A tmpfs (temporary file system) that holds temporary files created by program during the boot process & runtime.

15. /srv (Service) :

Contains data directories for services provided by the system

16. /sys (System):

A virtual file system providing information about kernel parameters, devices, and interfaces.

17. /tmp (Temporary):

A directory for temporary files created by system and user processes. It is usually cleared on system reboot.

18. /usr (Unix System Resources):

Contains user utilities and applications, including subdirectories for binaries, libraries, and documentation.

19. /var (Variable):

Contains variable data such as log files, spool directories, and temporary files that may change during the systems runtime.

20. /snap:

It the directory in which by default, where all the files and folders from installed snap packages appear on system.

To open terminal → ctrl + alt + t
To close terminal → ctrl + d

Types of Files in Linux

** In Linux everything is a file **

- Normal or Ordinary Files → These files contain data. It can either be text files (like abc.txt) OR binary files (like images, videos, etc.)

- Directory Files → Files that ~~represt~~ represent directories. Directories are to Linux is what Folders are to Windows.

- Device Files → In Linux, every device is represented as a file. By using this file we can communicate with that device.

Navigation Commands

Cd { Change Directory }

The 'cd' command in Linux is used to change the current working directory. It allows you to navigate the file system and move between directories.

Basic Syntax → cd [directory or path name]

. → Represents Current Directory
.. → Represents Parent directory

Change to a Specific Directory → cd /path /to /directory
Change to Home Directory → cd ~~or cd ..~~ or cd ~
Change to Parent Directory → cd .. → navigate* one level up.*

cd . → has no effect. for directories with Spaces → cd "my music" or cd my\ music

Change to Root Directory → cd /

Change to Previous Directory → cd -

Navigate Up multiple levels → cd ../..

→ up 2 levels

*** Use of Absolute and Relative Paths :-

Absolute Path : Starts from Root Directory (eg. '/home/user/music')

Relative Path : Relative to the Current Directory (eg.' ../../folder ')

Tab Completion :-

You can use Tab completion with the 'cd' command to quickly navigate through directories. Type the first few characters of a directory or file name and press Tab.

Using Environment Variables :-

cd $HOME → to change the current working directory to user's home directory.

Tips :-

- cd is a shell built-in command not a seperate executable one.
- The CDPATH environment variable can be set to a colon-seperated list of directories. When you use 'cd' without a full path, it looks for the specified directory in each of the directories listed in CDPATH.
- Make sure you have proper permissions to access the target directory.

pwd { print working directory }

Used to display the current working directory, showing the full path from the root directory.

Basic Syntax → pwd [OPTION]

Options -L or --logical :

This is the default behaviour. Display's the logical current working directory, which may contain symbolic links. This is useful if you want to see the path as it was entered, even if has symbolic links.

pwd -L

-P or --physical :

Display's the real current working directory, which does not contain symbolic links. If you wanna see resolved path without symbolic link references.

Eg. pwd → Out : /home/user/music

pwd -P → Out : /mnt/storage/music

We can get documentation of any command by using man. man ls → to know all about ls

ls {list}

is used to list contents of directories. It provides a wealth of information about files and directories within the specified location.

Basic Syntax → ls [OPTION] ... [FILE] ... .

Options

-a or --all : Shows all files including hidden ones {ls -a}
starting with a '.' like ".git"

-l or --long {ls -l} : Displays detailed information about files & directories, including permissions, owner, group, size, & modification date and name.

-h or --human-readable {ls -h} : Provides human readable sizes (KB, MB, GB)..

-R or --recursive {ls -R} : Lists all files & directories recursively including subdirectories.

{ls -S}
-S : Sorts files by size, largest first.

-t {ls -t} : Sorts files by modification time, newest first.

-u {ls -u} : Sorts files by access time.

-G {} or --no-group {ls -G} : Inhibits the display of group information

-d or --directory {ls -d}: Lists directories themselves, not their contents.

-i or --inode {ls -i}: Displays the index number (inode) of each file.

-r {ls -r}: Displays in reverse order.

ls | more or ls -1 : To display content line by line
{To come out use "q"}

More about * ls -l * command. V-imp.
long list of files.

Eg Output — Actual output from my computer.

hard links to a file or directory

total 4 — total no. of disk blocks

-rw-rw-r--	1	harsh	harsh	249	Apr	18	2023	File1.txt
-rw-rw-r--	1	harsh	harsh	0	Nov	6	15:05	File2.txt
-rw-rw-r--	1	harsh	harsh	0	Nov	6	15:06	File3.txt
drwx------	3	harsh	harsh	4096	Sep	15	19:58	'Old'
File permissions		owner	group	size in bytes	Date of last modification			Name

Q.

d represents what type of file is it.

d = Directory
- = Normal
l = Link
c = Character Special
b = Block Special
s = Socket File

system files

* I will discuss File Permissions in detail in just a bit *.

Using multiple options together.

ls -l -t -r or ls -l -r -t or ls -t -r -l

or ls -ltr or ls -trl .

You can use multiple options together & do whatever you need to .

For eg. What Command will make a long Listing of all the files in our system including hidden files, sorted by modification date (oldest first) ?

→ ls -lat or ls -l -a -t

newest first

→ ls -latr or ls -l -a -t -r

Q. How can you list all files in current directory, including hidden files, in a long format, sorted by size (largest first)

→ ls -l -a -S

mkdir usually creates directories with default permissions (usually 755)

Creating Files and Directories

mkdir { Make directory }

Used to create directories (folders). Allows users to make new directories at specified locations in the file system.

Basic Syntax : mkdir [OPTION] .... DIRECTORY ....

Options

-m, --mode = MODE : Set the file permission mode of the created directory
Eg : mkdir -m 755 dir 1

-p, --parents : Create parent directories if they do not exist
Eg : mkdir -p /path/to/parent/ dir 1

-v, --verbose : Print a message for each created directory
Eg : mkdir -v dir 1 dir 2

Create a Single Directory → mkdir dir 1

Create Multiple Directories → mkdir dir 1 dir 2 dir 3

To make dir 3 but dir 1 & dir 2 exist → mkdir dir 1/dir 2/dir 3

Rest are just options examples.

rmdir { Remove directories }

Command used to **remove empty directories**. Most useless command in linux. Unlike 'rm' which is used for deleting files and directories (including non-empty directories) 'rmdir' is specifically designed for removing empty directories.

Basic Syntax: rmdir [OPTION] ... DIRECTORY ...

Remove a Single Empty Directory → rmdir dir1

Remove Multipe Empty Directories → rmdir dir1 dir2 dir3

To Remove Dir 3 but Dir1 & Dir2 should be there → rmdir dir1/dir2/dir3

Options

-p, --parents : Removes a directory & all its ancestors.
eg. rmdir -p a/b/c . removes c then b the a if they are empty.

-v, --verbose : Outputs a diagnostic message for every directory processed.

eg. rmdir -v dir1 dir2

* will discuss Touch & Cp in later pages *

Disk Partitions

A disk partition is a contiguous and independent space on a storage device, such as a harddrive or SSD. Each partition behaves as a seperate unit with its own file system and directory structure.

Partitioning allows you to divide a storage device into multiple segments, each of which can be merged independently. This is useful for organization, data isolation, and implementing different file systems or operating systems on same physical disk.

Common Partition Types :-

Primary Partition : Main partition for installing an OS.

Extended Partition : Special type of Primary Partition that can be subdivided into logical partitions.

Logical Partition : Part of Extended Partition & used for data storage.

Partitioning Tools

fdisk → A ~~CLI~~ comman line utility for partitioning drives.

gparted → A graphical partitioning editor.

parted → similar to fdisk but with more features.

In Linux partitions are named as /dev/sdXn where X represents drive letter like 'a' 'b' 'c' .. & n represents partition number like '1' '2' '3' ...

File Systems.

A method used by an OS to organise and store files on a storage device. It defines how data is stored, retrieved, and organized within files and directories.

Common File Systems :

ext4 : Default file system for many Linux Distros.
ext3 : predecessor of ext4
XFS : known for scalability & performance
Btrfs : offers features like snapshots, checksums & RAID like functionality.
FAT32 & NTFS : Commonly used in Windows

Formatting Before using a partition, you need to format it with a specific file system. The 'mkfs' command is used. for eg mkfs.ext4 /dev/sdXn.

* **Mounting** Attaching a file system to a specified directory in file hierarchy. 'mount' command is used for eg, 'mount /dev/sdXn /mnt'

* **Unmounting**. Detaching a file system from a directory
eg - 'unmount /mnt'

Periodically a File system should be checked for errors with 'fsck' command.

RAID (Redundant Array of Independent Disks):
A technology that uses multiple disks to provide fault tolerance and increased performance.

LVM (Logical Volume Manager):
A system of managing logical volumes on hard drives or SSDs.

free

this command in Linux is used to display information about system memory usage, including total, used, and free memory. It also provides information about swap space usage.

Basic Syntax → free [OPTION] ...

free -b → to display memory in bytes
free or free -k → " " in kilobytes (DEFAULT)
free -m → " in megabytes
free -g → " in gigabytes

free -t → adds an additional column showing total memory

free -s 5 or free --seconds 5 → updates every N seconds *(default is 3)*

free -h → in human readable form.

Managing disk Space.

df : { disk free }

is used to display information about disk space usage on file systems. It provides information about total disk space, used space, available space, and the file system type.

Basic Syntax → df [OPTION] ... [FILE] ...

df -h → Displays sizes in human readable format.

df -T or --print-type → Additionally displays file system type.

-t or --type = TYPE : limits the report to specific file systems

eg: df -t ext4

-i or --inodes : Displays inode information.

eg: df -i

-a or --all : Includes pseudo, duplicate, inaccessible file systems.

eg. df -a

--total : Display a grand total at the end.

eg df --total.

Eg. Display Disk Space Usage in Human Readable format, including file system type and a grand total.

→ df -h -T --total

du : { Disk Usage }

is used to estimate file and directory space usage. Useful for finding out which directories or files are consuming the most disk space.

Basic Syntax : du [OPTION] ... [FILE] ...

-h, --human-readable : in human readable format
eg : du -h

-s, --summarize : only the total for each argument.
eg : du -s

-c, --total : grand total for disk usage.
eg : du -c

-a, --all : entry for each file in a directory tree
eg : du -a

-d, --max-depth = N : print the total for a directory or a file only if it is N or fewer levels below the command-line argument.
eg : du --max-depth = 2 or du -d 2

--exclude : to exclude the directory
eg : du -h --exclude = excluded_directory /path/to/dir/

You can also combine multiple commands with "|"

eg : du -h /path/to/dir | sort -rh

fdisk :

is a text-based utility used for the creation and manipulation of disk partition table. It allows users to create, delete, modify, and display partitions on a hard drive.

Basic Syntax : fdisk [options] device

To list partition table → sudo fdisk -l → sudo gives administerative privileges we need to use it to make fdisk work.

~~Create a New partition → fdisk /dev/sdX~~

View partition on a specific disk → sudo fdisk -l /dev/sda

View all fdisk commands → sudo fdisk /dev/sda
within this there are interactive mode commands

p → print partition table

n → Create a new partition

d → delete a partition

q → quit without saving changes

w → write changes & exit.

View size of partition → sudo fdisk -s /dev/sda

To see help listing → sudo fdisk -h

crucial for integrating external storage devices or network shares into linux file system hierarchy.

mount & unmount

*talked about them earlier. Let's talk about the command part.

mount command

Basic Syntax : mount [OPTION]... [DEVICE][DIRECTORY]

Some important options :

l : Lists all the file systems mounted yet.
h : Displays options for command
V : Displays the version information
a : Mount all devices described at /etc/fstab
t : Type of filesystem device uses.
T : Describes an alternative fstab file.
r : Read-only mode mounted.

Examples Displays information about file systems mounted
↳ sudo mount -l -t ext4

Mounts file systems
↳ sudo mount /dev/sda /music / yes

Mounts the file systems read only
↳ sudo mount -o ro /dev/sdb1 /mnt / HARSH

Mount a Device to Directory
↳ sudo mount /dev/sdb1 /mnt / HARSH

Mount all filesystems in '/etc/fstab'
↳ mount -a

Mount with Specific Filesystem Type and Options

↳ mount -t ext4 -o rw, exec /dev/sdb1 /mnt/HARSH

Unmount Command

Basic Syntax : unmount [OPTION] .... [DEVICE | DIRECTORY]

Unmount a Device or Directory → unmount /mnt/HARSH

Unmount All Filesystems in '/etc/fstab' → unmount -a

Lazy Unmount → unmount -l /mnt/HARSH

P.T.O.

Linux File Types

File Type	Command to Create the File	Located in	The file type in ls -l	file Command Output
Regular file	touch	Any directory/ Folder	-	text, ASCII, RAD, Image, etc
Directory file	mkdir	-	d	directory
Block Files	fdisk	/dev	b	Block Special
Character Files	mknod	/dev	c	Character Special
Pipe files	mkfifo	/dev	p	FIFO
Symbol link Files	ln	/dev	l	Symbol like to <linkname>
Socket files	socket() systemcall	/dev	s	Socket

Creation of Regular Files :

- By using <u>touch</u> command (to create empty file)
- By using <u>cat</u> command
- By using editors like gedit, vi, nano, etc.

Stat

to display detailed information about a file or file system like file type, permission, inode information, timestamps, and more.

Basic Syntax : stat [OPTION] ... FILE ...

Display Basic File Information
↳ stat file.txt

Customize Output Format
↳ stat -c "File : %n Size : %s bytes" file.txt

Display Terse Form → machine readable output format. Very Concise.
↳ stat -t file.txt

Display File System Information
↳ stat -f /path/to/file

Follow Symbolic Links
↳ stat -L file_Link

* Touch * used to create an empty file or update the access and modification timestamps of an existing file.

Syntax → touch [OPTION] ... FILE

Create an empty file → touch newfile.txt or touch newfile

To create multiple files → touch file1 file2 file3

Change access time only → touch -a file.txt

Date Yes
Page

change modification time only → touch -m file.txt

specify a custom date and time → touch -d "2024-02-04 10:00:00" file.txt

format is YYYY-MM-DD HH:MM:SS

My birthday :)

specify a custom data and time pt.2 → touch -t 202402041000.00 file.txt

format is YYYYMMDDHHMM.SS

reference file1's timestamp as file2 → touch -r file1 file2

Cat is used for concatenating and displaying the content of files.

Syntax → cat [OPTION] ... [FILE]

Display File Contents → cat file.txt

Create a file & add content → cat > newfile
in the next line write the content & press ctrl + d for saving & exiting the file. If the file doesn't exist it's automatically created.

to append instead of overwriting use → cat >> file.

Create file 2 and add file 1 content to file1 → cat file1 > file2

cat file1 >> file2
↓
to append.

Concatenate Multiple Files → cat file1 file2 > combined

Number of non-blank lines → cat -b file1.txt

Number All lines → cat -n file.txt

Suppress Repeated Empty lines → cat -s file.txt

Display $ at the End of Each line → cat -E file.txt

Display TAB characters as ^I → cat -T file.txt

Display Non-Printing Characters → cat -v file.txt.

Regex (Regular Expression) & Wildcard Characters

Regex in Linux are powerful patterns used for searching, matching, and manipulating text.

Basic Regex.

Character matching: '.' → matches any single character.

'[ ]' → character class. Matches any one of the enclosed characters.

'[^]' → negated character class. Matches any character not in enclosed characters.

eg → 'cat' matches to rat.

[aeiou] matches any vowel

[^0-9] matches any non-digit

Anchors

'^' → Anchors the regex at the beginning of the line
'$' → Anchors at the end of the line.

eg → '^start' matches the line starting with start
'end$' " " ending with end.

Quantifiers

'*' → matches 0 or more occurances of preceding element
'+' → " 1 or more "
'?' → " 0 or 1 "
{n} → matches n occurances
{n,} → n or more
{n,m} → b/w n and m.

eg → 'ab*' → a, ab, abb, abbb ..
'a+' → a, aa, aaa ...
'a?' → a or empty string
'\d{3}' → matches exactly 3 digits.

~~Special~~ **Special Characters**

'\' escapes a special character
'|' acts as logical OR.
'\\' matches backslash.
'a|b' matches a or b

Examples

* → 0 or more characters
? → only 1 character
[] → Range of characters
[abc] → either 'a', 'b', 'c'
[!abc] → except a, b, c
[a-z] → a to z any
[A-Z] → A to Z any
[a-z A-Z] → a to z & A to Z any
[0-9] → 0 - 9 any
[a-z A-Z 0-9] → any Alphanumeric
[!a-z A-Z 0-9] → except alphanumeric
[[:lower:]] → lowercase
[[:upper:]] → uppercase
[[:alpha:]] → ~~alphanumeric~~ alphabet
[[:digit:]] → digits
[[:alnum:]] → alphanumeric
[![:digit:]] → except a digit
{} → represent a list of files.

More

To list all files in pwd → ls *
To list all files with some extension → ls *.*
" start with a → ls a*
starts with a and ends with t → ls a*t
all .java files → ls *.java
when first character should be a & only has 2 characters → ls a?
files with only 3 characters → ls ???
atleast 3 characters → ls ???*
starts with a or b or c → ls [abc]*
does not start with a, b, c → ls [!abc]*
files that start with special symbol → ls [![:alnum:]]*

list out files with .java or .py extension → ls {*.java, *.py}

find a powerful utility for searching & locating files and directories within a system.

Syntax → find [path:...] [expression]

{ if no path given searches in present working directory.

Find Files by name → find /path/to/search -name "example.txt"

-iname ignore case → find . -iname olala.txt

To specify a file or directory → find . -type f/d -name f*

To find files with a perticular permission → find . -perm 777

Not with a perticular permission → find . !-perm 777

User with read permission → find . -type f -perm /u=r

empty files → find . -type f -empty

find files with a specific permission → find . -type f -perm -g=w,r -o=x

to find files ~~can't~~ with regex → find . -regex (regex of path)

to find files larger than 1MB → find . -size +1M

modified in the last 7 days → find . -mtime -7

Execute a command on found files → find . -name "*.txt" -exec ls -l {} \;

Point Pa

Eg

Files m
↳

locate

is use
a
by

locat
inde

Syn
l

Speed
Criteria
Real time
Case sensitiv

Print Paths of found files → find . -name "*.txt*" -print

Eg.

Files modified in last 30 days & larger than 1 GB
↳ find . -mtime -30 -size +1G

locate

is used to quickly find the location of files and directories in a pre-built database. This database is usually updated periodically by a system cron job.

locate is much faster than find cause it relies on a pre-built index. But locate has a much limited criteria.

Syntax → locate [OPTION]... PATTERN...

locate myfile.txt & locate -i myfile
↓
case sensitive search.

	locate	find
Speed	Fast	Slow
Criteria	limited	vast
Real time	no	yes
Case sensitivity	yes	no

if a file begins with '-', specify it as '--' in name.

rm {remove here}

is used to remove or delete files and directories for filesystem. It is the most powerful command as file deletes are generally never recoverable.

Syntax → rm [OPTION] ... FILE...

Remove a file → rm filename

Multiple files → rm file1 file2 file3

Remove a directories & its contents → rm -r directoryname (recursively)

Force removal without confirmation → rm -f file

Interactive removal → rm -i file

Verbose removal → rm -v file

To remove empty directory → rm -d dir

To remove your OS → sudo rm -rf --no-preserve-root /

:) don't do this thou.

cp {copy}

used to copy files & directories from one location to another.

Syntax → cp [OPTION]... SOURCE... DESTINATION.

Copy a file to destination → cp file.txt /path/to/dest

Copy multiple files to directory → cp file1 file2 file3 dir3

Copy File1 to File 2 → cp file1 file 2 {if file 2 is not available cp will create file 2. if file 2 is there & containing some data then this data will be overwritten with file content.

Copy all files of directory1 to directory 2 but dir 2 should be present → cp dir1/* dir2

Copy all files of directory recursively → cp -r dir1 dir2

Copy multiply directories → cp -r dir1 dir2 dir3 dir4

Interactive Copy → cp -i file1 file 2 /path/to/desk.

Update copy (Copy only if source is newer) → cp -u file1 file2

Verbose Copy → -v

Preserve Attributes During Copy → cp -r -a source_directory /path

to make a backup → -b

force copy → -f

mv used to "move" or "rename" files & directories.

Syntax → mv [OPTION] ... SOURCE ... DEST.

Rename a file → mv oldname newname

Rename a directory → mv dir 1 dirNew

Move a file to dest → mv file.txt /path

Move multiple files to a directory → mv file1 file 2 /path

Moving all files b/w directories → mv dir 1 /* dir 2

Moving complete directory → mv dir1 dir 2

Overwrite files without confirmation → mv -f f1 f2 /path (→ forceful)

Interactive move → -i backup → -b

Update move → -u version → --version

Verbose move → -v

Creation of Hidden Files & Directories.

any files that start with '.' are hidden. So you can make Homeworks folder hidden boisss.

touch .homework

cat > .homework

mv homework.txt .homework.txt

mv homework .homework

very important to hide your homework folder guys

Cat Continued. not only create it can be used to view files as well. I listed the functionalities a lil bit before... lemme reiterate them.

cat < file or cat file → to view content.

cat file1 file2 → to display multiple files.

n → number all the output lines.
b → give numbering to all lines apart from blank lines.
s → supress repeated empty output lines
v → use ^ and M - notation for non-Printable characters.
T → Display Tab as ^I
E → Display $ at the end of each line.
e → Equivalent to vE.
t → Equivalent to vT.
A → Equivalent to vET.

tac reverse of cat. it is vertical reversal. last line will become the first line

Display File contents in Reverse Order → tac filename

Multiple files → tac file1 file2

b → attach the seperator before instead of after
r → regex
s → seperator for a new Line.

~~rev~~ rev each line is reversed → horizontal reversal.

rev filename rev file 1 file 2

that's it for this command lol.

head to display the beginning / top portion of text files or output streams.

Syntax → head [OPTION]... [FILE]...

Display First 10 Lines of a file → head fileDame

Display First 5 Lines of Multiple files → head -n 5 file1 file2

Display First 20 Bytes of a File → head -c 20 filename

Display First 30 lines → head -n 30 file1 or head -30 file1

All lines except last 20 → head -n -20 file1

Display files without seperator → head -q file1 file2

verbose → head -v file1 Displays the file's name in this case.

tail to display few lines from the bottom of the file. Opposite from head.

Syntax → tail [OPTION] ... [FILE] ...

Display last 10 lines → tail filename

Display last 30 lines → tail -n 30 filename or tail -30 filename

4th line to last line → tail -n +4 filename

200 bytes from bottom → tail -c 200 file1.txt

without seperator → tail -q file1 file2

verbose → tail -v file1

more is a simple text viewer that allows users to view text files one screen / page at a time. Displays text sequentially.

more filename → will display first page. Enter to view the next line. Space Bar to next page. q to quit.

more -d file1.txt → -d → details

-n to specify how many lines are printed at a time
eg → more -n 10 filename, prints 10 lines at a time

-s → squeeze multiple blank lines into one.

less improved version of 'more'. enables ~~forward~~ & backward movement, searching & more efficient scrolling through large files.

d → to next page b → to previous page.

---

passwd used to manage user account passwords.

Syntax → passwd [options] [username]

Change own password → passwd

Change another user's password → sudo passwd username

lock user account → sudo passwd -l username

Unlock user account → sudo passwd -u username

Delete Password → sudo passwd -d username.

---

wc Word Count.

wc filename $\xrightarrow[\text{output}]{\text{Sample}}$ 5 20 100 filename

5 → no. of lines; 20 → no. of words; 100 → no. of bytes.

- -l → lines
- -w → words
- -c → bytes
- -m → characters
- -L → length of longest line.

Echo to display text or variables as output to terminal or shell.

echo [options] [string(s)]

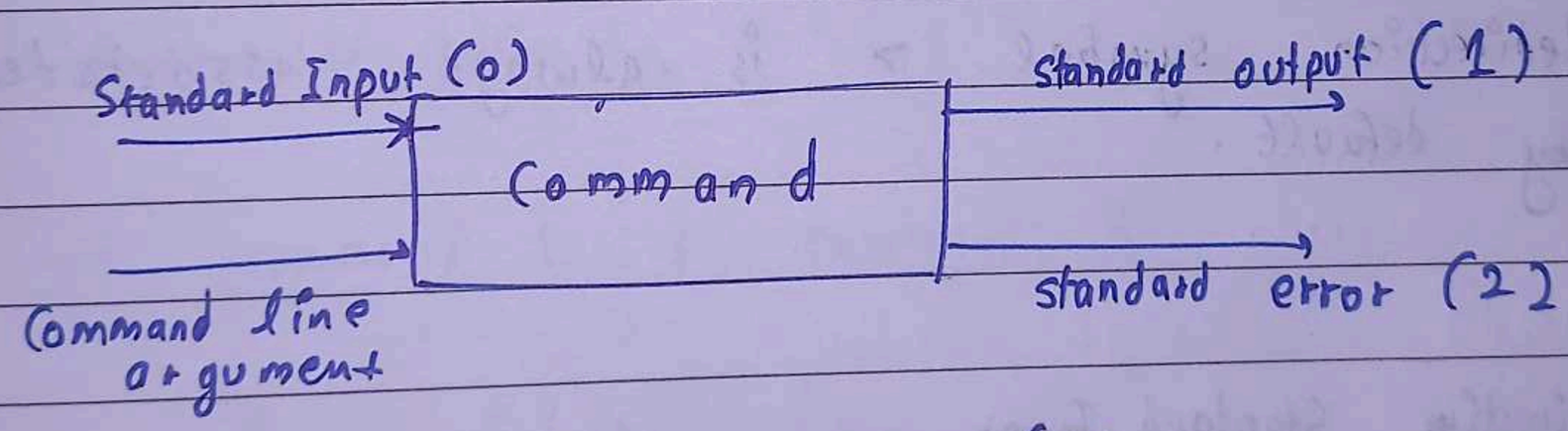
To print → echo " Hello "

echo To print value of variable → var = " 20 "
echo " Value : $var "

To without newline → echo -n " Hola "

Use of escape sequence → echo -e " Hola. \n Hermosa "

Input & Output of Commands



Commands can take input from Standard Input (0) or Command line argument s. Then the Command with either return into Standard output (1) or into standard error (2) stream.

We can pipe & redirect b/w standard streams & data but not to command line argument since they are static.

Redirection

Redirecting Standard Output

by using > and >> symbols

> will perform overwriting
>> will perform appending

Eg. To redirect standard output of cat from terminal to output.txt.

$ cat 1> output.txt

1 represents Standard Output (last page)

Redirection symbol > is always associated with 1 by default.

Redirecting Standard Error

ls non_existent_directory 2> error.log
overwrite

2>>
append

Redirecting stderr & stdout seperately

Command_name > output.log 2> error.log

&> means both stdout & stderr

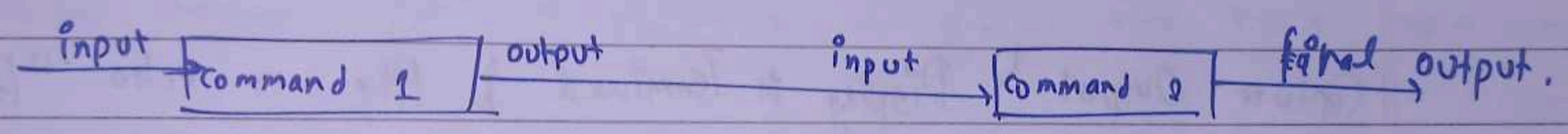
Redirecting stderr to stdout → command_name 2>&1

Redirecting Standard Input "<" symbol.
↳ associated with 0 by default.

Redirecting stdin from a file → command_name < input.txt

Redirecting stdin and stdout together → command_name < input.txt > output.txt.

Piping to use the output of one command as input to another command.



It is implemented using a " | ".

Syntax → command 1 | command 2

Eg → ls -l /etc | wc
ls -l /etc | more
ls -l /etc | wc | wc -l
ls -l /etc | head -5

Piping commands → ls -l | grep ".txt"

Counting lines in a file → cat file.txt | wc -l

Displaying Processes → ps aux | grep "firefox"

tee is used to read the standard input & write it to both standard output (terminal) & one or more files simultaneously.

Syntax → command | tee [options] [file 1] [file 2 ...]

Writing Output to a File → ls -l | tee output.txt
displays both in terminal & writes to file.

Appending Output to a file → cat file.txt | tee -a log.txt
(-a → append)

Capture Output & Display in Terminal & File → echo "Hello, World" | tee greeting.txt

---

chmod change mode → Symbolic
→ Numeric / Octal.

Syntax → chmod [options] [mode] [File_name]

Options
- -R → recursively change permissions within a directory
- -v → verbose prints message with every permission changed
- -c → like -v but for only files whose permission are properly changed
- -f → force
- -h → permission of symbolic links

Symbolic chmod

By default → rw- r-- r--

File permissions are total 9. --- --- ---
user group others

+ → to add permission - → to remove permission

= → to set permission to a specified value.

r → read w → write x → execute

u → owner g → group o → others a → all

Eg. chmod u+x, g+w, o-r demo.txt

Permissions changed from → rw- r-- r-- → rwx rw- ---

Eg. chmod u+x demo.txt

chmod u+w, g+rw, o+r demo.txt

chmod u+x, g-w, o+w demo.txt

chmod u=rw, g=rw, o=r demo.txt

chmod a=- demo.txt res → --- --- ---

a=rwx res → rwx rwx rwx

Octal Chmod.

4 → Read
2 → Write
1 → Execute.

_ _ _ → user, group, others

3 digit number

∴ 1 → Execute
2 → Write
3 → 2 + 1 → Write & Execute
4 → Read
5 → 4 + 1 → Read & Execute
6 → 4 + 2 → Read & Write
7 → ~~Read~~ 4 + 3 → 4 + 2 + 1 → Read, Write & Execute

Eg. chmod 674 file

6 → rw
7 → rwx
4 → r

File Ownership commands (chown, chgrp)

chown used to change the ownership of files & directories.

Syntax → chown [options] new-owner [:new-group] file(s)

-R → changes ownership recursively i.e everything in a directory

Changing Ownership of a file → chown user1 filename

Changes owner of filename to user1.

Changing Ownership of a Directory Recursively → chown -R user1:group1 directory

Changing only group ownership → chown :new-group filename

Changing ownership of symbolic links → chown -h user1 : some_group symlink

chgrp "change group" used to change group owner ship.

~~ch~~ Syntax → chgrp [OPTIONS] Group File (s)

Changing group ownership of a file → chgrp group1 filename

Changing Group ownership of a directory recursively → chgrp -R group1 dir

to print -v → chgrp -v group 1 file1

to match to a reference file → chgrp --reference = ref file File1

to change of the target file instead of link → chgrp --dereference g1 symbolic link

to change symbolic link → chgrp -h grp1 symLink

to report only when changes are made → chgrp -c g1 File1