**LINUX OPERATING SYSTEM**

**Software**

Software is a **set of instructions** or program to perform a specific task.

Examples for software are MS office, Canva and many others.

**Types of Software**

1. **System Software**
2. **Application Software**

**System Software:** It is used to manage the computer hardware.

Examples are Operating System (OS), BIOS (Basic input output system)

**Application Software:** It is a kind of software that performs specific functions given by the user.

Examples are MS Office tools, Canva

**Operating Systems (OS)**

Operating system is a type of system software which acts as an interface between user and hardware.

Examples are Windows, MacOS, Android, iOS

**Architecture of Operating System**

We have 3 layers in architecture of OS

1. Hardware
2. Operating System (OS)
3. Application

|  |
| --- |
| Application |
| OS |
| Hardware |

Chrome, Canva

Windows, Linux

CPU, Storage, Memory

**Functionalities of OS**

* **Memory Management-** Allocates and deallocates memory space as needed by processes.
* **Process Management-** Decides the order in which tasks are executed.
* **File Management System-** Manages file creation, deletion, and organization.
* **Device Management-** Enables communication between the OS and hardware devices.
* **User Interface-** Provides a visual interface with windows and icons.

**Introduction to Linux**

**Linux OS**

Linux is an open source OS which will manage the system hardware and software.

**History of Linux Operating System**

* In the year of 1983 Richard Stalman started GNU’s (GNU Not a Unix) in later 1960 and early 1970 there was a Operating system called Unix which was paid software. Because it was paid software to provide free and open source software to public **Richard Stalman** started GNU’s project.
* In the year of 1985 he founded FSF (Free Software Foundation) to support the GNU project.
* So they started developing GNU herd which was the kernel for the GNU software due to lack of technology and developers they were unable to complete the GNU herd.
* Then later in the year of 1991 the person named **Linus Torvalds** he developed Linux Kernel.
* In the year of 1992 Richard Stalman had a conversation with Linus Torvalds by combining Linux kernel and GNU components so they released Linux OS this is also called as GNU.

**Flavors of Linux (Distribution of Linux)**

Flavors are also known as **Linux Distribution**.

Linux Distribution is a combination of **software packages**, **environment** and **configurations**

**Examples of flavors of Linux are**

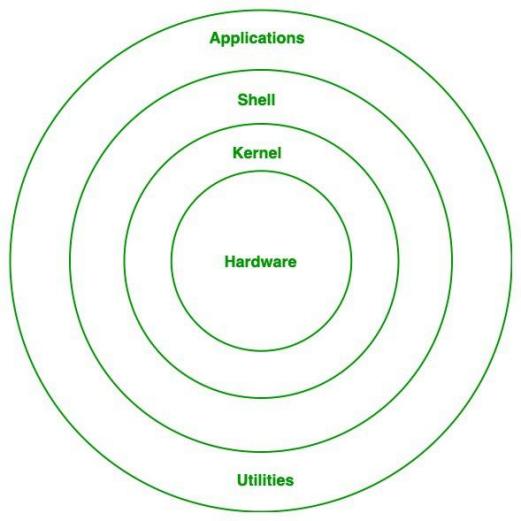
* Ubuntu
* Fedora
* Open SUSE
* Debian
* Cent OS.

Around **500** Linux distributions are there in the market.

**Features of Linux**

* **Open Source**: Linux is free and open-source software, meaning its source code is available for anyone to view, modify, and distribute.
* **Multitasking**: Linux can run multiple applications simultaneously without affecting performance.Multitasking is known for running more than one function at a time.
* **Multi-user**: Multiple users can access and use system resources like memory and hard drive space at the same time without interfering with each other.
* **Portability**: Linux can run on various hardware platforms, from desktops and laptops to servers and mobile devices.
* **Security**: Linux has strong security features, including file permissions and user roles, making it less vulnerable to malwares and viruses and also it contains features such as authentication, authorization and encryption.
* **Stability**: Known for its stability and reliability, Linux systems rarely crash and can run for long periods without needing a reboot.

**Architecture of Linux OS**



1. **Hardware:** Hardware is nothing but physical components of a system.

Examples are CPU, RAM, Storage.

1. **Kernel:** It’s a program that store inside an Operating System.

Kernel is also responsible for hardware and software interaction.

It takes input from shell and pass to the hardware.

1. **Shell:** It’s a command line interface (CLI) responsible for the communication between the user and kernel.
2. **Application:** Where end users can use the applications.

**Difference between Linux and Windows**

|  |  |
| --- | --- |
| **Linux** | **Windows** |
| Linux is free to use and open source | Windows is paid and it’s closed source |
| It’s not user-friendly | It’s user friendly |
| Linux is stable and no need to reboot | Windows requires reboot while updating |
| It’s more secured | Its less secured |
| Linux supports multitasking and multi-user | Windows supports less multitasking and multi-user than Linux |
| Its uses more command line interface (CLI) less graphical user interface (GUI) | It uses more GUI and very little CLI |
| In Linux there is no need of antivirus | In windows there is a need of antivirus |

**Basic Commands of Linux**

* **To create a directory**
* **mkdir dir\_name:** This command is used to create a directory.
  + Ex: mkdir sample – This will create a directory name called sample
* **mkdir dir1 dir2 dir3:** This command will create multiple directories at a time .
  + Ex: mkdir sample1 sample2 sample3
* **mkdir -p dir/subdir1/subdir2:** This will create directory inside another directory generally known as sub directories.
  + Ex: mkdir -p india/Karnataka/Bengaluru
* **Change directory (cd) commands**
* **cd dir\_name:-** This will helps to enter into the directory.

Ex: cd sample – this enters into the directory called sample.

* **cd .. :-**This will come back to the previous directory.
* **cd ../.. :-** used to change the current working directory to the parent of the parent directory.
* **cd / :-** This will go to root directory.
* **ls :-** This command is used to list directory and files.

Ex: ls sample

* **ls -l :-** This command is used to list directory and files(It’s particularly useful for checking file permissions, ownership, sizes, and modification dates). Ex: ls sample
* **pwd :-** This command will print the path of present working directory (pwd).
* **rmdir dir\_name :-** This command will delete the directory which is empty

Ex: rmdir sample

* **rm -r dir\_name –** This command is used to remove a directory which contain files and sub directories.

Ex: rm -r sample.

* **clear –** This command is used to clear the screen.
* **echo –** This command is used to print a line of text.

Ex: echo “statement”

* **history –** This command is used to get history of commands.

**File Commands**

1. **touch filename –** This command is used to create a file

**Ex:** touch file1 – this will create a file name with test1

1. **touch file1 file2 file3-** This command is used to create multiple files at a time.

Here file1 file2 file3 are file names.

1. **touch file{1..n}.ext –** This command is simplified version of above command . Here we’ll give file name series like 1,2,3 in curly braces atlast with its common extension that is .txt, .pdf and so on.

**Ex:** touch rio{1..5}.txt – this command will create 5 files of rio such as rio1, rio2, rio3, rio4, rio5

1. **rm filename –** This command will remove a file from the directory.

Ex: **rm file1.txt** – this command will remove file1 from its respective directory.

**Editors**

**1) vi filename –** This command will open the terminal which is called as vi editor, where these editors are used to create a file and add content in the file.

**Operations need to be performed in vi editors are:**

1. **insert mode [i]-** To add content in the file first we need to be in insert mode.

**Press i** to be in insert mode.

1. **To exit-** To exit from the vi editor firstly click on **esc** then type **:wq**

**w-** save the file

**q-** quit from the terminal.

**2) nano filename-** This command will open the terminal called as nano editor. where these editors are used to create a file and add content in the file..

Here we can add content directly without being in insert mode.

* **To save:** Click on ctrl+s to save the contents of the file.
* **To exit:** Click on ctrl +x to exit from the editor

**3) cat**

* **cat filename-** This command is used to display content which is present in a file in the Linux/Ubuntu terminal itself.
* **cat > filename :** This command is used to rewrite the file content if exists, if it is not then it will create a file and add the content.
* **cat >> filename :** This command is used to add content to the file if exists, if it is not then it will create a file and add the content.

**Paths in Linux**

A file path is the human-readable representation of a file or folder’s location on a computer system.

There are 2 types of paths present in Linux environment.

**1. Absolute path**

**2. Relative path**

1. **Absolute path**

An absolute path specifies the location of a file or directory from the root directory (/). It always starts with a forward slash (/) and provides a complete route to the file or directory, regardless of the current working directory.

Ex : **/home/user/documents/file.txt**

1. **Relative path**

A relative path specifies the location of a file or directory in relation to the current working directory. It does not start with a forward slash and is relative to the directory you are currently in.

Ex:

* If your current directory is **/home/user**, the relative path to **file.txt** inside the documents directory is: **documents/file.txt**
* If your current directory is **/home/user/documents**, and you want to refer to the **file.txt** in the same directory: **file.txt**

**File Management**

Managing files such as moving, copying, renaming, removing.

**Copying (cp)**

In this we’ll copy files from source path to destination path.

The command for copying file from source to destination is

**cp source\_path destination\_path**

**Ex1: cp demo.txt Sample**

Relative path Relative path

Here the demo.txt file is copying to folder called Sample.

**Ex2: cp demo.txt /home/desktop/Test**

Relative path Absolute path

Here the demo.txt file is copying to Test folder where we specified path for the folder Test

**Ex3: cp /home/Desktop/Sample/demo.txt Rio**

Absolute path Relative path

Here the demo.txt file which is specified in absolute path is copying to Rio folder

**Moving (mv)**

This command will move file from source path to destination path.

The command to move file is **mv source\_path destination\_path**

Here source path and destination path can be relative or absolute.

**Ex: mv demo.txt Sample**

**Rename**

In this we’ll rename the file name.

The command for rename is **mv old-filename new-filename**

Here we’ll rename the file which is in same folder.

**Ex: mv demo.txt sample.txt**

This will rename the file demo.txt to sample.txt in the same folder.

**User Management**

In Linux, user management is setting up user permission and accessing the control over the users.

Here we’ll come across 2 users

1. Super user- **Root user**, where they will get all the permissions.
2. Created user- Limited permissions.

Here only **Root** can only create a user.

**Commands for user management**

1. **adduser username –** This command will create an user.
2. **useradd -m -s /bin/bash username-** This command will create an user.

Here -m means make directory.

-s means shell.

**3. su username-**This command will switch to created user.

**4. exit –** This command will exit from the user.

**5. userdel username-** This command will delete a created user.

**6. cat /etc/passwd –** This command will display the user list.

**7. sudo su –** This command will switch to root user.

**Group Management**

Groups in Linux allow administrators to organize and control user access to various resources and file.

**Commands in Group Management**

1. **groupadd groupname-** This command will create a group.
2. **gpasswd groupname-** This command will set password for the created group.
3. **usermod -aG groupname username-** This command will add user **into a g**roup.
4. **gpasswd -d username groupname –** This command will remove users from the group.
5. **groupdel groupname-** This command will delete the group.
6. **cat /etc/group-** This command will list the group and users.

**File Permission**

Every file or directory within Linux has a set of permissions that control who may read, write and execute the contents.

In file we have 3 categories which will give permission

1. Owner/user (u)
2. group (g)
3. others (o)
4. All(a) - (u+g+o)

There are 3 permissions which we will give to file

1. read (r)- to see content in a file.
2. Write (w)- to add content in a file.
3. Execute (x)- to execute a file.

To check file is having permissions or not we’ll be using command:

**ls -l**

For each category of files we’ll be having 3 types of permissions

**rwx |rwx |rwx**

**user/ group others**

**owner**

**Examples**

1. **rw-|r--|r--**

In this we have

* read and write permissions for user/owner
* read permission for group
* read permission for others

1. **r--|rw-|--x**

In this we have

* read permission for user/owner
* read and write permission for group
* execute permission for others

We have values for r,w and x .The values are

* read(r) – 4
* write(w)- 2
* execute(x)- 1

So to give 3 permission at a time we can use number like **777**  which resembles all permission for user/owner, group and others.

The command to give permission to file

* **chmod 777 filename-** This command will give all permission to read, write and execute for all 3 categories.
* **chmod 761 filename-** This command will give all permission to user, read and write permission to group and execute permission for others.
* **chmod u+x filename-** This command will give execute permission for user
* **chmod g+x filename-** This command will give execute permission for group.
* **chmod u-x filename –** This command will remove permission to execute from user.
* **chmod a+x filename-** This command will give all permission to execute for user, group and others.

**To delete permission:**

**Ex: chmod u-x filename**

**File Compression**

In file compression we’ll reduce the size of the file and combine to single folder.

1. tar
2. zip
3. gzip

**1.tar (tape archive)**

In tar we’ll collect multiple files to single file.

To achieve tar we’ll use command

**tar cf archive.tar files**

where

c- create, f-file

Ex: tar -cf archive.tar Demo.java Sample.java readme.txt

**To extract tar file :- tar xf archive.tar**

where x- extract f-file

Ex: tar -xf archive.tar

**2.zip**

**To Create zip files**

**Command:** zip name.zip files

**Ex:-** zip zipfile.zip a.txt b.java v.js

**To extract zip files**

**Command:**unzip name.zip

**Ex :-** unzip zipfile.zip

**3.gzip**

This is used to compress the file.

**To create gz file**

**Command:** gzip filename

**Ex :-** gzip zipfile.zip (this gives a file zipfile.zip.gz)

**To extract gz file**

**Command:** gzip -d filename

**Ex:-** gzip -d zipfile.zip.gz

**File Systems**

Linux supports a variety of file system types, each designed to meet different needs and performance requirements. Here’s a detailed look at some of the most commonly used file system types in Linux:

**1. Ext (Extended File System) Family**

* **Ext2**:
  + **Features**: Introduced in 1993, it lacks journaling, which means it’s faster but less resilient to crashes.
  + **Use Case**: Suitable for flash drives and SD cards where write cycles are a concern.
* **Ext3**:
  + **Features**: Adds journaling to Ext2, improving reliability by keeping a log of changes.
  + **Use Case**: Commonly used in older Linux distributions, suitable for most general-purpose storage needs.
* **Ext4**:
  + **Features**: Introduced in 2008, it includes improvements such as larger file system and file size limits, faster file system checking, and more efficient allocation.
  + **Use Case**: The default file system for many modern Linux distributions, offering a good balance between performance and reliability.

**2. XFS(X file system)**

* **Features**: A high-performance journaling file system created by Silicon Graphics in 1993. It excels in handling large files and supports advanced features like delayed allocation for improved performance and scalability.
* **Use Case**: Ideal for systems that require high performance and handle large files, such as media servers and scientific computing.

**3. Btrfs (B-tree File System)**

* **Features**: Developed by Oracle, it focuses on fault tolerance, repair, and easy management. It includes features like copy-on-write, snapshotting, integrated multi-device spanning, and self-healing.
* **Use Case**: Suitable for modern, large-scale storage systems that require high reliability and ease of management.

**4. JFS (Journaled File System)**

* **Features**: Developed by IBM, it provides a balance of performance, scalability, and reliability. It supports large file systems and is known for its fast recovery times.
* **Use Case**: Suitable for enterprise environments where reliability and performance are critical.

**5. ZFS(Z file system)**

* **Features**: Originally developed by Sun Microsystems, it offers high storage capacities, data integrity verification, and built-in snapshotting and replication. It uses a pooled storage model and is highly scalable.
* **Use Case**: Ideal for environments requiring high reliability, data integrity, and scalability, such as data centers and cloud storage services.

**6. VFAT (Virtual File Allocation Table)**

* **Features**: A Linux implementation of the FAT file system used in Windows. It’s compatible with both Linux and Windows but lacks advanced features like journaling.
* **Use Case**: Commonly used for USB drives and memory cards that need to be accessible from multiple operating systems.

**7. NTFS (New Technology File System)**

* **Features**: The default file system for Windows, known for its robustness and support for large files and volumes. Linux can read and write to NTFS with the help of ntfs-3g driver.
* **Use Case**: Useful for dual-boot systems and external drives shared between Windows and Linux.

**8. F2FS (Flash-Friendly File System)**

* **Features**: Designed by Samsung for NAND flash memory storage devices such as SSDs and SD cards. It aims to extend the lifespan of flash memory and improve performance.
* **Use Case**: Best for use with flash-based storage devices to optimize performance and longevity.

**9. ISO 9660**

* **Features**: Standard file system for CD-ROM and DVD-ROM discs. It is read-only and widely supported across different operating systems.
* **Use Case**: Used for distributing software and media on optical discs.

**10. TMPFS (Temporary File System)**

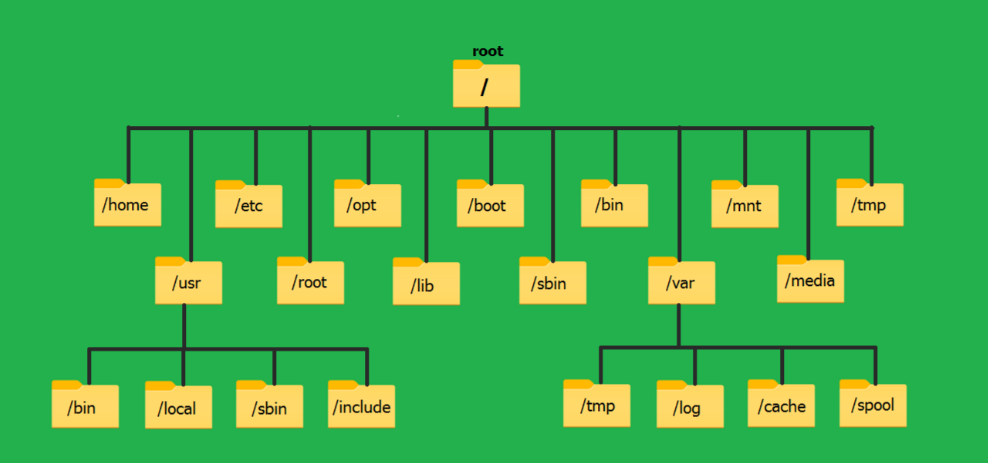
* **Features**: A temporary file system stored in volatile memory (RAM). It is very fast because it leverages system memory for storage. TMPFS can dynamically adjust its size, using available RAM and swap space as needed.
* **Use Case**: Ideal for temporary storage of files that do not need to persist after a reboot, such as temporary files in /tmp or runtime data for applications.

**11. NFS (Network File System)**

* **Features**: Developed by Sun Microsystems, NFS allows a system to share directories and files with others over a network. NFSv4 includes enhanced security and performance features.
* **Use Case**: Commonly used in enterprise environments for network-attached storage, facilitating easy sharing of files between servers and clients.

Commands to see the file systems in your Linux machine.

**File System Hierarchy**

****

**Root Directory (/)**

The top level of the file system hierarchy. All other directories and files stem from this root directory.

**Standard Directories**

* **/bin**
  + Contains essential command binaries (executables) needed for the system to boot and run in single-user mode.
  + Common utilities include ls, cp, mv, rm, and cat.
* **/boot**
  + Contains the boot loader files, kernel images, and other files required for the system to boot.
* **/dev**
  + Contains device files that represent hardware and virtual devices. These files allow software to interact with hardware components.
  + Examples include /dev/sda (hard disk), /dev/tty (terminal devices), and /dev/null.
* **/etc**
  + Contains system-wide configuration files and shell scripts used for system management and configuration.
  + Examples include /etc/passwd (user account information), /etc/fstab (filesystem mount points), and /etc/hosts (network hosts).
* **/home**
  + Contains the home directories of all users. Each user has a subdirectory within /home named after their username.
  + For example, /home/sachin would be the home directory for the user sachin.
* **/lib**
  + Contains essential shared libraries needed by binaries in /bin and /sbin.
* **/media**
  + Mount point for removable media such as USB drives, CDs, and DVDs.
* **/mnt**
  + Temporary mount point for filesystems, typically used for manually mounting filesystems.
  + For example, you might temporarily mount a remote file system to /mnt/nfs.
* **/opt**
  + Contains optional software packages and third-party applications that are not part of the default system installation.
  + Typically used for large software packages, such as /opt/google/chrome.
* **/proc**
  + A virtual filesystem providing information about running processes and the kernel.
* **/root**
  + The home directory for the root (superuser) account.
  + Unlike other user home directories located in /home, the root user’s home directory is directly under the root directory.
* **/run**
  + Contains runtime data, such as information about running system services and logged-in users.
  + Files and directories are typically cleared at boot.
* **/sbin**
  + Contains essential system binaries and commands that are generally used by the system administrator.
  + Examples include ifconfig, iptables, and reboot.
* **/srv**
  + Contains data for services provided by the system, such as web and FTP servers.
  + For example, /srv/www might hold web server data.
* **/sys**
  + A virtual filesystem providing a view into the kernel's device tree, providing information about hardware devices and their drivers.
  + Files and directories represent the system’s hardware structure.
* **/tmp**
  + Temporary files created by system processes and applications.
  + Files in /tmp are usually cleared upon reboot.
* **/usr**
  + Contains user-installed software and utilities that are not required for the system to boot or operate in single-user mode.
  + Subdirectories include:
    - **/usr/bin**: Non-essential command binaries.
    - **/usr/lib**: Libraries for binaries in /usr/bin and /usr/sbin.
    - **/usr/local**: Locally installed software and packages.
    - **/usr/share**: Architecture-independent data, such as documentation and icons.
* **/var**
  + Contains variable data files, such as logs, spool files, and temporary email files.
  + Subdirectories include:
    - **/var/log**: System log files.
    - **/var/spool**: Spool directories for tasks like printing and mail.
    - **/var/tmp**: Temporary files that should be preserved between reboots.

**Networking Commands**

**1)ifconfig**

* Displays or configures network interfaces.
* Example: **ifconfig**

displays all active interfaces and their configuration.

**2)ip**

* A powerful and versatile command for network management.
* Example: **ip address**

shows all network interfaces and their IP addresses

**3)ping**

* Tests connectivity to another host by sending ICMP echo requests.
* Example: **ping google.com**

checks connectivity to Google.

4) **netstat**

* Displays network connections, routing tables, interface statistics, masquerade connections, and multicast memberships.
* Example: **netstat -tuln**

shows all listening ports.

**5)ss**

* Displays socket statistics. It's a modern replacement for netstat.
* Example: **ss -tuln**

shows all listening ports.

**6)traceroute**

* Traces the route packets take to a network host.
* Example: **traceroute google.com**

shows the path packets take to Google.

**7) nslookup**

* Queries DNS servers for information about hostnames and IP addresses.
* Example: **nslookup google.com**

fetches DNS information for Google.

**8) dig**

* A powerful DNS query tool.
* Example: **dig google.com**

fetches DNS information for Google.

**9) route**

* Displays or modifies the IP routing table.
* Example: **route -n**

shows the routing table

**Extra commands**

**To search the files**

**1) Find**

For searching for files and directories in a directory hierarchy.

Ex: **find -name "example.txt"**

Find all files named "example.txt" in the current directory and its subdirectories.

**2)** **Locate**

For quickly finding files and directories in a pre-built database.

Ex: **locate filename**

Find all files containing "example" in their name.

**grep (Global regular expression point)**

* **grep “test” example.txt**

shows the lines with the word test in the file example.txt

* **grep -i “test” example.txt**

case insensitive search

* **grep -n “test” example.txt**

shoes the line number

* **grep -n “test” example.txt**

excluding the lines with test word

**wc (word count)**

* **wc -lwc filename**

calculate lines, words and characters count of the file

**df(disk free)**

* **df -h**

A temporary file system in a human-readable format.

* **df -T**

To display the type of each file system.

**lscpu**

* **lscpu**

Gives detailed information about the CPU.

**hostname**

* **hostname**

shows the current hostname

**free**

Display information about the system's memory usage.

* **free -h :-**Show memory usage in human-readable format.
* **free -t :-** Display a line showing the total memory used and available.

**| (pipe)**

Pipes in Unix/Linux allow you to pass the output of one command directly into another command as input.

* **ls -l | grep "\*.txt"**
  + ls -l: Lists files and directories in the current directory in long format.
  + |: The pipe symbol.
  + grep “\*.txt": Filters the list to show only. The files ending with .txt extension.
* **cat filename.txt | wc -l**
* cat filename.txt: Outputs the contents of filename.txt.
* | wc -l: Pipes the output to wc -l, which counts the number of lines.

**To check user id**

* **id username**

Shows the id and group id of the user.

**To convert normal user to super user**

Add them to sudo group