

Linux

Index

Introduction of Linux	2
Linux Architecture	3
Feature of Linux	4
Category & Evaluation of Linux	6
Popular Variant of Linux	7
Directive of Linux	8
Linux Packages and Formats	9
Management & Operation of Linux	10
File System of Linux	13
User Group	17
File & directory Permission	18
Special Permission	20
Disk Management	23
Service & process management	24

Introduction of Linux:

Linux's story began in 1991 when Linus Torvalds, a Finnish student, developed a free Unix-like kernel. This core, built on open-source principles, allowed anyone to contribute and modify the code. The GNU Project, with its vast collection of userland utilities, provided the missing pieces for a complete operating system. This collaboration fostered a vibrant community that continues to shape Linux today. Initially used on servers for its stability and security, Linux gained traction on desktops in the 2000s with user-friendly distributions like Kali, Red Hat, and Ubuntu. These distros offered a familiar experience for Windows users and showcased Linux's versatility. Constant development ensures Linux remains relevant. New features are added, security is enhanced, and specialized distributions cater to specific needs (cloud computing, scientific research). Today, Linux dominates the server market and holds a growing presence on desktops. Its open-source nature, constant innovation, and strong community make it a future-proof operating system. Linux boasts a unique blend of features that make it a compelling choice for desktops, servers, and beyond. Here are some highlights:

- **Open-Source:** The core principle. Free access to the source code allows for constant improvement by a vast community, fostering innovation and customization.
- **Secure:** Open-source scrutiny allows for quick identification and patching of vulnerabilities, leading to a robust and secure system.
- **Stability:** Renowned for its uptime reliability, Linux is a rock-solid foundation for servers and mission-critical applications.
- **Versatility:** No one-size-fits-all approach. Hundreds of distributions cater to specific needs, from user-friendly desktops (Ubuntu) to secure testing powerhouses (Kali Linux).
- **Cost-Effective:** Free to use and modify, Linux eliminates licensing fees associated with proprietary operating systems.
- **Customizable:** Power users can tailor the system to their preferences, making it a playground for those who love tinkering.
- **Lightweight:** Compared to some proprietary systems, Linux can run efficiently on older hardware, extending the lifespan of your machine.
- **Package Management:** Package managers simplify software installation, updates, and removal, making software management a breeze.
- **Active Development:** The Linux community is constantly evolving, ensuring the system stays up-to-date with the latest technologies and security patches.

These features combine to make Linux a powerful, adaptable, and secure operating system, offering a compelling alternative for users of all levels. Today, Linux continues to evolve. Specialized distributions cater to diverse needs, from scientific computing to media centers. The core kernel undergoes constant updates, ensuring compatibility with modern hardware. Linux's role in modern computing is multifaceted. It reigns supreme in the server market, powering the infrastructure behind countless websites and applications. Its versatility extends to embedded systems in devices like routers and smart TVs, even playing a crucial role in cloud computing and supercomputers. Linux's story is one of open collaboration and continuous improvement. From a student's vision to a global phenomenon, it remains a cornerstone of modern computing, offering a secure, adaptable, and cost-effective platform for a vast range of applications.

Linux Architecture:

The architecture of Linux can be understood from several perspectives: kernel architecture, system architecture, and hardware architecture.

□ Hardware Layer:

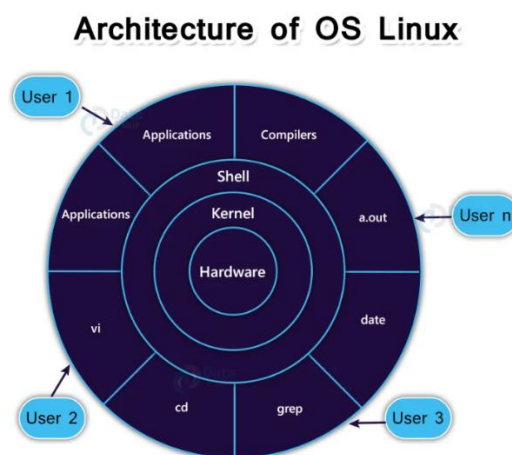
- At the lowest level, Linux interacts directly with hardware components such as CPU, memory, disks, network interfaces, etc.
- Device drivers manage communication between the hardware and the kernel.

□ Linux Kernel:

- The Linux kernel provides core operating system services:
 - **Process Management:** Creating, scheduling, and terminating processes.
 - **Memory Management:** Allocating and deallocating memory for processes.
 - **File System:** Managing file storage and retrieval.
 - **Device Drivers:** Communicating with hardware devices.
 - **Network Stack:** Handling network communications.
 - **System Calls:** Interfaces for applications to request kernel services.
 - **Secure:** Enforcing access controls and permissions.

□ System Libraries:

- Libraries such as **libc** (C library) provide a standard set of functions for programs running on Linux.
- They abstract low-level kernel interfaces into higher-level APIs that applications can use.



□ System Utilities:

- Essential command-line tools and utilities for managing the system:
 - **Shell (like Bash) :** Bash is a command-line interpreter that provides a text-based interface to interact with the operating system. allows users to execute commands, run scripts, manage files, and control processes. Bash is highly customizable and widely used as the default shell on most Linux distributions.
 - **File utilizes (ls, cp, mv):** ls: Lists directory contents, cp: Copies files or directories from one location to another. mv: Moves files or directories from one location to another (can also rename files)
 - **Process management tools (PS, top):** PS: Reports a snapshot of current processes. top: Provides a dynamic real-time view of system processes
 - **Networking tools (ifconfig, ping):** ifconfig: Displays and configures network interfaces. ping: Checks the network connectivity to a remote host by sending ICMP echo request packets.
 - **Package management tools (apt, yum):** Used to manage software packages.
 - **apt-get, apt-cache** are related commands for package management, searching, and querying package information.

□ Application Software:

- User-facing programs and applications that run on top of the kernel and libraries:
 - Web browsers
 - Office suites
 - Media players
 - Development tools

Feature of Linux

The heart of Linux lies in its open-source nature. The source code is freely available for anyone to inspect, modify, and distribute. This fosters a collaborative spirit, with a global community contributing to its development. This constant improvement leads to a more secure and feature-rich system compared to closed-source alternatives. Linux is renowned for its uptime reliability. Servers running Linux are known to operate for extended periods without crashing, making it a trusted choice for mission-critical applications. The open-source approach also contributes to security. With the code readily available for scrutiny, vulnerabilities are identified and patched quickly, enhancing the overall system security. One size doesn't fit all, and Linux recognizes that. Unlike some monolithic operating systems, Linux boasts a vast array of distributions (distros) catering to diverse needs. Beginner-friendly distros like Ubuntu offer a user-friendly interface, while power users can leverage advanced distros like Arch Linux for ultimate customization. Additionally, specialized distros like Kali Linux cater to security professionals with pre-loaded penetration testing tools.

Linux plays a critical role in both cybersecurity and system administration, offering a robust and versatile foundation for professionals. Here's a closer look at its strengths in each domain:

Cyber-security:

- **Secure Focus:** Linux is inherently secure due to its open-source nature. The publicly available code allows security researchers to identify and address vulnerabilities quickly.
- **Penetration Testing:** Distributions like Kali Linux come pre-loaded with a vast arsenal of tools specifically designed for penetration testing, vulnerability assessment, and security auditing. These tools empower security professionals to ethically explore vulnerabilities and identify weaknesses in networks and systems before malicious actors can.
- **Customizability:** Linux allows for deep customization, enabling security professionals to tailor the system to their specific needs. They can install and configure security tools precisely for their environment, enhancing overall security posture.
- **Active Community:** The vast Linux community includes a significant number of security experts who actively contribute to the development of security tools and share knowledge. This fosters a continuous learning environment for cybersecurity professionals.

System Administration:

- **Stability and Uptime:** Linux servers are renowned for their stability and uptime reliability. This ensures critical systems and services run smoothly with minimal downtime, ideal for system administrators who need a reliable platform.
- **Command Line Efficiency:** System administration tasks are often handled efficiently through the command line. Linux offers a powerful command-line interface (CLI) that allows for automation, scripting, and fine-grained control over the system – a significant advantage for experienced system administrators.
- **Package Management:** Package managers in Linux simplify software installation, updates, and removal. This streamlined process saves system administrators time and reduces the risk of errors compared to manual installations.
- **Open-Source Software Ecosystem:** A vast array of open-source software is available for Linux, catering to various system administration needs. From security tools to server management applications, system administrators have a wide range of options at their disposal, often free of licensing costs.

In conclusion, Linux empowers both cybersecurity professionals and system administrators with a secure, versatile, and efficient platform. Its open-source nature, powerful tools, and active community make an invaluable asset in today's ever-evolving landscape.

Overall, Linux empowers cybersecurity professionals with the tools, flexibility, and stability they need to combat evolving cyber threats and protect critical infrastructure.

Category & Evaluation of Linux:

There are two main ways to look at "types" of Linux:

1. By Kernel vs. Distribution:

- **Linux Kernel:** This is the core of the operating system, the software that manages communication between hardware and applications. It's open-source and constantly evolving. There's technically only one Linux kernel, although it has many versions with different features and functionalities.
- **Linux Distributions (Distros):** These are built on top of the Linux kernel and provide a complete operating system experience with a user interface, applications, and configuration tools. There are hundreds of Linux distros, each catering to specific needs and user preferences. Here are some examples:
 - **Desktop-oriented:** Ubuntu, Mint, Fedora (user-friendly, familiar for Windows users)
 - **Server-oriented:** Red Hat Enterprise Linux (RHEL), CentOS (stable, secure, reliable)
 - **Secure-focused:** Kali Linux (pre-loaded with penetration testing tools)
 - **Lightweight:** Lubuntu, Puppy Linux (ideal for older hardware)

2. By Purpose:

Here, we categorize Linux based on its intended use:

- **Desktop Linux:** Designed for everyday use on personal computers, offering a graphical user interface and familiar applications (web browsers, email clients, office suites).
- **Server Linux:** Optimized for running servers, prioritizing stability, security, and uptime for critical applications like web servers, databases, and email servers.
- **Embedded Linux:** Tailored for devices with limited resources like routers, smart TVs, wearables, and industrial control systems. Often lightweight and optimized for specific tasks.
- **Mobile Linux:** While less common, there are Linux distributions designed for smartphones and tablets (e.g., Android is based on the Linux kernel).

The term "variation" for Linux can be interpreted in two ways:

1. **Linux Kernel Versions:** There's technically only one Linux kernel, but it has many versions. These versions represent different stages in the kernel's development, each offering improvements, bug fixes, and sometimes new features. Distributions typically choose a specific kernel version for stability or incorporate newer versions for cutting-edge features.
2. **Linux Distributions (Distros):** This is the more common interpretation of "variation" in Linux. Distributions are built on top of the Linux kernel and provide a complete operating system experience. Here's how distros create variations:

- **Package Selection:** Distros curate a selection of software packages pre-installed on the system. These packages can include desktop environments, productivity tools, system utilities, and more. The choice of packages caters to the distro's specific purpose (desktop, server, secure, etc.).
- **Desktop Environment:** Distros offer different graphical user interfaces (GUIs) called desktop environments. These environments provide a user-friendly way to interact with the system. Popular desktop environments include GNOME, KDE Plasma, Xfce, and LXDE. Each offers a unique look and feel.
- **Configuration:** Distributions can have different default configurations for various aspects of the system, such as secure settings, network settings, and power management. These configurations cater to the distro's target audience and intended use.

Here's a list of some popular Linux distributions categorized by their purpose:

Desktop-oriented:

- **Ubuntu:** User-friendly, beginner-friendly, GNOME desktop by default, vast software library.
- **Mint:** Based on Ubuntu, known for its ease of use, Cinnamon desktop by default.
- **Fedora:** Popular desktop distro, known for being cutting-edge, KDE Plasma desktop by default.
- **Elementary OS:** Sleek and stylish interface, GNOME-based desktop environment.
- **Manjaro:** Arch Linux-based, user-friendly installer, customizable, various desktop environments available.

Server-oriented:

- **Red Hat Enterprise Linux (RHEL):** Stable, secure, and commercially supported, ideal for enterprise environments.
- **CentOS Stream:** Free, community-driven version of RHEL, good for development and testing servers.
- **Debian:** Stable and reliable, popular choice for servers and personal use.
- **OpenSUSE:** User-friendly server distro, good for beginners, multiple desktop environment options.

Secure-focused:

- **Kali Linux:** Ethical hacking and penetration testing platform, pre-loaded with secure tools.
- **Parrot OS:** Secure-focused distro based on Debian, good for privacy and anonymity.
- **Black Arch Linux:** Rolling release distro with a vast repository of secure tools.

Lightweight:

- **Lubuntu:** Lightweight version of Ubuntu, good for older or low-resource machines.
- **Puppy Linux:** Extremely lightweight distro, known for its portability and fast boot times.
- **Bodhi Linux:** Lightweight distro based on Ubuntu, focuses on simplicity and efficiency.

Other Specialized Distros:

- **Android (Mobile):** Based on the Linux kernel, powers most smartphones and tablets.
- **Chrome OS:** Web-based operating system, lightweight and secure, used in Chromebooks.
- **CentOS Stream (Scientific Computing):** Scientific computing variant of CentOS Stream, pre-loaded with scientific software.

Understanding these categories can help you choose the right type of Linux for your needs. If you're new to Linux, starting with a beginner-friendly desktop distro like Ubuntu or Mint is recommended.

Directive of Linux:

□ Navigation and File Management:

- **ls:** List directory contents.
 - Example: `ls -l` (list in long format)
- **cd:** Change directory.
 - Example: `cd /home/user` (change to /home/user directory)
- **pwd:** Print working directory.
- **mkdir:** Make directories.
 - Example: `mkdir my folder` (create a directory named my folder)
- **rm:** Remove files or directories.
 - Example: `rm myfile.txt` (remove myfile.txt)
- **cp:** Copy files or directories.
 - Example: `cp file1.txt file2.txt` (copy file1.txt to file2.txt)
- **mv:** Move/rename files or directories.
 - Example: `mv file1.txt newname.txt` (rename file1.txt to newname.txt)

- **ps**: Report a snapshot of current processes.
 - Example: `ps aux` (list all running processes with detailed information)
- **kill**: Send a signal to terminate a process.
 - Example: `kill PID` (terminate process with ID PID)

□ **Networking:**

- **ping**: Send ICMP ECHO_REQUEST to network hosts.
 - Example: `ping google.com` (ping Google to check network connectivity)
- **ifconfig** or **ip**: Configure network interfaces.
 - Example: `ifconfig eth0 up` (bring up the eth0 network interface)
- **wget** or **curl**: Download files from the web.
 - Example: `wget http://example.com/file.zip` (download file.zip from example.com)

□ **User and Permissions:**

- **sudo**: Execute a command as the superuser (root).
 - Example: `sudo apt update` (update packages using apt with superuser privileges)
- **chmod**: Change file permissions.
 - Example: `chmod 755 file.sh` (give file.sh read, write, and execute permissions to owner, and read and execute permissions to group and others)
- **chown**: Change file owner and group.
 - Example: `chown user:group file.txt` (change owner to user and group to group for file.txt)

□ **Package Management:**

- **apt** (Advanced Package Tool): Package management tool for Debian-based systems.
 - Example: `sudo apt update` (update package lists)
- **yum** or **dnf** (Dandified YUM): Package management tool for Red Hat-based systems.
 - Example: `sudo yum install package_name` (install a package using yum)

Management & Operation of Linux:

Managing and operating Linux involves various tasks and responsibilities to ensure the system runs smoothly, securely, and efficiently. Here's a comprehensive overview of key aspects:

1. System Administration Tasks:

- **User and Group Management:**
 - Creating, modifying, and deleting user accounts (`useradd`, `usermod`, `userdel`).
 - Managing user permissions and access control (`chmod`, `chown`, `sudo`).

- **File System Management:**
 - Creating, mounting, and managing file systems (mount, umount, mkfs).
 - Monitoring disk usage (df, du) and maintaining storage (fstrim).
- **Package Management:**
 - Installing, updating, and removing software packages (apt, yum, dnf).
 - Managing software repositories and dependencies (apt-get, rpm).
- **Network Configuration:**
 - Configuring network interfaces (ifconfig, ip).
 - Managing network services (systemctl, service).

2. Monitoring and Performance Tuning:

- **System Monitoring:**
 - Monitoring system performance (top, htop, vmstat, sar).
 - Checking system logs (dmesg, journalctl, /var/log).
- **Performance Tuning:**
 - Optimizing CPU and memory usage (nice, renice, ulim).
 - Tuning kernel parameters (sysctl, /proc/sys).

3. Secure Management:

- **User Authentication and Authorization:**
 - Managing passwords and authentication mechanisms (passwd, pam).
 - Configuring sudo access and permissions (visudo).
- **Firewall and Network Secure:**
 - Configuring firewall rules (iptables, firewalld).
 - Securing network services (SSH, TLS/SSL).
- **System Hardening:**
 - Applying secure patches (apt update, yum update).
 - Disabling unnecessary services and restricting access (systemd).

4. Backup and Recovery

- **Data Backup:**
 - Creating and scheduling backups (rsync, tar, cron).
 - Using backup solutions and cloud storage (rsnapshot, Bacula, AWS S3).
- **System Recovery:**
 - Recovering from system failures (grub, systemd rescue mode).
 - Restoring data from backups (tar, rsync).

5. Automation and Scripting

- **Shell Scripting:**
 - Writing and executing shell scripts (bash, sh).

- Automating respective tasks (cron, systemd timers).
- **Configuration Management:**
 - Using configuration management tools (like Ansible, Chef, Puppet) for consistent system configuration and deployment.

6. Virtualization and Containerization

- **Virtualization:**
 - Running virtual machines (KVM, VirtualBox, VMware).
 - Managing virtual networks and resources.
- **Containerization:**
 - Using containers (Docker, Podman) for lightweight application deployment.
 - Orchestrating containers with tools like Kubernetes.

7. Collaboration and Documentation:

- **Documentation:**
 - Documenting configurations, procedures, and troubleshooting steps.
 - Using wikis, version control systems (G), or documentation tools (Markdown, Sphinx).
- **Collaboration:**
 - Sharing knowledge and best practices within teams (Slack, Teams, Zoom).
 - Using issue tracking and project management tools (Jira, Redmine).

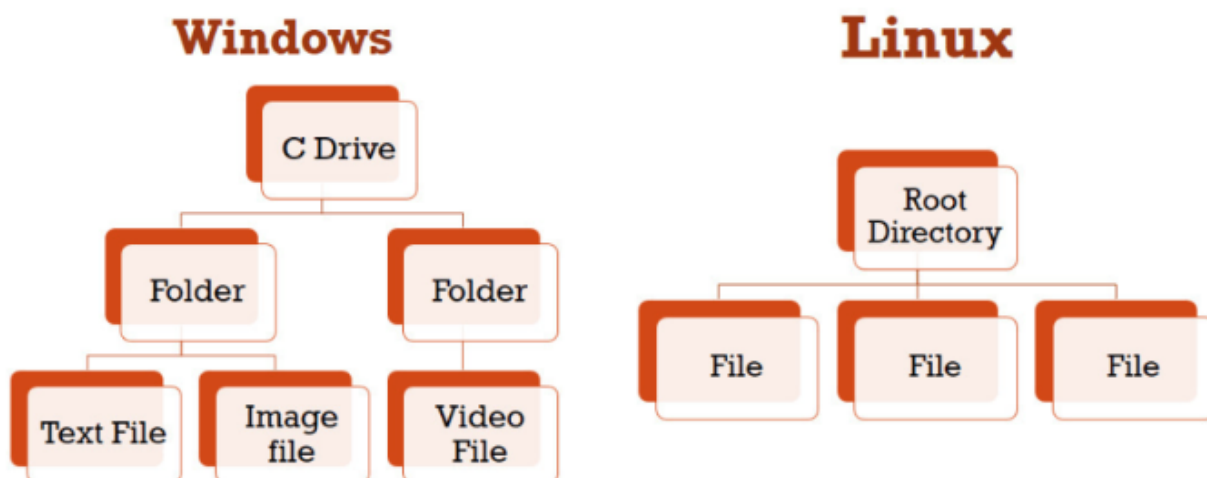
8. Continuous Learning and Improvement

- **Staying Updated:**
 - Keeping abreast of Linux distributions, updates, and secure advisories.
 - Following community forums, blogs, and official documentation (Linux Foundation, Red Hat, Ubuntu).

File System of Linux:

Linux operating system used to handle the data management of the storage. helps to arrange the file on the disk storage. manages the file name, file size, creation date, and much more information about a file. The Linux file system structure is a complex hierarchy that includes multiple layers, each with its own role in managing and organizing data on storage devices. At the top of the hierarchy is the root directory (/), which contains all other directories and files. The file system is designed to manage non-volatile storage data by organizing into files and directories, which can be accessed and modified by users and applications. Linux uses different file systems such as ext4, XFS, Btrfs, JFS, and ZFS to manage and store data on storage devices. also provides a namespace, which is a way of organizing files and directories into a logical hierarchy. The namespace allows multiple users and applications to access the same storage device without conflicts.

FILE SYSTEM

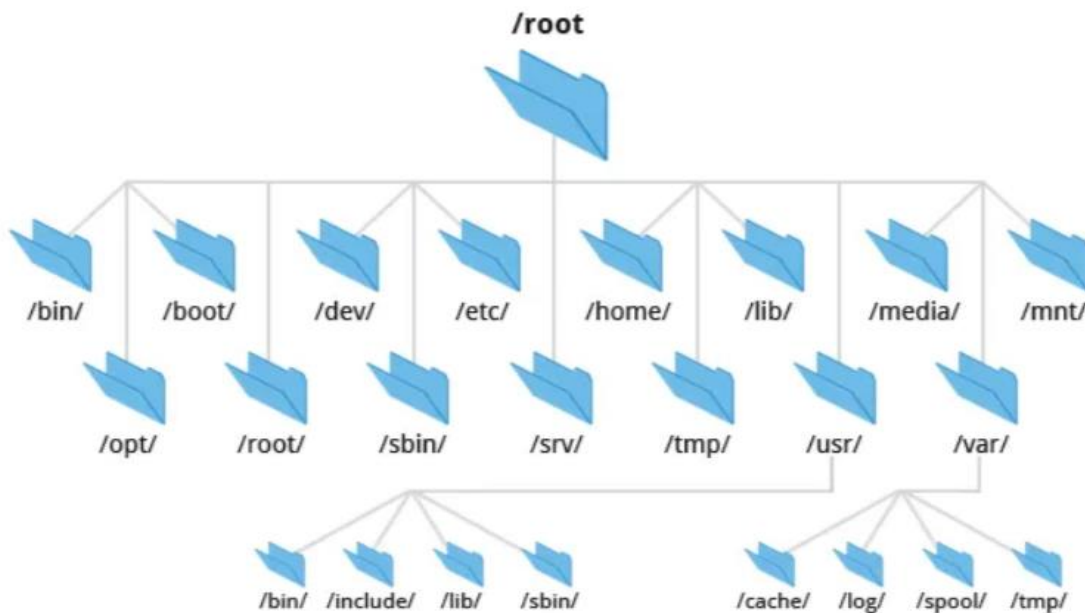


Here's an overview of the file system structure and some key concepts:

1. File System Types

Linux supports various file system:

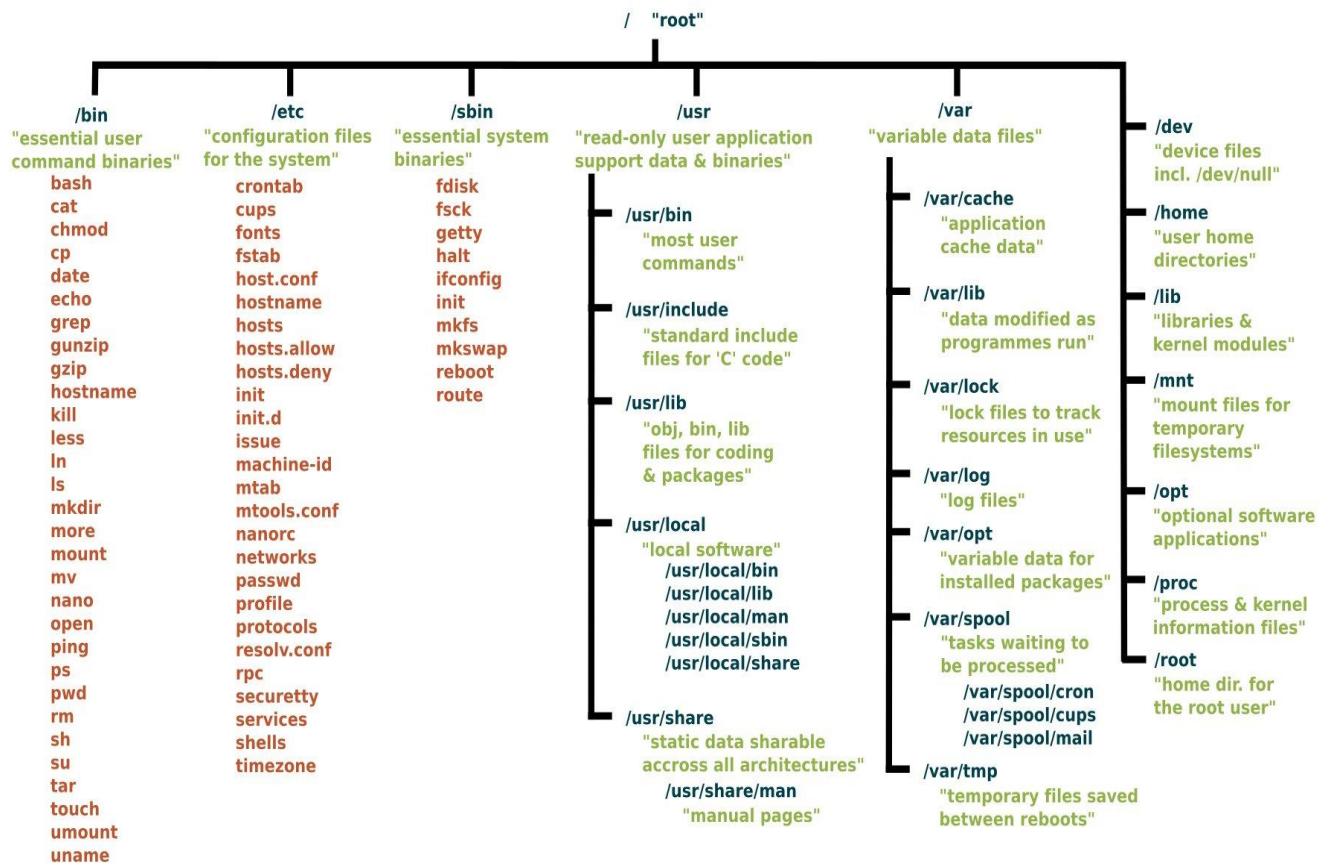
- **ext4:** The default file system for most Linux distributions, offering reliably, journaling, and support for large files and partions.
- **XFS:** Known for high-performance, scalability, and support for large files and volumes.
- **Btrfs:** Modern file system with features like snapshots, checksums, and built-in RAID support.
- **ZFS:** Advanced file system with features like data integrity, snapshotting, and volume management (commonly used on distributions like Ubuntu and Debian through ZFS on Linux).
- **FAT32 and NTFS:** File systems compatible with Windows, often used for external drives and compatibility purposes.



2. Hierarchical Structure

Linux follows a hierarchical file system structure, starting from the root directory (/) and branching out into various subdirectories. Here are some important directories and their purposes:

- **/bin**: Essential user binaries (commands) needed for system operation (e.g., ls, cp, mv).
- **/boot**: Boot loader files and kernel images.
- **/dev**: Device files representing hardware devices.
- **/etc**: System-wide configuration files.
- **/home**: User home directories.
- **/lib** and **/lib64**: System libraries.
- **/mnt** and **/media**: Mount points for temporary and removable storage devices.
- **/opt**: Optional application software packages.
- **/proc** and **/sys**: Virtual file systems providing kernel and system information.
- **/root**: Home directory for the root user.
- **/sbin**: System binaries (commands) used for system administration.
- **/tmp**: Temporary files.
- **/usr**: Secondary hierarchy with read-only user data and programs.
- **/var**: Variable files (e.g., log files, spool directories).



Linux File System Features:

In Linux, the file system creates a tree structure. The topmost directory called the root (/) directory. All other directories in Linux can be accessed from the root directory.

Some key features of Linux file system are as following:

- **Specifying paths:** Linux does not use the backslash (\) to separate the components; uses forward slash (/) as an alternative. For example, as in Windows, the data may be stored in C:\ My Documents\ Work, with areas, in Linux, would be stored in /home/ My Document/ Work.
- **Partition, Directories, and Drives:** Linux does not use drive letters to organize the drive as Windows does. In Linux, we cannot tell withether we are addressing a partition, a network device, or an "ordinary" directory and a Drive.
- **Case Sensitive:** Linux file system is case sensitive. distinguishes between lowercase and uppercase file names. Such as, there is a difference between test.txt and Test.txt in Linux. This rule is also applied for directories and Linux commands.
- **File Extensions:** In Linux, a file may have the extension '.txt,' but is not necessary that a file should have a file extension. While working with Shell, creates some problems for the beginners to differentiate between files and directories. If we use the graphical file manager, symbolizes the files and folders.
- **Hidden files:** Linux distinguishes between standard files and hidden files, mostly the configuration files are hidden in Linux OS. Usually, we don't need to access or read the hidden files. The hidden files in Linux are represented by a dot (.) before the file name (e.g., ignore). To access the files, we need to change the view in the file manager or need to use a specific command in the shell.

User and Group in Linux:

User

Create a new user account

Example: `useradd Adam` (Create a new user account named "Adam")

Set or change a user's password

Example: `passwd Adam` (Change the password for the user "Adam")

Modify user account properties

Example: `usermod -aG sudo Adam` (Add the user "Adam" to the sudo group)

Delete a user account

Example: `userdel Adam` (Delete the user account "Adam")

Group

Create a new group

Example: `groupadd linux` (Create a new group named "linux")

Modify group properties

Example: `groupmod -n new_linux linux` (Rename the group "linux" to "new_linux")

Delete a group

Example: `groupdel linux` (Delete the group "linux")

Display user and group information

Example: `id Adam` (Display information about the user "Adam")

Change file or directory ownership

Example: `chown Adam:linux file.txt` (Change the ownership of "file.txt" to user "Adam" and group "linux")

Change group ownership of files or directories

Example: `chgrp linux file.txt` (Change the group ownership of "file.txt" to "linux")

Display group membership of a user

Example: groups Adam (Display the groups that user "Adam" belongs to)

File/Folder Permissions:

In Linux systems file permissions are core to the security model. They determine who can access files and directories on a system.

This string is actually an expression of three different sets of permissions:

- `rw-`
- `r--`
- `r--`

The first set of permissions applies to the owner of the file. The second set of permissions applies to the user group that owns the file. The third set of permissions is generally referred to as "others." All Linux files belong to an owner and a group.

Within permissions and users are represented by letters, that is called symbolic mode. For users, `u` stands for user owner, `g` for group owner, and `o` for others. For permissions, `r` stands for read, `w` for write, and `x` for execute.

Within Linux file permissions are represented by numbers, it's called numeric mode. In numeric mode, a three-digit value represents specific file permissions (for example, 744.) These are called octal values. The first digit is for owner permissions, the second digit is for group permissions, and the third is for other users. Each permission has a numeric value assigned to it:

- `r` (read): 4
- `w` (write): 2
- `x` (execute): 1

drwxrwxrwx


d = Directory

r = Read

w = Write

x = Execute

chmod 777



rwx | rwx | rwx
Owner | Group | Others

7	rwX	111
6	rw-	110
5	r-X	101
4	r--	100
3	-wX	011
2	-w-	010
1	--X	001
0	---	000

In the permission value 744, the first digit corresponds to the user, the second digit to the group, and the third digit to others. By adding up the value of each user classification, you can find the file permissions.

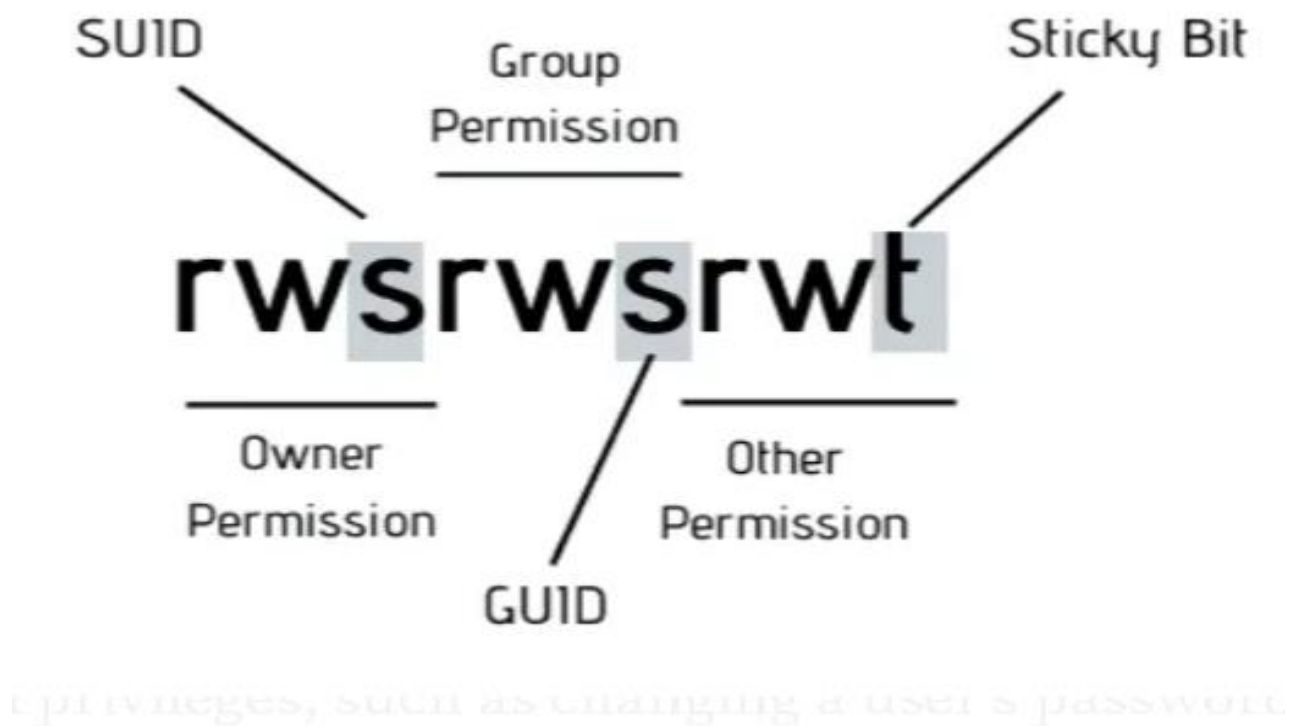
For example, a file might have read, write, and execute permissions for its owner, and only read permission for all other users. That looks like this:

- Owner: $rwX = 4+2+1 = 7$
- Group: $r-- = 4+0+0 = 4$
- Others: $r-- = 4+0+0 = 4$

The results produce the three-digit value 744.

Special Permissions

In Linux, special permissions add an extra layer of control to file and directory access. These permissions—Sticky Bit, SGID (Set Group ID), and SUID (Set User ID)—introduce unique functionalities to enhance security and manage file execution.



SUID Permission

SUID, short for Set User ID, is a special permission that can be assigned to executable files. When an executable file has the SUID permission enabled, it allows users without execute the file to temporarily assume the privileges of the file's owner. This means that even if a user does not have the necessary permissions to access or perform certain actions, they can do so by executing a file with the SUID permission.

For example, consider a system administrator wants to allow regular users to execute a specific system utility, such as `passwd`, which is usually restricted to privileged users. By assigning the SUID permission to the `passwd` executable, regular users can execute it and change their passwords without requiring administrative privileges.

Numeric representation

```
SETUID = 4  
SETGID = 2  
STICKY = 1  
NONE = 0  
READ = 4  
WRITE = 2  
EXECUTE = 1
```

SGID

SGID, which stands for Set Group ID, is another special permission that can be applied to executable files and directories. Within an executable file has the SGID permission enabled, it allows users with execute the file to temporarily assume the group ownership of the file. For directories with the SGID permission enabled, newly created files and directories within that directory inherit the group ownership of the parent directory rather than the user's default group ownership.

One practical use case of SGID permission is in collaborative environments where multiple users need to work on shared directories. By setting the SGID permission on a shared directory, all files and directories created within it will automatically have the same group ownership, ensuring proper collaboration and access control.

Sticky Bit

The sticky bit is a special permission that can only be set on directories. Within the sticky bit is enabled on a directory, it restricts the ability to delete or rename files within that directory to the file owner, the directory owner, and the superuser. It ensures that each user can only remove or modify their files, even if they have written permissions on the directory.

An illustrative example of the sticky bit's usefulness is the /tmp directory, which is commonly used for temporary file storage. By enabling the sticky bit on /tmp, users can safely create and delete their temporary files without worrying about other users accidentally or maliciously tampering with them.

Here, from left to right, the character # represents an access level. There are three access levels—user, group, and others. To determine what each digit is, we use the following:

- Start at 0
- If the **read** permission should be set, add 4
- If the **write** permission should be set, add 2
- If the **execute** permission should be set, add 1

This is calculated on a per access level basis. Let's interpret this permissions example:

-rw-r-x---

The permissions are represented as 650. How did I arrive at those numbers?

- The user's permissions are: rw- or 4+2=6
- The group's permissions are: r-x or 4+1=5
- The others's permissions are: --- or 0

To put this into the command syntax, it looks like this:

chmod 650 test.txt

Special permissions make up a fourth access level in addition to **user**, **group**, and **other**. Special permissions allow for additional privileges over the standard permission sets (as the name suggests). There is a special permission option for each access level discussed previously. Let's take a look at each one individually, beginning with Set UID:

user + s

Commonly noted as **SUID**, the special permission for the user access level has a single function: A file with **SUID** always executes as the user who owns the file, regardless of the user passing the command. If the file owner doesn't have execute permissions, then use an uppercase **S** here.

Now, to see this in a practical light, let's look at the **/usr/bin/passwd** command. This command, by default, has the **SUID** permission set:

```
$ ls -l /usr/bin/passwd
-rwsr-xr-x. 1 root root 33544 Dec 13 2019 /usr/bin/passwd
```

Note the **s** where **x** would usually indicate execute permissions for the user.

group + s

Commonly noted as **SGID**, this special permission has a couple of functions:

- If set on a file, it allows the file to be executed as the **group** that owns the file (similar to SUID)
- If set on a directory, any files created in the directory will have their **group** ownership set to that of the directory owner

```
$ ls -l
total 0
drwxrws---. 2 tcarrigan tcarrigan 69 Apr 7 11:31 my_articles
```

Disk Management:

Disk management in Linux involves several key commands to handle tasks such as viewing, partitioning, formatting, mounting, and monitoring disks. Here's a detailed look at these commands:

lsblk

- **Example Output:**

```
plaintext
Copy code
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
sda   8:0  0 100G 0 disk
├─sda1 8:1  0  50G 0 part /
├─sda2 8:2  0  20G 0 part /home
└─sda3 8:3  0  30G 0 part /var
```

Lists the partition tables of all disks.

```
sudo fdisk -l
```

- **Example Output:**

```
Disk /dev/sda: 100 GiB, 107374182400 bytes, 209715200 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x12345678
```

sblk: List block devices

Example: lsblk (List all available block devices)

fdisk: Manipulate disk partition table

Example: fdisk -l (List partitions on all disks)

Example: fdisk /dev/sda (Interactively manage partitions on a specific disk)

parted: Create, delete, resize partitions Example: parted /dev/sdb (Interactively manage partitions on a specific disk)

mkfs: Create a file system Example: mkfs. ext4 /dev/sda1 (Create an ext4 file system on a specific partition)

mount: Mount a file system Example: mount /dev/sda1 /mnt (Mount a file system on a specific mount point)

umount: Unmount a file system

Example: umount /mnt (Unmount a file system from a mount point)

df: Show disk space usage

Example: df -h (Display disk space usage in human-readable format)

du: Estimate file and directory space usage

Example: du -sh /path/to/directory (Estimate the space usage of a specific directory)

lsblk: Identify disk partitions and their sizes

Example: lsblk -o NAME,SIZE,MOUNTPOINT (List disk partitions with their sizes and mount points)

lvm: Manage Logical Volumes (LVM)

Example: lvcreate -L 10G -n myvolume myvg (Create a logical volume of size 10GB)

Service and Process Management

The standard service manager for most modern Linux distributions. It controls how services are started, stopped, and managed at boot and during runtime.

Key Commands:

Use systemctl to interact with systemd services. You can:

- Start: `systemctl start <service_name>` (e.g., `systemctl start apache2` to start the Apache web server)
- Stop: `systemctl stop <service_name>`
- Restart: `systemctl restart <service_name>`
- Enable (start at boot): `systemctl enable <service_name>`
- Disable (prevent auto-start): `systemctl disable <service_name>`
- Check status: `systemctl status <service_name>`
- Display logs: `journalctl -u < service_name >`