Linux

Contents:

Commands

1. PWD - Present Working Directory
   - ⇨ It Displays present working directory

   Syntax: pwd
2. Ls – List of files
   - ⇨ It displays list of files or directories of present working directory

   Syntax: **ls**
   - ⇨ ls / -It Displays list of files or directories in root directory.
   - ⇨ ls -l : To view the long list of file and directory.
   - ⇨ ls -a: To view the hidden files.
   - ⇨ ls -al: To view the hidden files in long format.
   - ⇨ ls D*: It display the file or directory, starting character with D.
   - ⇨ ls -d ????: It displays the file or directory, whose length is of 4 characters.
   - ⇨ ls dir_name: Display the list of files or directories in the specific directory.
3. Cd - Change directory
   - ⇨ It is used to switch from one directory to another directory.

   Syntax: cd
   - ⇨ **cd dir_name**: It change to the specified directory.
   - ⇨ **cd ..** : It goes to one step back means previous working directory.
   - ⇨ **cd ../..** : It goes to two step back.
4. Cat : Concatenate and combining files.
   - ⇨ It is used create a file.
   - ⇨ It is used to display the content of the file.
   - ⇨ It is used to combining of files and copying the content to the another file.
   - ⇨ It is used to append the content to the file.

Syntax: cat > file_name – It creates a file and then prompt us to write content. Use ctrl+d to save the content.

- ⇨ **cat file_name** : It displays the content of file.
- ⇨ **cat file_name1 file_name2 > file_name3** – Combining of two files and copying the content to the another new file.
- ⇨ **cat >> file_name1:** Appending the content to the existing file.

5. Touch : Creates an empty file
   - ⇨ It is used to create empty file and also creates multiple empty files at a time.

   Syntax: touch file_name – Creates an empty file.
   - ⇨ **Touch file_name1 file_name2** – Creates multiple empty files simultaneously.
6. MKDIR – Making Directory
   - ⇨ It creates directory and also create directory structure.

   Syntax: mkdir dir_name – Creates a directory.
   - ⇨ **mkdir -p dir_structure –** Creates a nested directory structure.
   - ⇨ **Note :** To view the nested directory structure, we follow below syntax
   - ⇨ **ls -R**
7. RMDIR – Remove Directory.
   - ⇨ It removes the directory and also removes the directory structure.

   Syntax: **rmdir dir_name** – Removes a directory.
   - ⇨ **rmdir dir_structure** – Removes a top level sub-directory in the directory in the directory structure.
   - ⇨ **rmdir -p dir_structure** – Removes a directory structure with sub-directories.
8. RM – Remove files and directories.
   - ⇨ It removes files and directories in the home directory.

   Syntax: **rm file_name** – Removes file.
   - ⇨ rm -rf dir_structure – Removes the whole directory structure.
9. Mv – Rename and move.
   - ⇨ It is used to move the file from one file to another.

   Syntax: **mv file_name1 file_name2** – It moves the content from file_name1 to the file_name2. There is no content in the file_name1
10. Cp – Copying the file.
    - ⇨ It is used to copying the file.

    Syntax: cp file_name1 file_name2 – It copying the content from file_name1 to the file_name2. Both the files having the same content.
11. Echo – It is used to displaying lines of text or string and use directions to store the content in a new file.

    Syntax: echo "Write content here" > file_name

SOME HELP COMMANDS

1. **whatis <command_name>:** The command gives a one-line description of a command.

2. **Man <command_name>:** The man command provides detailed documentation about commands.
3. **Info <command_name>:** The info command shows a more detailed and structured form of documentation, often used for GNU utilities.
4. **<command_name> --help:** Many commands have a --help flag that displays a brief summary of how to use the command and its options**.**
5. **Which <command_name>:** The which command shows the location of an executable file for a command.

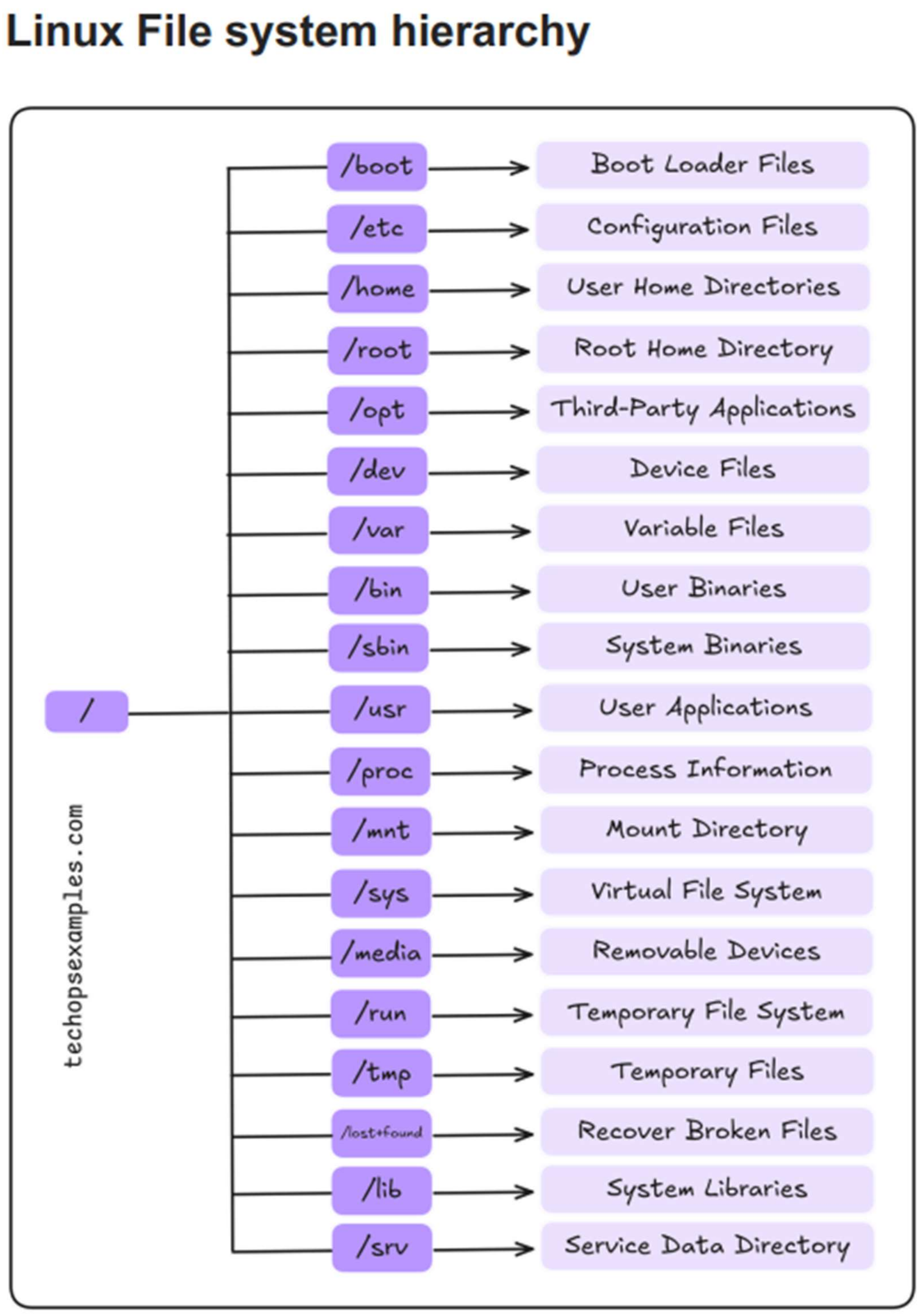
FILE SYSTEM HIERARCHY

The Linux File System is also known as the **Filesystem Hierarchy Standard(FHS)**, provides a standard directory structure for Unix-Like Operating Systems.

Directories and their Purpose:

1. **Root Directory(/):** It is base of the file system hierarchy, where all other directories and files are nested. It's the starting point of the directory structure.
2. **Boot Loader Files(/boot):** It stores boot loader files, including the Linux kernel and initial RAM disk.
3. **Configuration Files (/etc):** It contains system-wide configuration files. It is important to note that /etc should hold static configuration files and not binaries.
4. **User Home Directories (/home):** The directory for user home directories, where personal files and user-specific settings and stored.
5. **Root Home Directory (/root):** The home directory of the root (superuser), containing configuration and environment files specific to root. Only the root user has access to this directory.

6. **Process Information (/proc):** A virtual file system that provides process and kernel information as files, dynamically generated by the sys



7. **Mount Directory (/mnt) :** It is used for temporarily mounting file systems.
8. **Virtual File System (/sys):** It provides information about devices, drivers, and some kernel features.
9. **Removable Devices (/media):** It serves as a mount point for removable media like CD-ROMs.

10. **Temporary File System (/run):** It contains run-time variable data, such as information about the system since the last boot.
11. **Temporary Files (/tmp):** A directory for temporary files, which may not be preserved between system reboots.
12. **Recover Broken Files (/lost+found)**: Created during the recovery process after a system crash or improper shutdown. It contains recovered files that were part of the file system but became corrupted or lost.
13. **System Libraries (/lib):** It includes essential libraries needed for the binaries in /bin and /sbin.
14. **Service Data Directory (/srv):** Contains data for services offered by the system, such as web server data for Apache or FTP data for FTP servers. This directory holds files related to server operations.
15. **Third-Party Applications (/opt):** Contains add-on software packages and third-party applications that are not part of the default system. For example, large software installations like Java or other proprietary programs can be placed here.
16. **Device Files (/dev):** It contains houses device files that represent hardware components and other system devices.
17. **Variable Files (/var):** It contains variable files, such as logs and spool files, whose content is expected to continually change during normal operation.
18. **User Binaries (/bin):** It contains essential user command binaries that are required for basic system operation and are available to all users, such as \`ls\`, \`cp\`, \`mv\`, and \`rm\`. These are crucial commands for interacting with the system.
19. **System Binaries (/sbin):** It stores essential system binaries, like fsck, init, and route.
20. **User Applications (/usr):** A second hierarchy for read-only user data, including the majority of user utilities and applications.
21. **Process Information (/proc):** A virtual file system that provides process and kernel information as files, dynamically generated by the sys.

EDITORS IN LINUX

The most commonly used text editors in Linux can vary depending on the user's expertise, environment (command-line or graphical), and purpose.

1. Vim/Vi (Command-Line)

Popularity: Widely used by system administrators, developers, and advanced users.

Reason: Pre-installed on almost every Linux distribution, powerful for fast text manipulation, and highly customizable.

Strengths:

a. Extremely efficient for large-scale editing tasks.
b. Once familiar, it can dramatically speed up text manipulation

Syntax: vi file_name

2. Nano (Command-Line)

Popularity: Popular among beginners and for quick file edits.

Reason: Simple to use, user-friendly, and often set as the default text editor on many distributions.

Strengths:

a. Ideal for quick, straightforward edits in the terminal.
b. No steep learning curve compared to Vim or Emacs

Syntax: nano file_name

3. Emacs (Command-Line and Graphical)

Popularity: Preferred by developers, power users, and enthusiasts who value extensibility.

Reason: Highly customizable with support for extensive plugins and can serve as more than just a text editor (e.g., for project management or email).

Strengths:

a. Highly extensible with Lisp-based scripting.
b. Versatile (can be used for coding, note-taking, task management, etc.).

Syntax: emacs <filename>

4. gedit (Graphical, GNOME Default Editor)

Popularity: Commonly used in GNOME desktop environments for casual or simple text editing.

Reason: Lightweight, user-friendly, and comes pre-installed with GNOME desktop environments.

Strengths:

a. Clean graphical interface.
b. Good for basic text editing, configuration files, and script writing.

Syntax: gedit <filename>

5. Visual Studio Code (VS Code) (Graphical)

Popularity: Extremely popular among developers, especially for coding projects.

Reason: Extensive plugin marketplace, cross-platform, and support for many programming languages.

Strengths:

a. Rich features for coding, such as IntelliSense, debugging, and Git integration.
b. Lightweight and extendable with numerous plugins.

Syntax: code <filename>

PERMISSIONS AND OWNERSHIP IN LINUX

In Linux, permissions and ownership are fundamental concepts for file management and security. They control who can read, write, or execute files and directories.

1. Ownership

Every file and directory in Linux has an owner and a group associated with it. There are three types of ownership:

- **User (Owner):** The user who owns the file. This is usually the person who created the file.
- **Group:** A set of users who share access to the file. Any user in the group can be given specific permissions to the file.
- **Others:** All other users who are not the owner or part of the group

2. Permissions

Linux permissions define what actions the owner, group, and others can perform on a file or directory. There are three types of permissions:

- **Read (r):** Allows a user to read the contents of the file or list the contents of a directory.
- **Write (w):** Allows a user to modify the contents of a file or create and delete files within a directory.
- **Execute (x):** Allows a user to execute a file (if it is a script).

These permissions can be represented numerically:
 * r = 4
 * w = 2
 * x = 1

For each user category (Owner, Group, Others), permissions are represented as a three-digit number:
* 777: Full permissions (rwx) for Owner, Group, and Others.
* 755: Full permissions (rwx) for the Owner, and read/execute (r-x) for Group and Others. * *
* 644: Read/write (rw-) for the Owner, and read-only (r--) for Group and Others.

Example Breakdown of 777

* Owner (7): Read (4), Write (2), Execute (1) = 7 (rwx)
* Group (7): Read (4), Write (2), Execute (1) = 7 (rwx)
* Others (7): Read (4), Write (2), Execute (1) = 7 (rwx)

COMMANDS:

1. **ls -l:** Displays a detailed list of files and directories, including their permissions and ownership

2. **Chown:** To change the ownership of a file or directory
   Syntax: chown user_name file_name

   To view the file, use the following command.
   ls -l
3. **Chgrp:** Changes group ownership of a file or directory**.**
   Syntax: chgrp g file_name

4. **Chmod:** Changes the permissions of a file or directory.
   Syntax: chmod [permissions] file_name

Methods for changing permissions:

a. **Symbolic Method:** Symbols are used with +, -, and = to add, remove, or set permissions.
   - ⇨ **chmod u+x file.txt:** Add execute permission for the owner (user)
   - ⇨ **chmod g-w file.txt:** Remove write permission for the group
   - ⇨ **chmod a=r file.txt:** Set read-only permissions for everyone

b. **Numeric (Octal) Method**:
   - ⇨ chmod 755 file.txt: Add all permissions to the user, add read and execute permissions to the group, add read and execute permissions to the others.

PROCESSES AND SIGNAL COMMANDS IN LINUX

In a Linux system, processes represent running programs or tasks. Every process is assigned a unique Process ID (PID). Managing these processes is essential for system performance, debugging, and ensuring stability.

Signals allow you to communicate with processes, for instance, to terminate or modify their behaviour, making it easier to automate tasks, manage workloads, and maintain system health

COMMANDS:

1. **ps:** Displays information about running processes.
   Syntax: ps
   - ⇨ **ps aux:** Displays all running processes in detail for all users.
   - ⇨ **ps -ef**: Another format to list all processes in full detail.

2. **Top:** Displays a real-time view of active processes.
   Syntax: top -This command shows processes along with CPU and memory usage, and updates dynamically**.**
3. **Kill:** Sends a signal (usually to terminate) to a process.

Syntax: kill signal PID -This sends the signal to forcibly terminate process with PID.

4. **Killall:** Kills all processes by name, rather than by PID.
   Syntax: killall
   - ⇨ **killall process_name -**This kills all instances of the firefox process.
5. **Htop:** A more user-friendly version of `top` that provides better visuals and controls.
   Syntax: htop - Allows you to easily kill processes, view tree structures
6. **Nice:** Adjusts the priority (niceness) of a process**.**
   Syntax: nice -n priority command
7. **Renice:** To change the priority of an already running process
   Syntax: renice [priority] PID

PACKAGE MANAGEMENT IN LINUX

Package Management in Linux refers to the process of installing, updating, and removing software packages on a Linux system. Different Linux distributions use different package managers, but the core concepts remain similar.

Debian-based systems (e.g., Ubuntu, Debian):

1. **APT** (Advanced Package Tool)
2. Package files: .deb

Red Hat-based systems (e.g., CentOS, Fedora, RHEL):

3. **YUM** (Yellowdog Updater Modified) / DNF (Dandified YUM)
4. Package files: .rpm

COMMANDS:

1. **sudo apt update:** Update package lists
   - ⇨ **sudo apt upgrade:** Upgrade installed packages
   - ⇨ **sudo apt install package_name:** Install a package
   - ⇨ **sudo apt remove package_name:** Remove a package
   - ⇨ **sudo apt purge package_name:** Purge a package (remove along with configuration files)
   - ⇨ **apt search package_name:** Search for a package
   - ⇨ **apt show package_name:** show package details.
2. **sudo dnf check-update:** Update package lists
   - ⇨ **sudo dnf upgrade:** Upgrade installed packages
   - ⇨ **sudo dnf install package_name:** Install a package
   - ⇨ **sudo dnf remove package_name:** Remove a package
   - ⇨ **dnf search package_name:** Search for a package
   - ⇨ **dnf info package_name:** show package details.

NETWORKING COMMANDS

1. **Ifconfig:** It Displays or configures network interface, such as IP addresses.
   Syntax: ifconfig eth0 – It displays interface information.
   - ⇨ **Ifconfig enpos3 192.189.2.6 netmask 255.255.255.8 –** It changes the IP address and also netmask in the interface.
   - ⇨ **sudo ifconfig down:** It is used to disable or shut down the network interface.
   - ⇨ **sudo ifconfig up:** It is used to enable or bring up a network interface.
2. **Ping**: It checks the connectivity between your system and a remote host.
   Syntax: **ping google.com** - Displays network connectivity between your machine and Google's servers.
3. **Netstat:** Displays network connections, routing tables, interface statistics**.**
   Syntax: netstat- It displays a list of active connections for TCP, UDP, and other protocols.
   - ⇨ netstat -l: It displays only the ports that are in a **listening** state.
   - ⇨ netstat -lt: It displays all the TCP ports that are currently in the listening state.
   - ⇨ netstat -lu: It displays all the UDP ports that are currently in the listening state.
   - ⇨ netstat -n: It displays active connections using numeric IP addresses.
   - ⇨ netstat -t: It displays lists all active TCP connections.
   - ⇨ netstat -u: It displays lists all active UDP connections.
4. **Curl:** It transfers data from or to a server using various protocols (HTTP, FTP, etc.).
   Syntax: curl http://example.com
5. **Wget:** It is used for downloading files from the web.
   Syntax: wget http://example.com/file.zip
6. **Nslookup:** The nslookup command in Linux is used to query Domain Name System (DNS) servers and obtain information about domain names or IP addresses. It is primarily used for troubleshooting DNS-related issues and performing DNS lookups.
   Syntax: nslookup google.com- This command queries the default DNS server and returns the IP addresses associated with google.com.
   - ⇨ **nslookup 8.8.8.8 -** This performs a reverse DNS lookup, translating an IP address back to its domain name.
   - ⇨ **nslookup google.com 8.8.8.8**: This command performs a DNS query for google.com using Google's public DNS server (8.8.8.8) instead of the system's default DNS server.
7. **Traceroute:** The traceroute command in Linux is used to trace the path that packets take to reach a destination, showing each hop along the route from your machine to the target.
   Syntax: traceroute google.com- This command will show the route that packets take to reach Google's servers, including all intermediate routers (hops) along the way.
   - ⇨ **traceroute -m 10 google.com**: Limits the traceroute to a maximum of 10 hops.

DISK MANAGEMENT COMMANDS

It is used to managing storage devices, partitions, and file systems. Here are some commonly used commands for managing disks in Linux.

1. **Df (Disk Usage):** It shows the amount of disk space used and available on file systems.
   Syntax: df -h (h – human readable format)
2. **Du (Directory Disk Usage):** It summarizes disk usage for directories and files.
   Syntax: du -sh /home/user/ (s – shows size)
3. **Fdisk (Disk partitioning):** A command-line utility to create, modify, or delete partitions on a disk.
   Syntax: sudo fdisk/dev/sda
4. **Mount (Mount File Systems):** It is used to mounts a file system to a specified directory.
   Syntax: sudo mount /dev/sda1 /mnt
   Note: To view the mount file systems use the following command**.**
   mount | grep "^/dev"
5. **Umount (Unmount File Systems):** Unmounts a file system
   Syntax: sudo umount /mnt
6. **Lsblk (List Block Devices):** Displays information about all available block devices (disks and partitions).
   Syntax: lsblk

USER AND GROUP MANAGEMENT

In Linux, managing users and groups is essential for system administration, access control, and security.

1. **Useradd:** Adds a new user to the system**.**
   Syntax: useradd user_name
   - ⇨ **useradd -m -d /home/newuser user_name -** To specify a home directory with useradd.

     To check the groups, use the following command
     tail /etc/passwd
   - ⇨ **useradd -u 2024 -c "Hi" -d /home/newuser -s /bin/csh user_name** – Displays new user with all the properties.
2. **Passwd (Set or Change Password):** Sets or changes the password for a user.
   Syntax: sudo passwd username

   To view the passwd properties use the following command
   grep user_name /etc/shadow

3. **Userdel (Delete a User):** Deletes a user from the system.
   Syntax: userdel user_name
   - ⇨ **userdel -r username:** remove the user's home directory.

4. **Usermod (Modify User Account):** Modifies a user's settings such as username, home directory, group, etc
   - ⇨ **sudo usermod -l newname oldname:** This command changes the login name of a user from oldname to newname.
   - ⇨ **sudo usermod -d /new/home/directory -m username:** Use this to change the location of a user's home directory. The -m option moves the content of the current home directory to the new directory.
   - ⇨ **sudo usermod -L username:** Locks a user's account, preventing them from logging in.
   - ⇨ **sudo usermod -U username:** Unlocks a previously locked account.
   - ⇨ **sudo usermod -u 1001 username:** The -u option changes the user's numerical user ID.
   - ⇨ **sudo usermod -r username:** The -r option modifies the user to a system user account (typically used for system services).
   - ⇨ **sudo usermod -s /bin/bash username:** The -s option changes the default shell for the user (e.g., changing to /bin/bash).

5. **Groupadd (Create a new Group): Adds a new group to the system.**
   Syntax: sudo groupadd groupname
6. **gpasswd** (Administer / Assign Group Passwords): Assigns a password to a group
   Syntax: sudo gpasswd groupname
7. **groups** (Show Groups for a User): Displays the groups a user is a member of.
   Syntax: groups username
8. **groupdel (Delete the group): Removes a group from the system.**
   Syntax: sudo groupdel group_name
9. **groupmod:** The groupmod command in Linux is used to modify an existing group's attributes, such as changing the group name or the Group ID (GID).
   Syntax: sudo groupmod [options] groupname
   - ⇨ **sudo groupmod -n new_name old_name:** You can use groupmod to rename an existing group.
   - ⇨ **sudo groupmod -g 1001 developers:** You can change the numeric Group ID (GID) of an existing group.
   - ⇨ **sudo groupmod -n <name> -g <id> staff:** You can change both the name and GID of a group in a single command.