MAYANK PRATAP SINGH

COMPLETE NOTE OF LINUX

1. Introduction to Linux

- **What is Linux?**

Introduction

What is Linux

Tux, the penguin, mascot of the Linux kernel

OS family:	Unix-like
Latest stable release:	2.6.22.5 (Linux kernel) / August 23, 2007
Kernel type:	Monolithic kernel
License:	Kernel: GNU General Public License/various
Working state:	Current

- Unix-like computer operating system
- One of the most prominent examples of free software and open source development
  - Underlying source code can be freely modified, used, and redistributed by anyone
  - Originally only ran on x86 processors

Linux is a versatile and powerful open-source operating system kernel that serves as the foundation for various operating systems known as Linux distributions or distros. Developed initially by Linus Torvalds in 1991, Linux is widely used across various platforms, from personal computers to servers, embedded systems, and supercomputers.

Key Features

1. Open Source

- **Definition**: The source code of Linux is freely available for anyone to view, modify, and distribute.
- **Benefit**: Encourages community collaboration and innovation, ensuring continuous improvement and customization.

2. Multitasking

- **Definition**: Linux can run multiple processes simultaneously.
- **Benefit**: Enhances productivity and efficiency, making it ideal for both desktop and server environments.

3. Multiuser Capabilities

- **Definition**: Supports multiple users accessing the system concurrently with separate user accounts and permissions.
- **Benefit**: Facilitates secure and organized user management, especially in multi-user environments like servers.

4. Security

- **Definition**: Linux incorporates advanced security features such as permissions, access controls, and encryption.
- **Benefit**: Provides a robust security framework, reducing vulnerabilities and protecting data.

5. Flexibility

- **Definition**: Linux is highly customizable, allowing users to tailor the system to their specific needs.
- **Benefit**: Supports a wide range of hardware and software configurations, making it suitable for diverse applications.

Components of Linux

1. Kernel

- **Definition**: The core component of Linux that manages hardware resources and provides essential system functions.
- **Function**: Handles process management, memory management, device drivers, and system calls.

2. Shell

- **Definition**: The command-line interface that allows users to interact with the system using text commands.
- **Function**: Provides a means to execute commands, run scripts, and manage system tasks.

3. File System

- **Definition**: The organization of data on storage devices, defining how files and directories are stored and accessed.
- **Function**: Provides a hierarchical structure for data storage and retrieval.

4. Utilities

- **Definition**: A collection of software tools and applications that perform various system functions.
- **Function**: Includes tools for file manipulation, system monitoring, network management, and more.

5. Distributions

- **Definition**: Various versions of Linux, each tailored for specific uses and audiences, such as Ubuntu, Fedora, CentOS, and Debian.
- **Function**: Includes the Linux kernel along with additional software, tools, and configurations suited for different needs.

Common Uses

- **Personal Computers**: Linux is used on desktops and laptops, providing a user-friendly environment for general computing tasks.
- **Servers**: Powers web servers, databases, and other enterprise applications due to its stability, performance, and security.

- **Embedded Systems**: Found in devices like routers, smartphones, and IoT devices, offering a lightweight and efficient operating system.
- **Supercomputers**: Used in high-performance computing environments for complex simulations and research.

Advantages

- **Cost**: Linux is free to use, reducing the cost of software licensing.
- **Stability**: Known for its reliability and uptime, making it suitable for mission-critical applications.
- **Community Support**: A large and active community provides support, documentation, and continuous development.

- **History of Linux**

Introduction

Linux is a widely-used open-source operating system kernel that has evolved significantly since its inception. Its history reflects a journey from a personal project to a global phenomenon, influencing various sectors from personal computing to enterprise servers and embedded systems.

Key Milestones

1. Early Beginnings (1991)

- **Linus Torvalds**: Linux was started by Linus Torvalds, a Finnish computer science student, on August 25, 1991. He announced his project on the comp.os.minix newsgroup, describing it as a free operating system kernel for 386(486) AT clones.
- **Initial Release**: The first version, Linux 0.01, was released in September 1991. It was a minimalistic kernel designed to run on Intel x86 architecture.

2. Growth and Development (1992-1994)

- **Open Source License**: In 1992, Linux was released under the GNU General Public License (GPL) by the Free Software Foundation, allowing anyone to freely use, modify, and distribute the software.

- **Kernel Improvements**: Over the next few years, Linux saw numerous improvements, including support for more hardware and features. The kernel's version 1.0 was released on March 14, 1994.

3. Expansion and Adoption (1995-1999)

- **Distributions Emergence**: Various Linux distributions (distros) began to appear, such as Red Hat, Debian, and Slackware. These distributions bundled the Linux kernel with other software to create complete operating systems.
- **Corporate Interest**: Companies like IBM and Compaq started showing interest in Linux, contributing to its development and integration into business environments.

4. Mainstream Acceptance (2000-2004)

- **Enterprise Adoption**: Linux gained traction in the enterprise sector with the release of Red Hat Enterprise Linux (RHEL) and Novell’s SUSE Linux Enterprise. It became known for its stability and cost-effectiveness in server environments.
- **Desktop Use**: Distributions like Ubuntu, released in 2004, made Linux more accessible to desktop users with a focus on user-friendliness and ease of installation.

5. Maturity and Innovation (2005-Present)

- **Wide Adoption**: Linux became a major player in cloud computing, supercomputing, and mobile devices (especially Android, which is based on the Linux kernel).
- **Ongoing Development**: The Linux kernel has continued to evolve with contributions from a global community of developers. As of now, Linux powers millions of servers, desktops, and embedded systems worldwide.

Key Contributions

- **Linus Torvalds**: Initiated and continues to lead the Linux kernel development.
- **Open Source Community**: A diverse group of developers and organizations that contribute to and support the Linux ecosystem.

Impact

- **Innovation**: Linux has driven innovation in software development, system architecture, and cloud computing.
- **Cost**: Provides a free and open-source alternative to proprietary operating systems, reducing software costs for businesses and individuals.
- **Flexibility**: Supports a wide range of hardware platforms and use cases, from personal desktops to high-performance computing clusters.

- **Linux Distributions**: Debian, Fedora, Ubuntu, CentOS, Arch Linux, etc.

Linux Distributions Overview

1. Debian

- **Overview**: Debian is one of the oldest and most influential Linux distributions, known for its stability and robustness. It serves as the foundation for many other distributions.
- **Key Features**:
    - **Stability**: Debian focuses on stability and reliability, making it a popular choice for servers and production environments.
    - **Package Management**: Uses the APT (Advanced Package Tool) system and .deb packages.
    - **Community-Driven**: Developed by a large community of volunteers and contributors.

2. Fedora

- **Overview**: Fedora is a cutting-edge distribution sponsored by Red Hat. It serves as a testing ground for new technologies that may later be incorporated into Red Hat Enterprise Linux (RHEL).
- **Key Features**:
    - **Innovation**: Often features the latest software and technologies.
    - **Package Management**: Uses the RPM (Red Hat Package Manager) system and .rpm packages.

    - **Red Hat Integration**: Acts as a proving ground for technologies that will be used in RHEL.

3. Ubuntu

- **Overview**: Ubuntu is one of the most popular Linux distributions for desktop and server use, derived from Debian. It aims to be user-friendly and accessible.
- **Key Features**:
    - **Ease of Use**: Known for its user-friendly interface and ease of installation.
    - **Long-Term Support (LTS)**: Offers LTS versions with extended support and updates for five years.
    - **Package Management**: Uses APT and .deb packages.
    - **Community and Commercial Support**: Backed by Canonical Ltd. and a large user community.

4. CentOS

- **Overview**: CentOS (Community ENTerprise Operating System) was a free and open-source alternative to Red Hat Enterprise Linux (RHEL). It has been discontinued and succeeded by CentOS Stream.
- **Key Features**:
    - **Enterprise Focus**: Aimed at providing a free and stable alternative to RHEL for servers.
    - **Compatibility**: Designed to be binary-compatible with RHEL.
    - **Package Management**: Uses RPM and .rpm packages.
    - **CentOS Stream**: A rolling-release distribution that sits between Fedora and RHEL, providing a preview of future RHEL updates.

5. Arch Linux

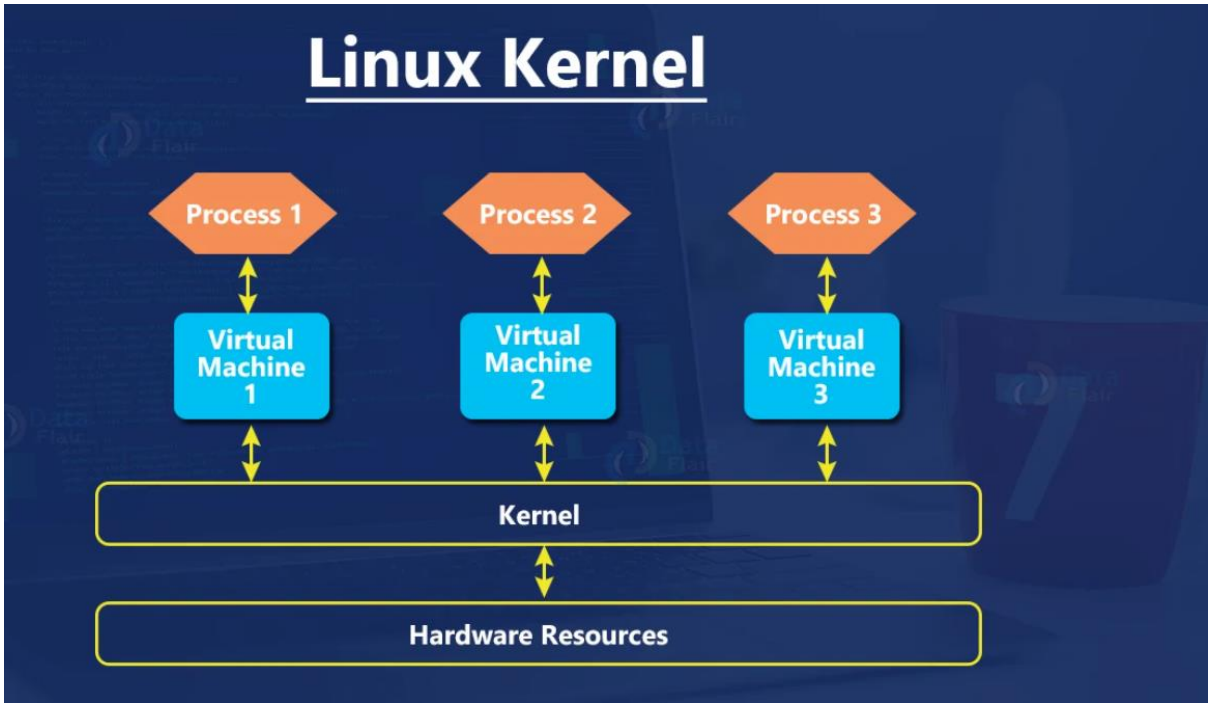
- **Overview**: Arch Linux is known for its simplicity, flexibility, and customization. It follows a rolling-release model and provides a minimal base system.
- **Key Features**:

- **Rolling Release**: Continuously updated with the latest software.
- **Customizability**: Offers a minimal base system for users to build their own customized environment.
- **Pacman**: Uses the Pacman package manager for handling .pkg.tar.zst packages.
- **Arch Wiki**: An extensive and well-maintained documentation resource.

6. Other Notable Distributions

- **openSUSE**: A versatile distribution with both stable and rolling release versions, using the YaST configuration tool and Zypper package manager.
- **Manjaro**: Based on Arch Linux, it aims to provide a more user-friendly experience with a pre-configured system and a rolling release model.
- **Kali Linux**: A distribution designed for digital forensics and penetration testing, based on Debian.

- **Linux Kernel Overview**



Linux Kernel Overview

What is the Linux Kernel?

The Linux Kernel is the core component of a Linux operating system. It is responsible for managing hardware resources, providing essential services, and enabling the interaction between software and hardware.

Key Functions of the Linux Kernel

1. **Hardware Abstraction**: The kernel provides a layer of abstraction between the hardware and user applications, allowing software to interact with hardware devices through standardized interfaces.
2. **Process Management**: It manages the creation, scheduling, and termination of processes. The kernel ensures efficient process execution and multitasking.
3. **Memory Management**: Handles the allocation and deallocation of memory to processes. It includes virtual memory management, paging, and swapping.
4. **File System Management**: Manages file systems and provides a way for applications to read and write files. It supports various file systems such as ext4, xfs, and btrfs.
5. **Device Drivers**: Includes drivers for hardware components, allowing the kernel to communicate with devices like printers, disk drives, and network interfaces.
6. **System Calls**: Provides a set of APIs through which user applications can request services from the kernel. This includes file operations, process control, and inter-process communication.
7. **Security and Access Control**: Implements security features such as user permissions, access control lists (ACLs), and security modules to protect the system from unauthorized access.

Kernel Architecture

- **Monolithic Kernel**: The Linux Kernel is a monolithic kernel, meaning it includes all core functionalities and device drivers in a single large codebase. This design provides high performance and efficiency.
- **Modules**: The kernel can be extended with loadable kernel modules (LKMs) that add functionality or support new hardware without requiring a reboot.

- **User Space vs. Kernel Space**: The kernel operates in kernel space with privileged access to hardware, while user applications run in user space with restricted permissions.

Versions and Releases

- **Stable Releases**: Linux Kernel releases are categorized into stable, long-term support (LTS), and development versions. LTS versions receive extended support and updates.
- **Development Cycle**: New kernel versions are released approximately every 2-3 months, with major changes and features incorporated into each new release.

2. Basic Commands and Navigation

Basic Commands:

1. **pwd - Show current directory**
2. **ls - List files and directories**
3. **cd - Change directory**
4. **mkdir - Create a new directory**
5. **rmdir - Remove an empty directory**
6. **rm - Delete files or directories**
7. **cp - Copy files or directories**
8. **mv - Move or rename files or directories**
9. **cat - Display file content**
10. **more / less - View file content page by page**
11. **man - Display command manual**
12. **grep - Search for text in files**

Navigation:

- **~ - Home directory**
- **.. - Parent directory**

- **. - Current directory**
- **Tab - Auto-complete file and directory names**

- **The Linux Command Line Interface (CLI)**

What is the CLI?

- **Command Line Interface (CLI):** A way to interact with the operating system by typing commands in a terminal window.

Key Components:

1. **Terminal/Console:** The interface where you enter commands.
2. **Shell:** The command interpreter that processes commands. Common shells include bash, zsh, and sh.

Basic Usage:

1. **Opening a Terminal:**
    - On most Linux distributions, you can open a terminal from the application menu or by pressing Ctrl+Alt+T.
2. **Entering Commands:**
    - Commands are typed at the prompt, followed by pressing Enter.
    - Example: ls
3. **Using Command Options:**
    - Commands can have options (flags) that modify their behavior.
    - Example: ls -l
    - Here, -l provides a detailed listing format.
4. **Command Structure:**
    - **Command** [**options**] [**arguments**]
    - Example: cp file1.txt /home/user/ # `cp` is the command, `file1.txt` is the argument

Basic Navigation:

1. **View Current Directory:** pwd
2. **List Files:** ls
3. **Change Directory:** cd /path/to/directory
4. **Create Directory:** mkdir new_directory
5. **Delete Directory:** rmdir directory_name
6. **Delete Files:** rm file_name

Help and Documentation:

1. **Command Manual:** man command_name
2. **Command Help:** command_name --help

Tips:

- **Tab Completion:** Press Tab to auto-complete file and directory names.
- **Up/Down Arrow Keys:** Navigate through previously used commands

- **Basic Commands**: ls, cd, pwd, mkdir, rmdir

Basic Commands:

1. **ls** - List
    - Shows the files and directories in the current directory.
2. **cd** - Change Directory
    - Navigates to a different directory.
    - Example: Move to /home/user directory.
3. **pwd** - Print Working Directory
    - Displays the full path of the current directory you are in.
4. **mkdir** - Make Directory
    - Creates a new directory.
    - Example: Create a directory named new_folder.
5. **rmdir** - Remove Directory

- Deletes an empty directory.
- Example: Remove a directory named old_folder.

- **File and Directory Management**: cp, mv, rm, touch, find

File and Directory Management:

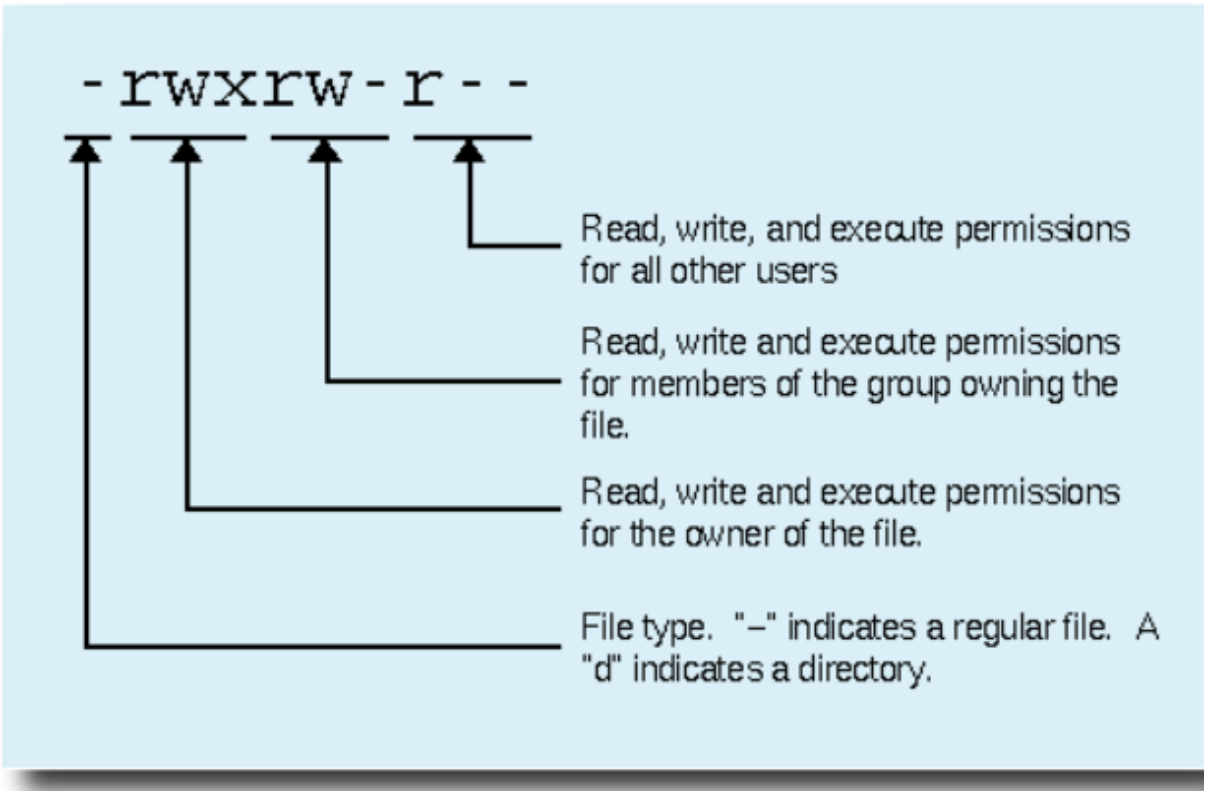
1. **cp** - Copy
    - Copies files or directories from one location to another.
    - Example: Copy a file named file.txt to /backup.
2. **mv** - Move
    - Moves or renames files or directories.
    - Example: Move file.txt to /documents, or rename old_name to new_name.
3. **rm** - Remove
    - Deletes files or directories. Use with caution, especially with the -r option to remove directories and their contents.
    - Example: Delete file.txt, or remove a directory and its contents with rm -r directory_name.
4. **touch** - Create/Update File
    - Creates a new empty file or updates the timestamp of an existing file.
    - Example: Create a file named new_file.txt.
5. **find** - Search
    - Searches for files and directories within a directory hierarchy based on various criteria.
    - Example: Find files named file.txt in the /home/user directory.

- **Viewing File Contents**: cat, more, less, head, tail

Viewing File Contents:

1. **cat** - Concatenate
    - Displays the entire contents of a file.
    - Example: Show contents of file.txt.
2. **more** - View File Page by Page
    - Displays file contents one screen at a time; useful for long files.
    - Example: View file.txt page by page.
3. **less** - View File with Scrolling
    - Similar to more, but allows backward movement in the file and has more features.
    - Example: View file.txt and scroll forward and backward.
4. **head** - View the Start of a File
    - Displays the first few lines of a file.
    - Example: Show the first 10 lines of file.txt (default behavior).
5. **tail** - View the End of a File
    - Displays the last few lines of a file.
    - Example: Show the last 10 lines of file.txt (default behavior). You can also use tail -f to follow new lines being added to a file in real-time.

3. File Permissions and Ownership



- **Understanding File Permissions**: r, w, x

File Permissions:

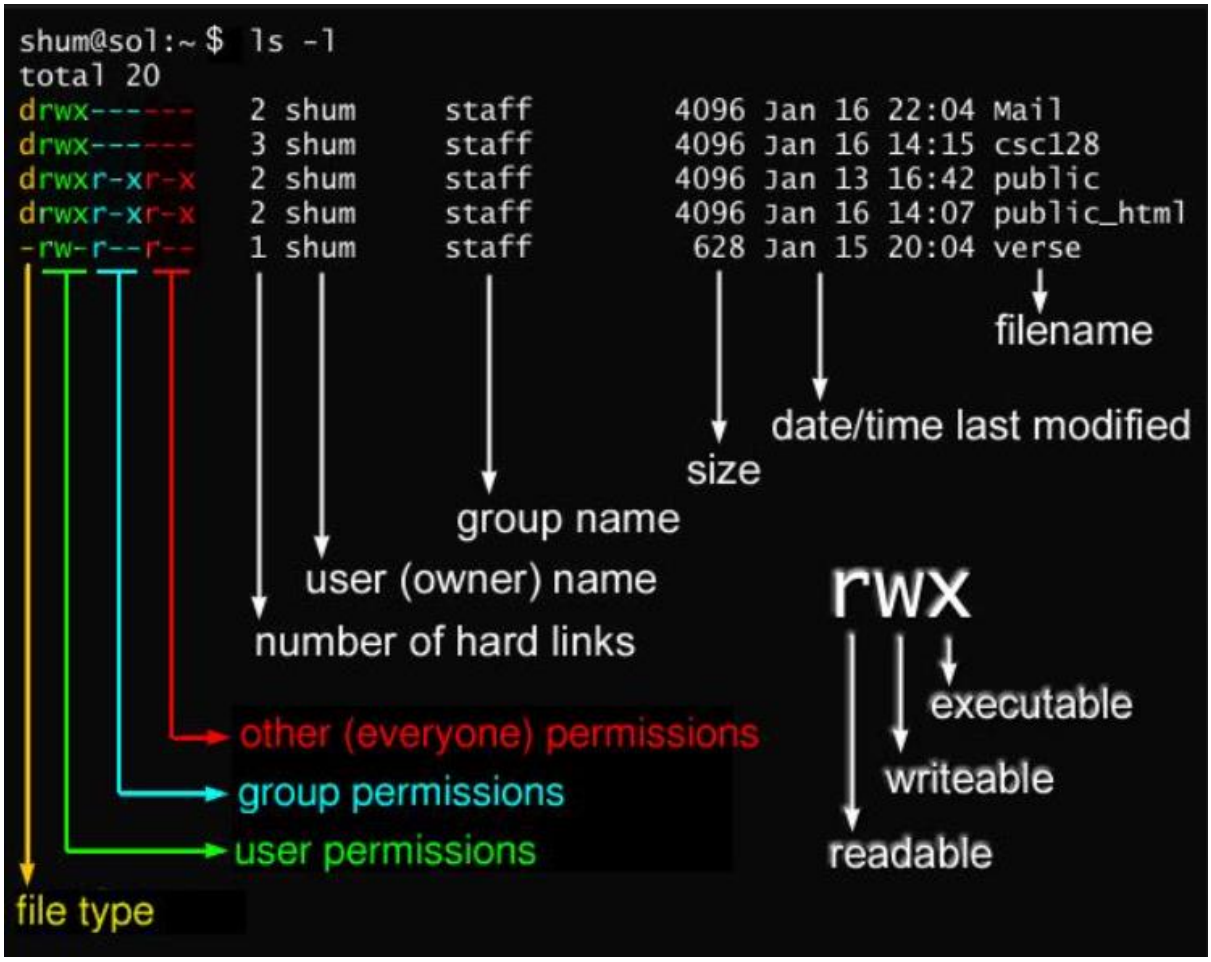
1. **r (Read)**
    - Allows viewing the contents of a file or listing the contents of a directory.
    - For files: You can open and read the file.
    - For directories: You can list files within the directory.
2. **w (Write)**
    - Allows modifying or deleting a file or directory.
    - For files: You can edit or overwrite the file's content.
    - For directories: You can add or remove files within the directory.
3. **x (Execute)**
    - Allows running a file as a program or script, or accessing a directory.
    - For files: You can execute the file if it’s a program or script.

- For directories: You can enter the directory and access files and directories within it.

Permission Representation:

- **Symbolic Notation:**
  - rwx represents permissions: Read, Write, and Execute.
  - For example, -rwxr-xr-- shows permissions for a file where:
    - - indicates a regular file (or d for directory).
    - rwx (owner permissions): Read, Write, and Execute.
    - r-x (group permissions): Read and Execute.
    - r-- (others permissions): Read only.
- **Numeric Notation:**
  - Permissions are also represented numerically: 0 (no permission), 1 (execute), 2 (write), 4 (read).
  - These numbers are added together to set permissions, e.g., 7 (read, write, execute) and 6 (read, write).

- **Changing Permissions**: chmod



Changing Permissions with chmod:

1. **Symbolic Mode:**
    - **Format:** chmod [permissions] [file/directory]
    - **Permissions:**
        - r - Read
        - w - Write
        - x - Execute
    - **Examples:**
        - chmod u+x file.txt: Adds execute permission for the file owner (user).
        - chmod g-w file.txt: Removes write permission for the group.
        - chmod o=r file.txt: Sets read-only permission for others.
2. **Numeric Mode:**

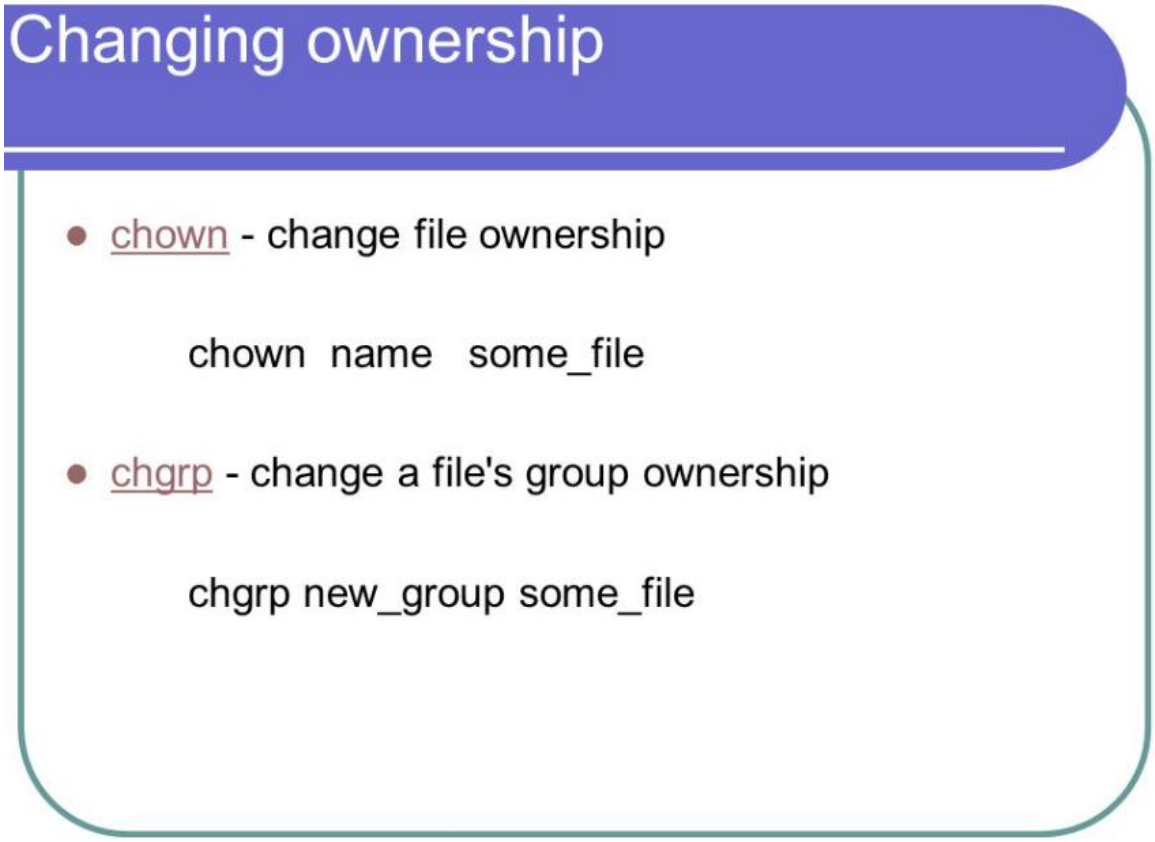
- **Format:** chmod [permissions] [file/directory]
- **Permissions are represented by three digits:**
  - **Owner** - User permissions
  - **Group** - Group permissions
  - **Others** - Permissions for everyone else
- **Digit Values:**
  - 4 - Read
  - 2 - Write
  - 1 - Execute
- **Examples:**
  - chmod 755 file.txt: Sets permissions to rwxr-xr-x (owner can read, write, and execute; group and others can read and execute).
  - chmod 644 file.txt: Sets permissions to rw-r--r-- (owner can read and write; group and others can only read).

Permission Bits Breakdown:

- **7** = rwx (Read, Write, Execute)
- **6** = rw- (Read, Write)
- **5** = r-x (Read, Execute)
- **4** = r-- (Read only)
- **3** = wx- (Write, Execute)
- **2** = w-- (Write only)
- **1** = x-- (Execute only)
- **0** = --- (No permissions)

Using chmod effectively helps manage who can access or modify files and directories on a Linux system.

- **Changing Ownership**: chown, chgrp



Changing Ownership with chown:

1. **Format:**
    - chown [owner][:group] [file/directory]
2. **Owner and Group:**
    - **Owner**: The user who will own the file.
    - **Group**: The group that will own the file (optional).
3. **Examples:**
    - chown user file.txt: Changes the owner of file.txt to user.
    - chown user:group file.txt: Changes the owner to user and the group to group.
    - chown :group file.txt: Changes only the group to group, leaving the owner unchanged.

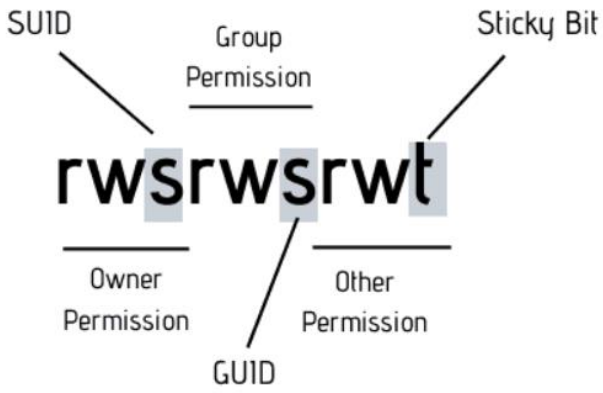
Changing Group Ownership with chgrp:

1. **Format:**
    - chgrp [group] [file/directory]
2. **Group:**
    - **Group**: The group that will own the file.
3. **Examples:**
    - chgrp group file.txt: Changes the group ownership of file.txt to group.

Usage Tips:

- Use chown to change both owner and group, or just the owner if you omit the group.
- Use chgrp to change only the group ownership of a file or directory.

- **Special Permissions**: SUID, SGID, Sticky Bit



1. **SUID (Set User ID)**
    - **Purpose:** Allows a user to execute a file with the permissions of the file owner, rather than the permissions of the user who is running the file.
    - **Set Using:** chmod u+s [file]
    - **Example:** If file is an executable with SUID set, it runs with the permissions of the file's owner, which is often used for commands like passwd that need elevated privileges.

- **Representation:** -rwsr-xr-x (The s in the owner's execute position indicates SUID is set).

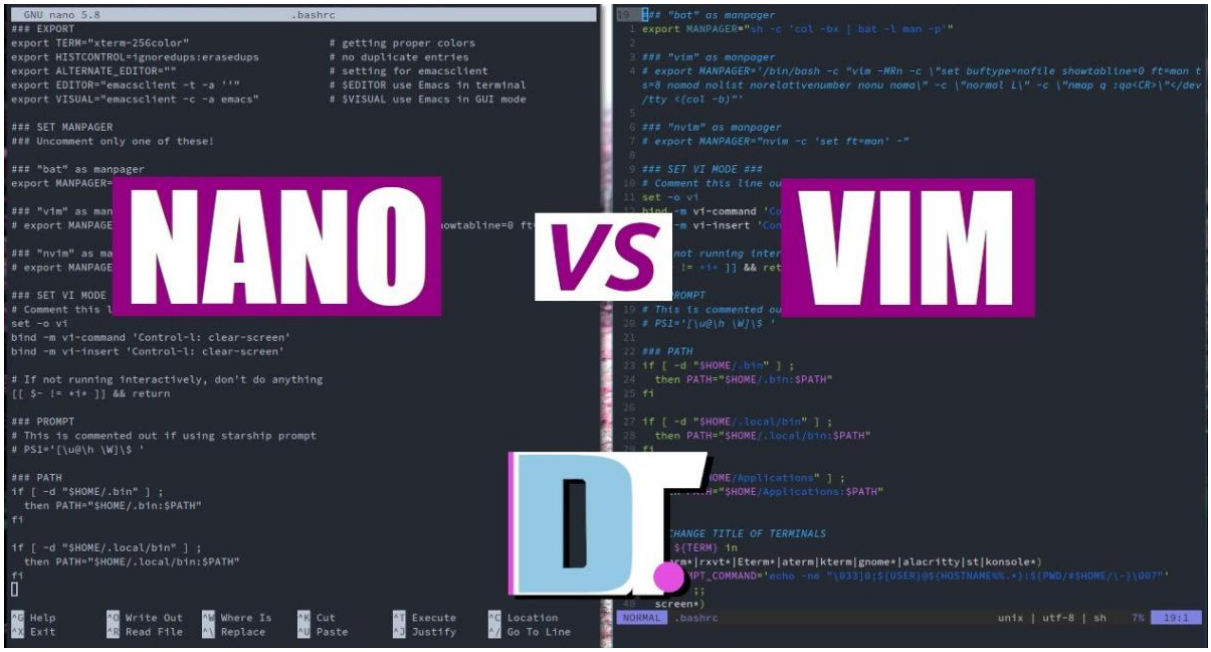
2. **SGID (Set Group ID)**
    - **Purpose:** When set on a file, it allows users to execute the file with the permissions of the file's group. When set on a directory, files created within the directory inherit the directory's group.
    - **Set Using:** chmod g+s [file/directory]
    - **Example:** Setting SGID on a directory makes files created in that directory belong to the directory's group.
    - **Representation:**
        - **File:** -rwxr-sr-x (The s in the group execute position indicates SGID is set).
        - **Directory:** drwxrwsr-x (The s in the group execute position indicates SGID is set).
3. **Sticky Bit**
    - **Purpose:** Ensures that only the file owner can delete or rename files within a directory. This is commonly used in directories like /tmp to prevent users from deleting each other's files.
    - **Set Using:** chmod +t [directory]
    - **Example:** The /tmp directory often has the Sticky Bit set to prevent users from deleting files created by other users.
    - **Representation:** drwxrwxrwt (The t in the others execute position indicates the Sticky Bit is set).

Usage Tips:

- **SUID** is commonly used for binaries that require higher privileges, but it should be used carefully due to security implications.
- **SGID** on directories is useful for collaborative environments to ensure files are created with the correct group permissions.
- **Sticky Bit** helps manage directories where multiple users can write files but should not remove each other's files.

4. Text Processing

- **Text Editors**: nano, vim, emacs



1. Nano:

- **Description:** A simple, easy-to-use text editor, ideal for beginners.
- **Opening a File:** nano filename
- **Basic Commands:**
    - **Save:** Ctrl + O, then press Enter
    - **Exit:** Ctrl + X
    - **Cut Text:** Ctrl + K
    - **Paste Text:** Ctrl + U
- **Features:** Minimalistic, straightforward interface, suitable for quick edits.

2. Vim:

- **Description:** A powerful, highly configurable text editor, preferred by advanced users.
- **Opening a File:** vim filename
- **Basic Modes:**

    - **Normal Mode:** Default mode for navigation and commands.
    - **Insert Mode:** For text input (switch with i).
    - **Command Mode:** For commands and saving (switch with :).
- **Basic Commands:**
    - **Save and Exit:** :wq or :x
    - **Exit Without Saving:** :q!
    - **Copy Text:** y (yank)
    - **Paste Text:** p
- **Features:** Extensive capabilities, support for plugins, and customizable.

3. Emacs:

- **Description:** A highly extensible, customizable text editor, often used for coding, text editing, and more.
- **Opening a File:** emacs filename
- **Basic Commands:**
    - **Save:** Ctrl + X, then Ctrl + S
    - **Exit:** Ctrl + X, then Ctrl + C
    - **Cut Text:** Ctrl + W
    - **Paste Text:** Ctrl + Y
- **Features:** Extensive functionality, including integrated development environments (IDEs), support for various programming languages, and a wide range of extensions.

Choosing an Editor:

- **Nano:** Best for beginners or quick edits.
- **Vim:** Ideal for users who prefer a more powerful editor with extensive features and keyboard shortcuts.
- **Emacs:** Great for users who want a highly customizable environment and are willing to invest time in learning its features.

- **Searching and Editing**: grep, sed, awk

1. grep: Search

- **Description:** Searches for patterns within files and outputs matching lines.
- **Basic Usage:**
    - **Search for a Pattern:** grep 'pattern' filename
    - **Case-Insensitive Search:** grep -i 'pattern' filename
    - **Search Recursively:** grep -r 'pattern' directory
    - **Show Line Numbers:** grep -n 'pattern' filename
- **Example:** Find all occurrences of error in logfile.txt. grep 'error' logfile.txt

2. sed: Stream Editor

- **Description:** Edits and transforms text in a stream or file using scripting commands.
- **Basic Usage:**
    - **Substitute Text:** sed 's/old/new/' filename (Replaces the first occurrence of old with new in each line)
    - **Substitute Globally:** sed 's/old/new/g' filename (Replaces all occurrences of old with new in each line)
    - **In-Place Editing:** sed -i 's/old/new/g' filename (Edits the file directly)
    - **Delete Lines:** sed '2d' filename (Deletes the second line)
- **Example:** Replace foo with bar in file.txt.
-

  sed 's/foo/bar/g' file.txt

3. awk: Pattern Scanning and Processing

- **Description:** Processes and analyzes text files and generates formatted reports based on patterns and fields.
- **Basic Usage:**

    - **Print Columns:** awk '{print $1, $2}' filename (Prints the first and second columns)
    - **Field Separator:** awk -F ':' '{print $1}' filename (Uses : as a field separator)
    - **Pattern Matching:** awk '/pattern/ {print $0}' filename (Prints lines matching pattern)
    - **Execute Commands:** awk '{sum += $1} END {print sum}' filename (Sums values in the first column)
- **Example:** Print the first column of a file data.txt.

awk '{print $1}' data.txt

Choosing a Tool:

- **grep:** Use for simple pattern searching and filtering.
- **sed:** Use for basic text replacement and line editing.
- **awk:** Use for complex text processing, data extraction, and reporting.

- **Sorting and Filtering**: sort, uniq, cut, paste

1. sort: Sorting

- **Description:** Sorts lines of text files or input based on specified criteria.
- **Basic Usage:**
    - **Sort File:** sort filename (Sorts lines in ascending order)
    - **Reverse Order:** sort -r filename (Sorts lines in descending order)
    - **Sort Numerically:** sort -n filename (Sorts lines numerically)
    - **Sort by Specific Column:** sort -k 2 filename (Sorts by the second column)
- **Example:** Sort data.txt in reverse order.

sort -r data.txt

2. uniq: Removing Duplicates

- **Description:** Removes duplicate lines from a sorted file or input.

- **Basic Usage:**
    - **Remove Duplicates:** uniq filename (Removes consecutive duplicate lines)
    - **Count Duplicates:** uniq -c filename (Counts occurrences of each line)
    - **Show Only Unique Lines:** uniq -u filename (Displays lines that are not duplicated)
- **Example:** Remove duplicate lines from file.txt.

uniq file.txt

3. cut: Extracting Fields

- **Description:** Extracts sections from each line of input based on delimiters or byte positions.
- **Basic Usage:**
    - **Cut by Field:** cut -f 1,3 -d ',' filename (Extracts the first and third fields, using , as the delimiter)
    - **Cut by Byte Position:** cut -b 1-5 filename (Extracts bytes 1 through 5)
    - **Cut by Character Position:** cut -c 1-5 filename (Extracts characters 1 through 5)
- **Example:** Extract the first field from data.csv with comma delimiters.

cut -f 1 -d ',' data.csv

4. paste: Merging Lines

- **Description:** Merges lines of files side by side or joins columns from multiple files.
- **Basic Usage:**
    - **Merge Files Horizontally:** paste file1 file2 (Combines lines from file1 and file2)
    - **Specify Delimiter:** paste -d ',' file1 file2 (Uses , as a delimiter between columns)

    - **Merge with Newlines:** paste -s file1 (Joins lines of file1 into a single line separated by newlines)
- **Example:** Combine file1 and file2 with a tab delimiter.

paste file1 file2

Usage Tips:

- **Combine Tools:** Often, you can combine these tools in a pipeline to perform complex data manipulations, such as sort | uniq to remove duplicates from a sorted list.

5. Process Management

- **Viewing Processes**: ps, top, htop

1. ps: Process Status

- **Description:** Displays information about active processes.
- **Basic Usage:**
    - **Show All Processes:** ps -e or ps -A (Lists all processes)
    - **Show Detailed Information:** ps -ef (Provides detailed information including user, PID, and command)
    - **Show Processes for Current User:** ps (Lists processes for the current user)
    - **Show Processes with Tree Structure:** ps -ejH (Displays processes in a hierarchical tree format)
- **Example:** Display detailed information about all processes.

ps -ef

2. top: Dynamic Process Viewer

- **Description:** Provides a real-time, dynamic view of system processes and resource usage.
- **Basic Usage:**
    - **Start top:** Just type top in the terminal to start the interactive process viewer.

    - **Sort by Resource Usage:** By default, it sorts processes by CPU usage.
    - **Interactive Commands:**
        - **q**: Quit top
        - **P**: Sort by CPU usage
        - **M**: Sort by memory usage
        - **k**: Kill a process by PID
- **Example:** View real-time process information and system resource usage.

top

3. htop: Interactive Process Viewer

- **Description:** An enhanced version of top with a more user-friendly, interactive interface and additional features.
- **Basic Usage:**
    - **Start htop:** Just type htop in the terminal to start the interactive process viewer.
    - **Features:**
        - **Color-Coded Display:** Easier visualization of processes and resource usage.
        - **Tree View:** View processes in a hierarchical tree format.
        - **Interactive Commands:**
            - **F10**: Quit htop
            - **F9**: Kill a process
            - **F5**: Toggle tree view
            - **F6**: Sort by different criteria
- **Example:** View and manage processes with a user-friendly interface.

htop

Choosing a Tool:

- **ps:** Best for quick snapshots of process information or specific queries.
- **top:** Useful for real-time monitoring of system performance and process management.
- **htop:** Offers an improved, interactive experience with more features and easier navigation.

- **Managing Processes**: kill, pkill, killall, bg, fg, jobs

1. kill: Terminate a Process by PID

- **Description:** Sends signals to processes, often used to terminate them.
- **Basic Usage:**
    - **Terminate Process:** kill PID (Sends the default SIGTERM signal to gracefully terminate the process)
    - **Force Termination:** kill -9 PID (Sends the SIGKILL signal to forcefully terminate the process)
- **Example:** Terminate a process with PID 1234.

kill 1234

2. pkill: Terminate Processes by Name

- **Description:** Kills processes based on their name or other attributes.
- **Basic Usage:**
    - **Terminate by Name:** pkill process_name (Sends the default SIGTERM signal to all processes with the specified name)
    - **Force Termination:** pkill -9 process_name (Forcibly terminates all processes with the specified name)
- **Example:** Terminate all processes named firefox.

pkill firefox

3. killall: Terminate Processes by Name

- **Description:** Similar to pkill, but specifically designed to kill processes by name.

- **Basic Usage:**
    - **Terminate by Name:** killall process_name (Sends the default SIGTERM signal to all processes with the specified name)
    - **Force Termination:** killall -9 process_name (Forcibly terminates all processes with the specified name)
- **Example:** Terminate all processes named apache2.

killall apache2

4. bg: Resume a Stopped Job in the Background

- **Description:** Resumes a job that was stopped and puts it in the background.
- **Basic Usage:**
    - **Resume in Background:** bg %job_number (Replace %job_number with the job ID)
- **Example:** Resume job number 1 in the background.

bg %1

5. fg: Resume a Background Job in the Foreground

- **Description:** Brings a background job to the foreground.
- **Basic Usage:**
    - **Resume in Foreground:** fg %job_number (Replace %job_number with the job ID)
- **Example:** Bring job number 1 to the foreground.

fg %1

6. jobs: List Background and Stopped Jobs

- **Description:** Displays a list of jobs that are running in the background or are stopped.
- **Basic Usage:**
    - **List Jobs:** jobs (Shows the current jobs with their status and job number)

- **Example:** View the list of background and stopped jobs.

jobs

Usage Tips:

- **kill** is used when you know the PID of the process.
- **pkill** and **killall** are convenient for terminating processes by name.
- **bg** and **fg** are useful for managing jobs started from the command line that you want to move between the foreground and background.

6. User and Group Management

- **Adding and Removing Users**: useradd, userdel, passwd

. useradd: Add a New User

- **Description:** Creates a new user account.
- **Basic Usage:**
    - **Add User:** useradd username
    - **Add User with Home Directory:** useradd -m username (Creates a home directory for the user)
    - **Specify User ID (UID):** useradd -u UID username (Assigns a specific UID)
    - **Specify Group:** useradd -g groupname username (Assigns the user to a specific group)
- **Example:** Create a new user john with a home directory.

useradd -m john

2. userdel: Remove a User

- **Description:** Deletes a user account.
- **Basic Usage:**
    - **Delete User:** userdel username (Removes the user account but keeps the home directory)

    - **Delete User and Home Directory:** userdel -r username (Removes the user account and their home directory)
- **Example:** Delete user john and their home directory.

userdel -r john

3. passwd: Change User Password

- **Description:** Updates or sets a user's password.
- **Basic Usage:**
    - **Change Password for Current User:** passwd (Prompts to change the password for the currently logged-in user)
    - **Change Password for Specific User:** passwd username (Allows changing the password for another user, typically requires superuser privileges)
- **Example:** Change the password for user john.

passwd john

Usage Tips:

- **useradd** creates a new user and optionally sets up their home directory and group.
- **userdel** removes user accounts, with an option to delete associated home directories.
- **passwd** manages user passwords, ensuring secure access to user accounts.

- **Group Management**: groupadd, groupdel, usermod

. groupadd: Add a New Group

- **Description:** Creates a new group.
- **Basic Usage:**
    - **Add Group:** groupadd groupname
    - **Specify Group ID (GID):** groupadd -g GID groupname (Assigns a specific GID to the group)

- **Example:** Create a new group called developers.

groupadd developers

2. groupdel: Remove a Group

- **Description:** Deletes an existing group.
- **Basic Usage:**
    - **Delete Group:** groupdel groupname (Removes the group, but not the users)
- **Example:** Delete the group developers.

groupdel developers

3. usermod: Modify User Information

- **Description:** Modifies user account details, including group memberships.
- **Basic Usage:**
    - **Add User to Group:** usermod -a -G groupname username (Adds a user to a specific group without removing them from other groups)
    - **Change Primary Group:** usermod -g groupname username (Changes the primary group for the user)
    - **Change User Information:** usermod -c "Full Name" username (Updates the user's comment or full name)
- **Example:** Add user alice to the group developers.

usermod -a -G developers alice

Usage Tips:

- **groupadd** is used to create new groups for organizing users.
- **groupdel** removes groups but does not affect the users who are members.
- **usermod** is versatile for modifying user attributes and group memberships.

- **Permissions and Privileges**

1. File Permissions:

Linux file permissions are divided into three types:

- **Read (r):** Allows viewing the content of the file or listing the contents of a directory.
- **Write (w):** Allows modifying the content of a file or creating/deleting files within a directory.
- **Execute (x):** Allows running a file as a program or script, and accessing a directory.

Permissions are set for three categories of users:

- **Owner:** The user who owns the file or directory.
- **Group:** Users who are members of the file's group.
- **Others:** All other users.

Permission Representation:

Permissions are shown in a 10-character string (e.g., -rwxr-xr--):

- The first character indicates the file type (- for regular files, d for directories).
- The next three characters represent owner permissions.
- The following three represent group permissions.
- The last three represent permissions for others.

Examples:

- -rwxr-xr--: Owner can read, write, and execute; group can read and execute; others can only read.
- drwxr-xr-x: Directory with read, write, and execute permissions for the owner, and read and execute permissions for group and others.

2. Changing Permissions:

- **chmod (Change Mode):** Used to change file or directory permissions.

    - **Numeric Mode:** chmod 755 filename (Sets permissions to rwxr-xr-x where 7 is rwx, 5 is r-x, 5 is r-x)
    - **Symbolic Mode:** chmod u+x filename (Adds execute permission for the owner)
    - **Example:** Grant execute permission to all users on script.sh.

chmod a+x script.sh

3. Changing Ownership and Group:

- **chown (Change Owner):** Changes the file or directory owner and/or group.
    - **Basic Usage:** chown owner:group filename (Changes the owner and group of the file)
    - **Example:** Change the owner to alice and group to admins for file.txt.

chown alice:admins file.txt

- **chgrp (Change Group):** Changes the group ownership of a file or directory.
    - **Basic Usage:** chgrp groupname filename (Changes the group of the file)
    - **Example:** Change the group to developers for data.txt.

chgrp developers data.txt

4. Special Permissions:

- **SUID (Set User ID):** Allows a user to execute a file with the permissions of the file's owner.
    - **Set with:** chmod u+s filename
    - **Example:** The passwd command often has SUID set to allow users to change passwords.
- **SGID (Set Group ID):**
    - **On Files:** Allows users to execute the file with the permissions of the file's group.

    - **On Directories:** New files created in the directory inherit the directory's group.
    - **Set with:** chmod g+s filename or chmod g+s directoryname
    - **Example:** Shared directories often use SGID to ensure files have the correct group ownership.
- **Sticky Bit:** Ensures that only the file owner can delete or rename their files in a directory.
    - **Set with:** chmod +t directoryname
    - **Example:** The /tmp directory uses the Sticky Bit to prevent users from deleting each other's files.

5. Managing Privileges:

- **Root User:** The superuser with unrestricted access to the system.
- **Sudo:** Allows permitted users to execute commands with root privileges.
    - **Basic Usage:** sudo command (Executes command with root privileges)
    - **Example:** Update the system with root privileges.

sudo apt-get update

Usage Tips:

- **Permissions:** Regularly check and set file permissions to ensure proper access control.
- **Ownership:** Use chown and chgrp to manage file ownership and group assignments, especially for shared resources.
- **Special Permissions:** Apply special permissions like SUID, SGID, and Sticky Bit with caution, as they can affect system security.

9. Networking

- **Network Configuration**: ifconfig, ip, netstat, ss

ifconfig: Network Interface Configuration

- **Description:** A utility to configure and display network interfaces (deprecated in favor of ip but still commonly used).
- **Basic Usage:**
    - **Show All Interfaces:** ifconfig
    - **Show Specific Interface:** ifconfig interface_name (e.g., ifconfig eth0)
    - **Assign IP Address:** ifconfig interface_name IP_address (e.g., ifconfig eth0 192.168.1.10)
    - **Bring Interface Up:** ifconfig interface_name up (e.g., ifconfig eth0 up)
    - **Bring Interface Down:** ifconfig interface_name down (e.g., ifconfig eth0 down)
- **Example:** Display network configuration for interface eth0.

ifconfig eth0

2. ip: IP Configuration

- **Description:** A modern and more versatile command for network interface and routing configuration.
- **Basic Usage:**
    - **Show All Interfaces:** ip a or ip addr
    - **Show Specific Interface:** ip a show interface_name (e.g., ip a show eth0)
    - **Assign IP Address:** ip addr add IP_address/Prefix dev interface_name (e.g., ip addr add 192.168.1.10/24 dev eth0)
    - **Bring Interface Up:** ip link set dev interface_name up (e.g., ip link set dev eth0 up)
    - **Bring Interface Down:** ip link set dev interface_name down (e.g., ip link set dev eth0 down)
    - **Show Routing Table:** ip route
- **Example:** Assign IP address 192.168.1.10 to eth0.

ip addr add 192.168.1.10/24 dev eth0

3. netstat: Network Statistics

- **Description:** Displays network connections, routing tables, interface statistics, and more (deprecated in favor of ss).
- **Basic Usage:**
    - **Show Network Connections:** netstat -tuln (Displays active TCP/UDP connections)
    - **Show Listening Ports:** netstat -tuln (Shows listening sockets)
    - **Show Routing Table:** netstat -r
    - **Show Interface Statistics:** netstat -i
- **Example:** Display all listening ports and their associated processes.

netstat -tuln

4. ss: Socket Stat

- **Description:** A modern utility to investigate sockets, often used as a replacement for netstat.
- **Basic Usage:**
    - **Show All Sockets:** ss -a
    - **Show Listening Sockets:** ss -tuln (Displays listening TCP/UDP sockets)
    - **Show Established Connections:** ss -tn (Displays active TCP connections)
    - **Show Summary:** ss -s (Provides a summary of socket usage)
- **Example:** Display all listening sockets.

ss -tuln

Usage Tips:

- **ifconfig vs ip:** While ifconfig is still used, ip is more powerful and recommended for modern Linux systems.

- **netstat vs ss:** ss provides more detailed and faster information about network sockets than netstat.

- **Network Services**: ssh, scp, ftp, wget, curl

ssh: Secure Shell

- **Description:** A protocol and command-line tool for securely accessing remote systems.
- **Basic Usage:**
    - **Connect to Remote System:** ssh username@hostname (Connects to hostname as username)
    - **Specify Port:** ssh -p port username@hostname (Connects using a specific port)
    - **Execute Command Remotely:** ssh username@hostname 'command' (Runs a command on the remote system)
- **Example:** Connect to a remote server with IP 192.168.1.10.

ssh user@192.168.1.10

2. scp: Secure Copy

- **Description:** A command for securely copying files between hosts using SSH.
- **Basic Usage:**
    - **Copy File to Remote Host:** scp localfile username@hostname:/remote/path (Copies localfile to /remote/path on hostname)
    - **Copy File from Remote Host:** scp username@hostname:/remote/path/localfile (Copies localfile from /remote/path on hostname)
    - **Copy Directory Recursively:** scp -r localdir username@hostname:/remote/path (Copies localdir and its contents)
- **Example:** Copy a file document.txt to a remote server.

scp document.txt user@192.168.1.10:/home/user/

3. ftp: File Transfer Protocol

- **Description:** A protocol and command-line tool for transferring files between hosts (less secure compared to scp and sftp).
- **Basic Usage:**
    - **Connect to FTP Server:** ftp hostname (Connects to the FTP server at hostname)
    - **Login:** Enter username and password when prompted
    - **List Files:** ls or dir
    - **Upload File:** put localfile (Uploads localfile to the FTP server)
    - **Download File:** get remotefile (Downloads remotefile from the FTP server)
- **Example:** Connect to an FTP server and list files.

ftp ftp.example.com

4. wget: Download Files

- **Description:** A command-line tool for downloading files from the web.
- **Basic Usage:**
    - **Download File:** wget URL (Downloads the file from the specified URL)
    - **Download in Background:** wget -b URL (Downloads the file in the background)
    - **Download with Resume:** wget -c URL (Resumes an interrupted download)
- **Example:** Download a file from the web.

wget http://example.com/file.zip

5. curl: Transfer Data

- **Description:** A tool for transferring data from or to a server using various protocols (including HTTP, HTTPS, FTP, and more).

- **Basic Usage:**
    - **Fetch URL Content:** curl URL (Displays the content of the URL)
    - **Download File:** curl -O URL (Downloads the file from the specified URL)
    - **Upload File:** curl -T localfile URL (Uploads localfile to the specified URL)
    - **Send POST Request:** curl -d "param=value" URL (Sends data to the URL with a POST request)
- **Example:** Download a file and save it with the same name.

curl -O http://example.com/file.zip

Usage Tips:

- **ssh** is essential for secure remote access.
- **scp** is preferred for secure file transfers.
- **ftp** is less secure and generally replaced by sftp for secure transfers.
- **wget** is great for downloading files and supports various options for handling downloads.
- **curl** is versatile for interacting with web services and APIs, with support for multiple protocols.

- **Firewalls and Security**: iptables, ufw, firewalld

iptables: Advanced Firewall Configuration

- **Description:** A powerful tool for configuring the Linux kernel's packet filtering system. It provides granular control over network traffic.
- **Basic Usage:**
    - **List Rules:** iptables -L (Lists all current rules in the default filter table)
    - **Add Rule:** iptables -A INPUT -p tcp --dport port -j ACCEPT (Adds a rule to allow incoming TCP traffic on a specified port)

    - **Delete Rule:** iptables -D INPUT -p tcp --dport port -j ACCEPT (Deletes a specific rule)
    - **Save Rules:** iptables-save > /etc/iptables/rules.v4 (Saves the current rules to a file)
    - **Restore Rules:** iptables-restore < /etc/iptables/rules.v4 (Restores rules from a file)
- **Example:** Allow incoming SSH traffic on port 22.

iptables -A INPUT -p tcp --dport 22 -j ACCEPT

2. ufw (Uncomplicated Firewall): Simplified Firewall Configuration

- **Description:** A user-friendly front-end for iptables, designed to make managing a Netfilter firewall easier.
- **Basic Usage:**
    - **Enable UFW:** ufw enable (Activates the firewall)
    - **Disable UFW:** ufw disable (Deactivates the firewall)
    - **Allow Service:** ufw allow service (Allows incoming traffic for a specific service, e.g., ssh)
    - **Deny Service:** ufw deny service (Blocks incoming traffic for a specific service)
    - **Check Status:** ufw status (Displays the current status and rules of the firewall)
- **Example:** Allow incoming HTTP traffic.

ufw allow http

3. firewalld: Dynamic Firewall Management

- **Description:** A firewall management tool that supports dynamic updates without requiring a restart. It uses zones and services for easier management.
- **Basic Usage:**
    - **Start Firewalld:** systemctl start firewalld (Starts the firewalld service)

    - **Enable Firewalld:** systemctl enable firewalld (Enables firewalld to start on boot)
    - **Check Status:** firewall-cmd --state (Shows whether firewalld is running)
    - **List Rules:** firewall-cmd --list-all (Displays current rules and settings)
    - **Allow Service:** firewall-cmd --zone=public --add-service=service (Allows a service in the specified zone, e.g., ssh)
    - **Add Port:** firewall-cmd --zone=public --add-port=port/tcp (Allows traffic on a specific port)
    - **Reload Configuration:** firewall-cmd --reload (Applies changes without restarting the service)
- **Example:** Allow HTTP and HTTPS traffic.

firewall-cmd --zone=public --add-service=http --permanent

firewall-cmd --zone=public --add-service=https --permanent

firewall-cmd --reload

Usage Tips:

- **iptables** is highly configurable but complex; use it for detailed and specific firewall rules.
- **ufw** simplifies firewall management for basic use cases and is suitable for users who prefer ease of use.
- **firewalld** provides dynamic updates and is flexible with zones and services, making it suitable for modern Linux distributions.

10. Disk Management

- **Disk Usage**: df, du

df: Disk Space Usage

- **Description:** Displays information about disk space usage for filesystems.
- **Basic Usage:**

    - **Show Disk Space Usage:** df (Displays disk space usage for all mounted filesystems)
    - **Show Disk Space Usage in Human-Readable Format:** df -h (Displays sizes in human-readable format, e.g., GB, MB)
    - **Show Disk Space Usage for Specific Filesystem:** df /path/to/filesystem (Displays disk usage for the specified filesystem)
    - **Show Inode Usage:** df -i (Displays inode usage instead of block usage)
- **Example:** Display disk space usage for all filesystems in a human-readable format.

df -h

2. du: Disk Usage of Files and Directories

- **Description:** Shows disk space usage of files and directories.
- **Basic Usage:**
    - **Show Disk Usage for Current Directory:** du (Displays disk usage for files and directories in the current directory)
    - **Show Disk Usage in Human-Readable Format:** du -h (Displays sizes in human-readable format)
    - **Show Total Disk Usage for Directory:** du -sh /path/to/directory (Shows the total size of the specified directory)
    - **Show Disk Usage for Each Subdirectory:** du -h --max-depth=1 (Displays disk usage for each subdirectory at a specified depth)
- **Example:** Display disk usage for the /home/user directory in a human-readable format.

du -sh /home/user

Usage Tips:

- **df** provides an overview of disk space usage for entire filesystems, which is useful for monitoring overall disk capacity.

- **du** is helpful for detailed analysis of space usage within specific directories, allowing you to identify large files and directories.

- **Partitioning and Formatting**: fdisk, parted, mkfs

fdisk: Partitioning Tool

- **Description:** A command-line utility for managing disk partitions on MBR (Master Boot Record) partitions.
- **Basic Usage:**
    - **List Partitions:** fdisk -l (Lists all available disks and their partitions)
    - **Open a Disk for Partitioning:** fdisk /dev/sdX (Replace X with the appropriate disk identifier, e.g., sda)
    - **Create a New Partition:** Inside fdisk, use the n command to create a new partition
    - **Delete a Partition:** Inside fdisk, use the d command to delete a partition
    - **Write Changes:** Inside fdisk, use the w command to write changes to the disk
- **Example:** Open the /dev/sda disk for partitioning.

fdisk /dev/sda

2. parted: Advanced Partitioning Tool

- **Description:** A tool for managing disk partitions, supports both MBR and GPT (GUID Partition Table) disks.
- **Basic Usage:**
    - **Start Parted:** parted /dev/sdX (Replace X with the appropriate disk identifier, e.g., sda)
    - **List Partitions:** parted /dev/sdX print (Displays the partition table)
    - **Create a New Partition:** Use mkpart inside parted, e.g., mkpart primary ext4 1GB 10GB
    - **Resize a Partition:** Use resizepart, e.g., resizepart 1 15GB

    - **Delete a Partition:** Use rm, e.g., rm 1
- **Example:** Start parted on /dev/sda and print the partition table.

parted /dev/sda print

3. mkfs: Create a Filesystem

- **Description:** A command to create filesystems on partitions.
- **Basic Usage:**
    - **Create an ext4 Filesystem:** mkfs.ext4 /dev/sdXn (Replace X with disk identifier and n with partition number, e.g., sda1)
    - **Create a XFS Filesystem:** mkfs.xfs /dev/sdXn
    - **Create a FAT32 Filesystem:** mkfs.vfat /dev/sdXn
    - **Create an ext3 Filesystem:** mkfs.ext3 /dev/sdXn
- **Example:** Create an ext4 filesystem on /dev/sda1.

mkfs.ext4 /dev/sda1