Introduction to Linux Operating System

What is it?

- Free and open-source operating system (OS) based on the Linux kernel.
- Similar to familiar systems like Windows and macOS, but with a different architecture.
- Originally developed by Linus Torvalds in 1991. Now maintained by a global community of developers.

Key Features of Linux:

- **Free & Open-Source:** Free to use and modify, with a large community for constant improvement.
- **Stable and secure:** Known for reliability and strong security features.
- **Cost-effective:** Free to use and modify, reducing software licensing costs.
- **Flexible:** Highly customizable to fit specific needs.
- **Command Line:** Powerful command line interface for efficiency.
- **Multitasking & Multi-user:** Multiple users and programs can run simultaneously.
- **Hardware Compatibility / Versatile:** Runs on a wide range of devices, from desktops to servers to supercomputers.
- **Package Management:** Easy software installation, updates, and removal with package managers.

Components:

- **Kernel:** Core of the system, manages hardware resources like memory and processors.
- **Shell:** Command-line interface (CLI) for interacting with the system (although graphical interfaces are also available).
- **Software:** Wide variety of applications and utilities available for different tasks.
- **Distributions:** Different collections of software built around the Linux kernel, each with its own focus (e.g., Ubuntu, Mint, Fedora).

Examples of Use:

- **Personal computers:** Powerful and customizable desktop environment.
- **Servers:** Runs a large portion of the world's web servers and critical infrastructure.
- **Embedded systems:** Used in devices like routers, smart TVs, and even cars.

Basic Utilities in Linux

File Manipulation:

- **ls:** Lists files and directories in the current directory.
  - Example: `ls` will show a list of files and folders in your current location.
- **cd:** Changes directory.
  - Example: `cd Desktop` will move you to your Desktop folder.
- **mkdir:** Creates a new directory.
  - Example: `mkdir Documents` will create a new folder called Documents.
- **cp:** Copies files or directories.
  - Example: `cpmyfile.txt /home/user/folder` will copy "myfile.txt" to the specified folder.
- **mv:** Moves or renames files or directories.
  - Example: `mv oldname.txt newname.txt` will rename "oldname.txt" to "newname.txt".
- **rm:** Removes files or directories (use with caution!).
  - Example: `rm emptyfile.txt` will delete "emptyfile.txt" (be sure you don't need it first!).

File Viewing and Editing:

- **cat:** Displays the contents of a file.
  - Example: `cat message.txt` will show the text in "message.txt".
- **more/less:** View files one screen at a time (useful for long files).
- **nano/vim:** Text editors for creating and modifying files. (nano is a beginner-friendly option).

System Information:

- **pwd:** Shows your current working directory (where you are in the file system).
- **uname:** Displays information about the system kernel.
- **whoami:** Tells you the name of the user you are currently logged in as.
- **hostname:** Shows the hostname of the machine.
- **df:** Displays information about disk usage.

Network Utilities:

- **ping:** Checks connectivity to another host by sending packets.
  - Example: `ping google.com` will test your connection to Google.

- **wget:** Downloads files from the internet.
  - Example: `wget https://example.com/file.txt` will download "file.txt" from the web.

Other Useful Utilities:

- **sudo:** Allows you to run commands as another user, typically the root user.
- **man:** Provides detailed information (manual pages) about other commands.
  - Example: man ls will bring up the manual page for the "ls" command.
- **clear:** Clears the terminal screen.
- **help:** Provides basic help for some built-in shell commands.

Working with Files in Linux

Linux offers various ways to interact with files. Here's a quick breakdown:

File Types:

- **Regular Files:** These are the most common files containing data like text, programs, images, etc. (Example: document.txt, photo.jpg)
- **Directories:** Folders that organize other files. (Example: /home/user/documents)
- **Special Files:** Provide access to devices like hard drives or printers. (Example: /dev/sda)
  - Block special files (for devices accessed in chunks)
  - Character special files (for devices accessed a byte at a time)
- **Symbolic Links (Symlinks):** Shortcuts pointing to other files. (Example: shortcut -> /path/to/actual/file)
- **Named Pipes (FIFOs):** Allow data exchange between processes. (Less common)
- **Socket Files:** Facilitate network communication. (For advanced users)

File Management Commands:

- **Creating Files & Directories:**
  - `touch`: Creates an empty file (Example: `touch new_file.txt`)
  - `mkdir`: Creates a new directory (Example: `mkdir new_directory`)
- **Copying & Moving Files:**
  - `cp`: Copies a file (Example: `cp file1.txt new_directory`)
  - `mv`: Moves or renames a file (Example: `mv file1.txt renamed_file.txt`)
- **Deleting Files & Directories:**
  - `rm`: Deletes files (Example: `rm file.txt`) **Note:** `rm` is permanent by default.

    - `rmdir`: Deletes empty directories (Example: `rmdir empty_directory`)
- **Finding Files:**
    - `find`: Locates files based on criteria
        - Example: `find . -name "file*.txt"` - searches for all files with names ending in ".txt" in the current directory and its subdirectories

Viewing File Contents:

- `cat`: Displays file contents (Example: `cat file.txt`)
- `less` or `more`: View contents a page at a time (Useful for large files) (Example: `less file.txt`)

File Permissions:

- Control access to files using `chmod` (change mode) and `chown` (change owner). These are more advanced topics.

Shells in Linux

What is a Shell?

- A shell is a program that acts as an interface between the user and the Linux operating system.
- Think of it as a command prompt where you type instructions for the computer to follow.

What does a Shell do?

- Interprets user-typed commands.
- Executes programs and utilities.
- Provides a way to manage files and directories.
- Allows for automation through shell scripting.

Benefits of using Shell:

- Powerful and efficient for repetitive tasks.
- More control over the system compared to a graphical interface.
- Essential for system administration and automation.

Types of Shells in Linux

1. Bourne Shell (sh):

- The original Unix shell, known for its simplicity.
- Limited features compared to modern shells.

2. C Shell (csh):

- Syntax similar to the C programming language.
- Offers features like filename completion and command history.
- Less common today but some users prefer its syntax.

3. Bourne Again Shell (bash):

- The most popular shell on Linux systems by default.
- Offers a good balance of features and ease of use.
- Supports command history, filename completion, scripting, and more.

4. Z Shell (zsh):

- Powerful shell with a lot of customization options and plugins.
- Extends bash with features like autocompletion, spelling correction, and theming.
- Popular among power users and those who prefer a more feature-rich experience.

5. Friendly Interactive Shell (fish):

- Designed to be user-friendly and easy to learn.
- Offers a clear syntax and helpful suggestions while typing commands.
- A good choice for beginners or those who prefer a simpler shell.

Shell Programming

Shell programming involves writing scripts that automate tasks in Unix-based systems (Linux, macOS) using the command-line interface (CLI).

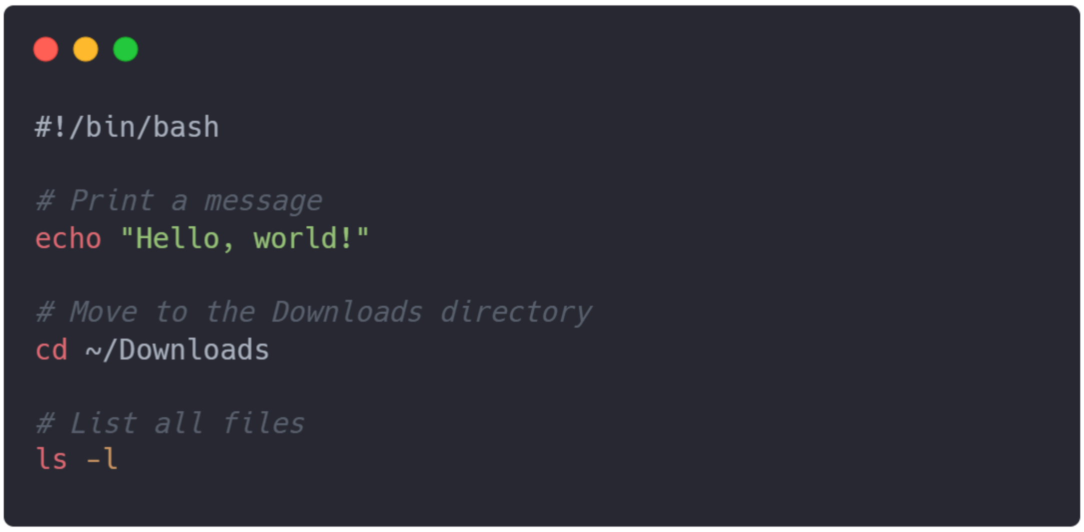
Benefits:

- Automates repetitive tasks, saving time and effort.
- Reduces errors compared to manual execution.
- Portable across Unix-like systems.

Shell Scripts:

- Programs written in a shell's scripting language.
- Contain a series of shell commands.
- Executed line by line by the shell.
- Saved as plain text files with a `.sh` extension (e.g., `myscript.sh`).

Example Script:



```bash
#!/bin/bash

# Print a message
echo "Hello, world!"

# Move to the Downloads directory
cd ~/Downloads

# List all files
ls -l
```

Text Editors in Linux

Linux offers a variety of text editors, catering to different needs and preferences. Here's a quick breakdown of some popular options:

1. Command-line editors:

- **Vim/Vi:** Powerful and highly customizable, but known for a steeper learning curve due to its modal editing style (command mode vs insert mode). Popular among programmers for its efficiency.
- **Nano:** Simpler and user-friendly alternative to Vim. Great for beginners or quick edits.

2. GUI editors:

- **gedit (GNOME):** Default editor for GNOME desktops. Offers basic functionalities like syntax highlighting, find & replace, and extensions for more features.
- **Kwrite (KDE):** Default editor for KDE desktops. Similar to gedit with a focus on KDE integration.
- **Sublime Text:** Feature-rich editor with a clean interface and powerful plugins. Free to evaluate, but requires a license for continued use.

3. Other options:

- **Emacs:** Another powerful editor with extensive customization options. Known for its loyal user base and wide range of functionalities.
- **Atom/VSCode:** Modern editors with a focus on extensibility and collaboration. Popular among web developers for their rich plugin ecosystems.

Introduction to Vim editor

- Vim stands for "Vi Improved" - a free and open-source text editor.
- Developed as an enhancement to the classic vi editor (1976).
- Known for its efficiency and powerful command-based editing.
- Offers a steeper learning curve than GUI-based editors.

Key Features of Vim

1. Modal Editing: Vim operates in different modes for specific tasks:

- **Normal Mode (default):** Use keyboard shortcuts for navigation, deletion, copying, etc.
- **Insert Mode:** Enter text like a regular editor. (Press `i` to enter Insert Mode)
- **Visual Mode:** Select text visually for editing. (Press `v` to enter Visual Mode)
- **Command-Line Mode:** Enter commands for saving, quitting, searching, etc. (Press `:` to enter Command-Line Mode)

2. Efficient Text Manipulation:

- Powerful motions to select text quickly (e.g., move by words, lines, to specific characters).
- Operators to perform actions on the selected text (e.g., delete, copy).
- Repetitive actions can be recorded and played back for automation.

3. Highly Customizable:

- Extensive configuration options through a simple text file.
- Wide variety of plugins for enhanced functionality (e.g., highlighting, code completion).

4. Lightweight and Versatile:

- Runs efficiently on any system with minimal resources.
- Available in both text-based and graphical versions (gVim).