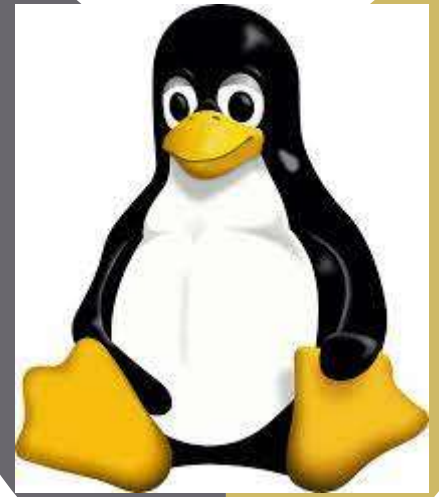
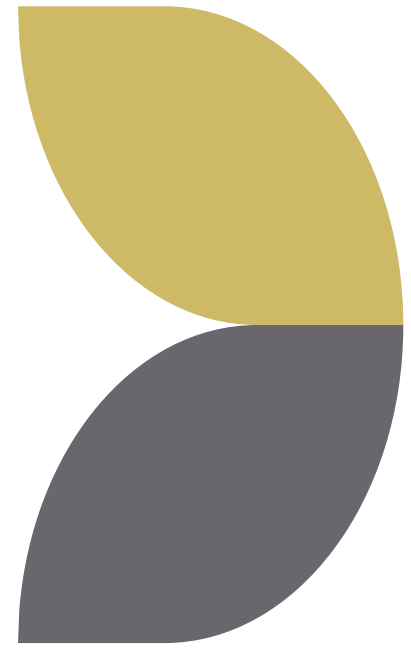


# Operating System with Linux as case study

Module 1.4



# Linux OS : Introduction



# LINUX : Introduction

- Linux is a community of open-source Unix like operating systems that are based on the **Linux Kernel**.
- It was initially released by **Linus Torvalds** on September 17, 1991.
- It is a **free and open-source** operating system and the source code can be modified and distributed to anyone commercially or noncommercially under the GNU General Public License.
- Linux is primarily written in the C programming language



- Initially, Linux was created for personal computers and gradually it was used in other machines like servers, mainframe computers, supercomputers, etc.
- Nowadays, Linux is also used in embedded systems like routers, automation controls, televisions, digital video recorders, video game consoles, smartwatches, etc.
- The biggest success of Linux is Android(operating system) it is based on the Linux kernel that is running on smartphones and tablets.
- Due to android Linux has the largest installed base of all general-purpose operating systems. Linux is generally packaged in a Linux distribution.

- **LINUX** - Lovable Intellect Not Using XP



# Linux Distributions



- A Linux distribution (often referred to as a distro) is a complete operating system package that includes the Linux kernel along with a collection of software applications, libraries, and tools.
- These distributions are built on top of the Linux kernel and designed to provide a user-friendly and coherent computing experience for users.
- The Linux kernel serves as the core of the operating system, responsible for managing hardware resources, providing system services, and facilitating communication between software and hardware components.
- However, the kernel alone is not sufficient for a fully functional operating system. That's where Linux distributions come in.

# Linux Distribution:Components

- **Linux Kernel:** The core component that manages system resources and provides essential services.
- **System Libraries:** Collections of pre-compiled software code that applications use to interact with the operating system.
- **Software Packages:** A vast selection of applications and utilities, such as web browsers, office suites, multimedia players, text editors, development tools, and more.
- **Package Management System:** A tool or set of tools that allow users to easily install, update, and remove software packages from the system.



- **Desktop Environment or Window Manager:** The graphical user interface that users interact with, providing icons, windows, menus, and other graphical elements.

There are many desktop environments to choose from: (GNOME, Cinnamon, Mate, Pantheon, Enlightenment, KDE, Xfce, etc.). Each desktop environment includes built-in applications (such as file managers, configuration tools, web browsers, and games).

**GNOME - GNU Object Model Environment**

**GNU - GNU's Not UNIX**

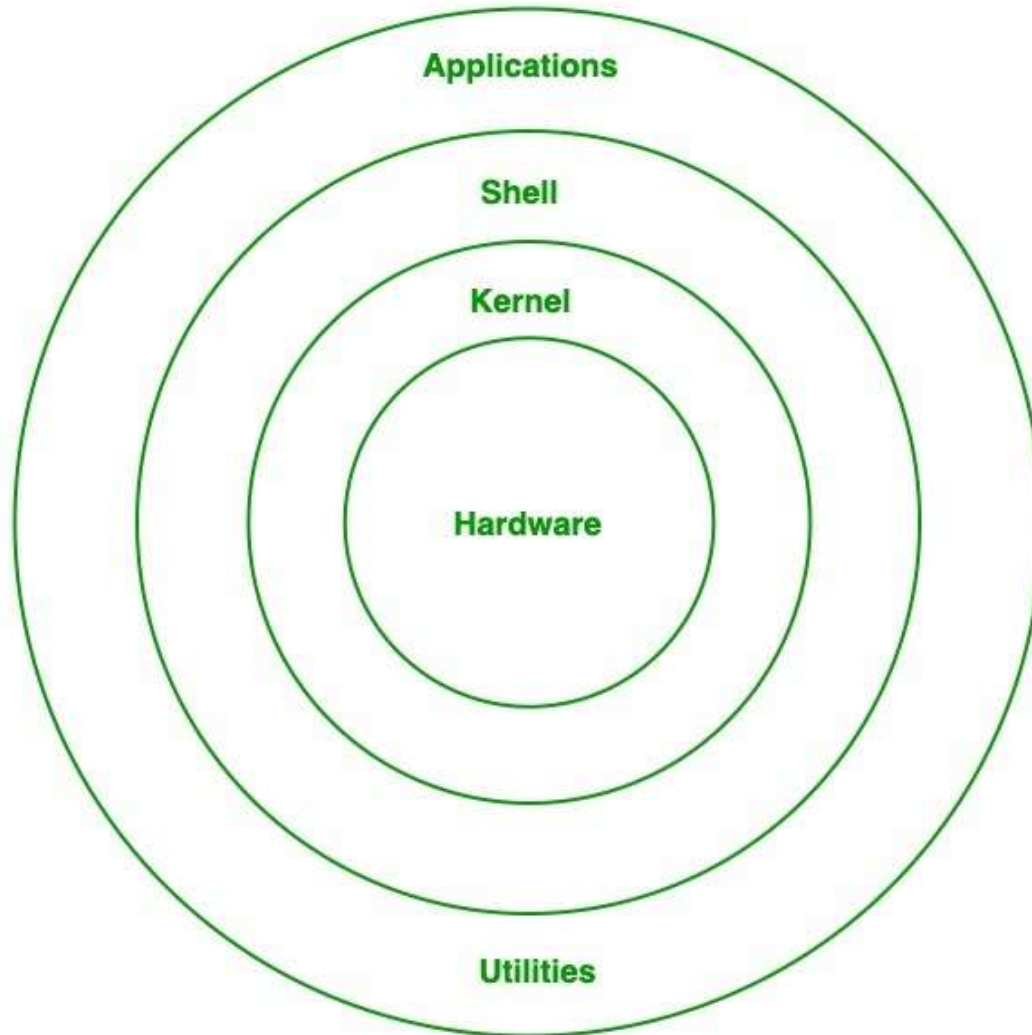


# Linux Distributions : Examples

- There are many different Linux distributions available, each tailored to various use cases and user preferences.
- Some of the most well-known Linux distributions include:
  - ★ Ubuntu
  - ★ Fedora
  - ★ Debian
  - ★ CentOS
  - ★ Arch Linux
  - ★ openSUSE
  - ★ Red Hat
- ★ Each distribution may have its own unique characteristics, package management system, and default desktop environment, but they all share the common foundation of the Linux kernel.
- ★ Users can choose the distribution that best suits their needs and install it on their computers to enjoy the benefits of Linux and open-source software.



# Linux : Architecture



# Linux Kernel

- The Linux kernel is the main component of a **Linux operating system (OS)** and is the core interface between a computer's hardware and its processes. It communicates between the two, managing resources as efficiently as possible.
- The kernel is so named because—like a seed inside a hard shell—it exists within the OS and controls all the major functions of the hardware, whether it's a phone, laptop, server, or any other kind of computer.



# Kernel : Functions

The kernel has 4 jobs:

- **Memory management:** Keep track of how much memory is used to store what, and where
- **Process management:** Determine which processes can use the central processing unit (CPU), when, and for how long
- **Device drivers:** Act as mediator/interpreter between the hardware and processes
- **System calls and security:** Receive requests for service from the processes



# Linux Shell

- The shell can be defined as a command interpreter within an operating system like Linux or Unix.
- It is a program that runs other programs. The shell facilitates every user of the computer as an interface to the Unix/GNU Linux system.
- Hence, the user can execute different tools/utilities or commands with a few input data.
- The shell sends the result to the user over the screen when it has completed running a program which is the common output device. That's why it is known as "command interpreter".
- The shell is not just a command interpreter. Also, the shell is a programming language with complete constructs of a programming language such as functions, variables, loops, conditional execution, and many others.

# Linux Shell

- Linux distributions typically use the Bash (Bourne Again SHell) as the default shell for the command-line interface (CLI).
- Bash is the most widely used and well-known shell in the Unix-like world, and it provides a rich set of features for interactive use and shell scripting.



# Linux : Advantages

- The main advantage of Linux, is it is an open-source operating system. This means the source code is easily available for everyone and you are allowed to contribute, modify and distribute the code to anyone without any permissions.
- In terms of security, Linux is more secure than any other operating system. It does not mean that Linux is 100 percent secure it has some malware for it but is less vulnerable than any other operating system. So, it does not require any anti-virus software.
- The software updates in Linux are easy and frequent.
- Various Linux distributions are available so that you can use them according to your requirements or according to your taste.
- Linux is freely available to use on the internet.
- It has large community support.
- It provides high stability. It rarely slows down or freezes and there is no need to reboot it after a short time.

# Linux : Disadvantages

- It is not very user-friendly. So, it may be confusing for beginners.
- It has small peripheral hardware drivers as compared to windows.



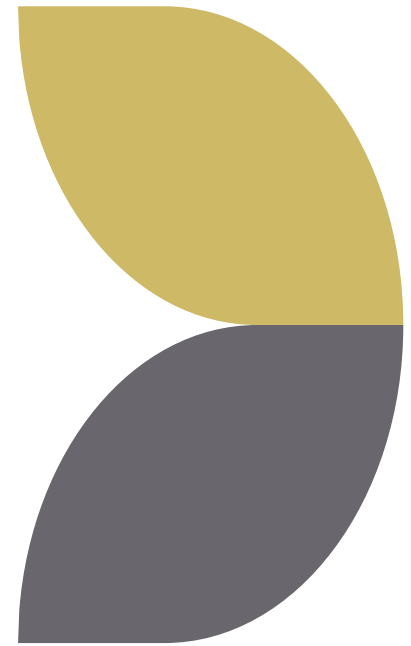
# Linux : Disadvantages

- It is not very user-friendly. So, it may be confusing for beginners.
- It has small peripheral hardware drivers as compared to windows.



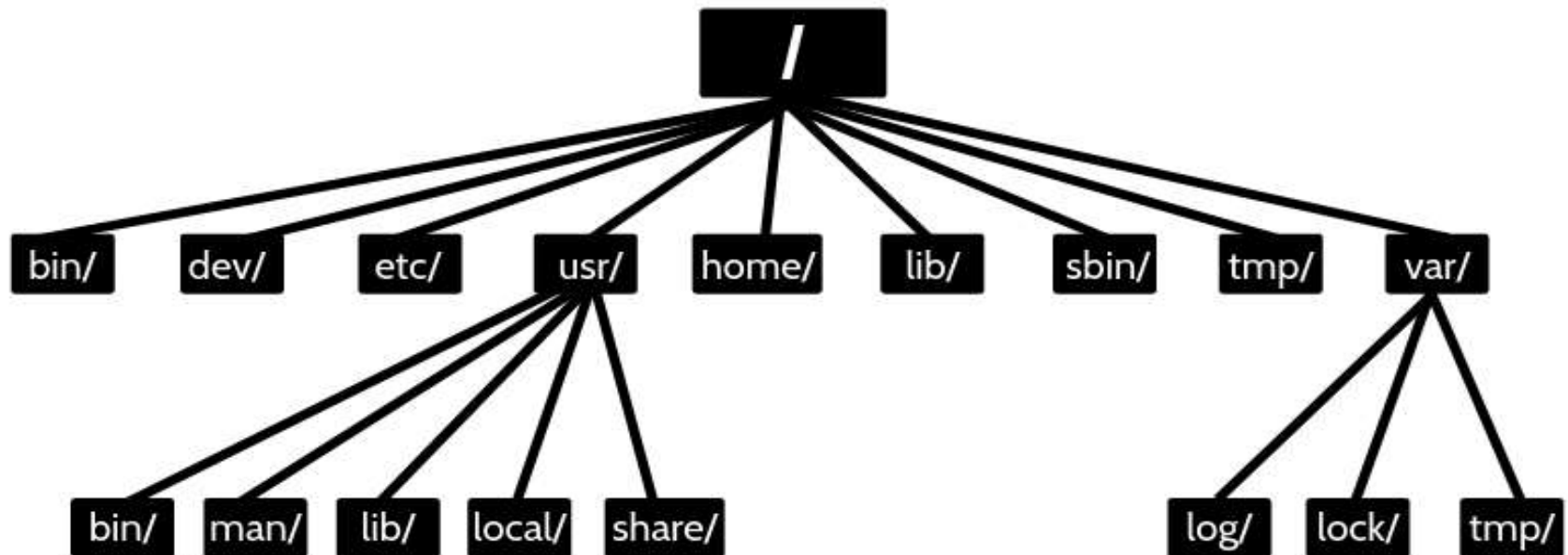


# Linux : File System



# File System Structure

- The Linux file system is organized in a **hierarchical structure**, starting from the root directory (“/”) and branching out into different directories.
- Each directory serves a specific purpose, making it easier to organize and locate files and resources.



# Linux file system and its key components

Linux supports various file systems, but the most commonly used file system is the **Extended File System** (ext) family, specifically ext2, ext3, ext4.

**File:** A file is a collection of data, such as text, programs, or multimedia content, which is stored on a storage device and accessed by its name.

**Directory:** A directory (also known as a folder) is a special type of file that can contain other files and directories. It provides a hierarchical organization of files and directories, allowing users to structure data systematically.


**Path:** A path is the location of a file or directory in the file system hierarchy. There are two types of paths: absolute and relative. An absolute path starts from the root directory (/), while a relative path starts from the current working directory.

**Root Directory:** The root directory (/) is the top-level directory in the file system hierarchy. All other directories and files are organized beneath it.

**Mount Point:** Linux allows you to attach different storage devices (like hard drives or USB drives) to the file system by mounting them to specific directories. The directories where these devices are attached are known as mount points.

**File Permissions:** Linux employs a robust file permission system that defines which users or groups can read, write, or execute a file or directory. The permissions are represented by three groups: owner, group, and others.


**File System Types:** Linux supports various file system types, including ext2, ext3, ext4 (the extended file system family), Btrfs, XFS, JFS, and more. Each file system type has its own features, performance characteristics, and suitability for different use cases.



**Inodes:** Inodes are data structures in the file system that store metadata about files and directories, such as permissions, ownership, size, timestamps, and pointers to data blocks.

**Virtual File Systems:** Linux uses virtual file systems to provide a unified view of various file systems and device types. For example, /proc and /sys are virtual file systems that expose information about the running processes and kernel settings.

**File System Utilities:** Linux provides a set of utilities to manage file systems, including mkfs (to create a file system), mount (to attach storage devices to the file system), umount (to unmount devices), ls (to list files and directories), cp (to copy files), rm (to remove files), and many others.



# Key Directories in the Linux file structure

- Root Directory (/):** The root directory is the top-level directory in the file system hierarchy. It serves as the starting point for navigating the entire file system. All other directories and files are organized beneath the root directory.
- /bin:** This directory contains essential binary executables (commands) that are required for basic system operation. Common commands like `ls`, `cp`, `mv`, `rm`, and others are stored here.
- /boot:** The `/boot` directory contains files related to the boot process, such as the Linux kernel, initial RAM disk (`initramfs`), and boot loader configuration files.
- /dev:** This directory contains device files that represent and provide access to various hardware devices connected to the system, like hard drives, USB devices, and more.

# Key Directories in the Linux file structure

**/etc:** Configuration files for system-wide settings and services are stored in the /etc directory. You'll find files for network configuration, user accounts, software repositories, and more.

**/home:** Each user on the system typically has a home directory located within the /home directory. User-specific files and settings are stored here.

**/lib and /lib64:** These directories contain shared library files used by the system and various applications.

**/media:** When removable media (e.g., USB drives) are mounted, their mount points are usually created under the /media directory.

**/mnt:** The /mnt directory is used as a temporary mount point for other file systems.



# Key Directories in the Linux file structure

**/opt:** Optional software packages can be installed in the /opt directory. These packages are usually self-contained and not part of the core system.

**/proc:** The /proc directory is a virtual file system that exposes information about the running processes and system configuration. It provides access to kernel and process-related data.

**/root:** This is the home directory of the root user, which is the administrative user on the system.

**/run:** The /run directory contains runtime data for system services. It is often used for communication and coordination between processes during the current system session.



# Key Directories in the Linux file structure

**/sbin:** This directory holds system binaries (commands) that are used for system administration tasks and typically require superuser (root) privileges.

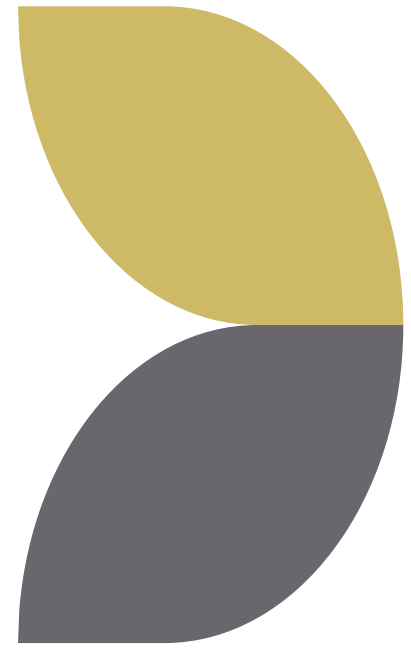
**/srv:** The /srv directory is used to store data for services provided by the system.

**/tmp:** The /tmp directory is intended for temporary files that are usually cleared upon system reboot.

**/usr:** This directory contains user-related data and software applications. It includes subdirectories such as /usr/bin, /usr/lib, /usr/share, and more.

**/var:** The /var directory holds variable data that changes during the system's operation, such as log files, spool files, and cached data.

# Shell Scripting



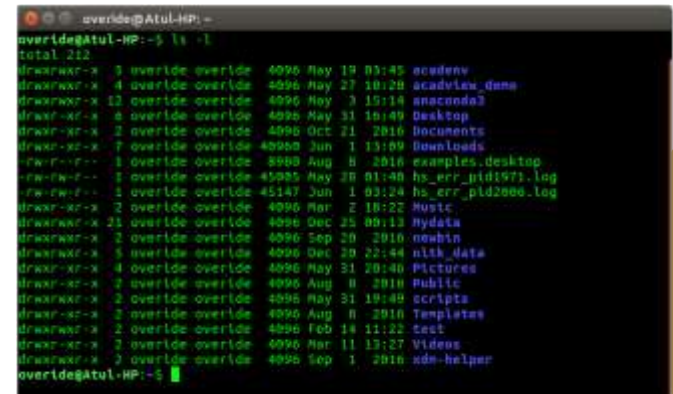
# Shell : Classification

Shell is broadly classified into two categories –

- Command Line Shell
- Graphical shell

## Command Line Shell

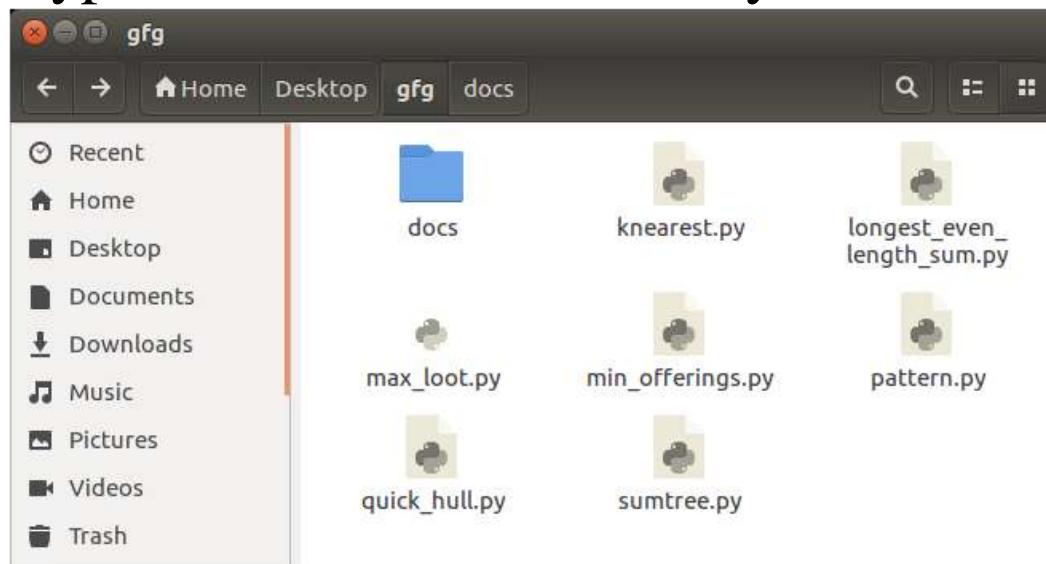
Shell can be accessed by users using a command line interface. A special program called Terminal in Linux/macOS, or Command Prompt in Windows OS is provided to type in the human-readable commands such as “cat”, “ls” etc. and then it is being executed. The result is then displayed on the terminal to the user. A terminal in Ubuntu 16.4 system looks like this –



```
override@Atul-HP: ~$ ls -l
total 212
drwxrwxr-x 3 override override 4096 May 19 03:45 academy
drwxrwxr-x 4 override override 4096 May 27 10:28 acadview_data
drwxrwxr-x 12 override override 4096 May 3 15:14 anaconda3
drwxrwxr-x 6 override override 4096 May 31 16:49 Desktop
drwxrwxr-x 2 override override 4096 Oct 21 2016 Documents
drwxrwxr-x 7 override override 4096 Jun 1 13:09 Downloads
-rw-r--r-- 1 override override 8960 Aug 8 2016 examples_desktop
-rw-r--r-- 1 override override 45885 May 28 01:48 hs_err_pid1971.log
-rw-r--r-- 1 override override 45147 Jun 1 03:24 hs_err_pid2060.log
drwxrwxr-x 2 override override 4096 Mar 2 18:22 Rustic
drwxrwxr-x 21 override override 4096 Dec 25 00:13 Mydata
drwxrwxr-x 2 override override 4096 Sep 20 2016 sshin
drwxrwxr-x 5 override override 4096 Dec 20 22:44 ntk_data
drwxrwxr-x 4 override override 4096 May 31 20:40 Pictures
drwxrwxr-x 2 override override 4096 Aug 8 2016 Public
drwxrwxr-x 2 override override 4096 May 31 19:49 scripts
drwxrwxr-x 2 override override 4096 Aug 8 2016 Templates
drwxrwxr-x 2 override override 4096 Feb 14 11:22 Test
drwxrwxr-x 2 override override 4096 Mar 11 13:27 Videos
drwxrwxr-x 2 override override 4096 Sep 1 2016 xde-helper
override@Atul-HP: ~$
```

# Graphical Shells

Graphical shells provide means for manipulating programs based on the graphical user interface (GUI), by allowing for operations such as opening, closing, moving, and resizing windows, as well as switching focus between windows. Window OS or Ubuntu OS can be considered as a good example which provides GUI to the user for interacting with the program. Users do not need to type in commands for every action. A typical GUI in the Ubuntu system –



# Shells Types

**Sh (Bourne Shell):** The Bourne shell (sh) is the original Unix shell, and it provides a basic set of features. While it's still available on most systems, it's less commonly used interactively due to its limited capabilities compared to more modern shells.


**Bash (Bourne Again SHell):** Bash is one of the most widely used and default shells in Linux. It's a successor to the original Bourne shell (sh) and provides a rich set of features for interactive use and scripting.

**Zsh (Z Shell):** Zsh is another popular shell that is highly customizable and user-friendly. It offers advanced features like improved tab completion, spelling correction, and powerful globbing capabilities. Many users prefer Zsh due to its extensive configuration options.

**Fish (Friendly Interactive SHell):** Fish is designed to be beginner-friendly with a user-friendly syntax and helpful auto-completion that displays suggestions as you type. It's easy to use and requires minimal configuration.



**Ksh (Korn SHell):** The Korn shell is another shell that's compatible with the original Bourne shell (sh) but includes enhancements like improved scripting capabilities and additional features for interactive use.

**Dash (Debian Almquist SHell):** Dash is a lightweight and efficient shell, optimized for scripting purposes. It's often used as the default system shell in some Linux distributions due to its minimal memory footprint.



**Csh (C SHell):** The C shell is known for its C-like syntax, and it provides some unique features, but it's less popular these days compared to other shells like Bash and Zsh.

**Tcsh (TENEX C SHell):** Tcsh is an enhanced version of the C shell, offering additional features like command-line editing and history substitution. It's compatible with Csh but includes some improvements.



# Shell Scripting

- Shell scripting refers to the process of writing scripts using a shell language (like Bash, Zsh, or others) to automate tasks and perform various operations in a Unix-like operating system, such as Linux.
- Shell scripts are essentially collections of shell commands and constructs organized into a script file that can be executed as a single unit.
- Shell scripts can be simple or complex, depending on the tasks they need to accomplish.
- They are widely used by system administrators and developers to automate repetitive tasks, manage system configurations, and streamline workflows.




**Here are some key concepts and features related to shell scripting:**

**Script File:** A shell script is a text file that contains a series of shell commands and instructions. The file typically has a .sh extension, but it's not required. The first line of the script, known as the "shebang," indicates which shell should be used to execute the script.

**Shebang:** The shebang line specifies the path to the shell executable that should be used to run the script. For example, `#!/bin/bash` indicates that the Bash shell should be used. It is essential to include this line at the beginning of the script.

**Variables:** Shell scripts use variables to store and manipulate data. Variables are denoted by a dollar sign followed by the variable name (e.g., `$variable_name`). You can assign values to variables and use them in commands.





**Control Structures:** Shell scripts support various control structures, such as loops (for, while) and conditional statements (if, else, elif), allowing you to control the flow of execution based on conditions.

**Command-Line Arguments:** Shell scripts can accept command-line arguments passed when executing the script. These arguments can be accessed inside the script using special variables like \$1, \$2, etc.

**Functions:** Shell scripts can define functions, allowing you to group sets of commands into reusable blocks of code.


**Input/Output:** Shell scripts can read input from users, files, or other commands, and they can display output to the screen or redirect it to files.





**Exit Codes:** After executing a script, it returns an exit code (a numerical value) that indicates whether it completed successfully or encountered an error. By convention, an exit code of 0 signifies success, while non-zero values indicate errors.

**Comments:** Shell scripts can include comments using the # symbol. Comments are used to add explanations or notes to the script, and they are ignored during execution.



# Vi Editor



- The vi editor is elaborated as visual editor.
- It is installed in every Unix system. In other words, it is available in all Linux distros.
- It is user-friendly and works same on different distros and platforms. It is a very powerful application.
- An improved version of vi editor is vim.



# Linux Commands

- Directory Oriented Commands
- File Oriented Commands
- File access permission
- General purpose Commands
- Process Oriented Commands
- Pipes & Filters



# **Directory Oriented Commands**



- **ls** - lists the contents of the specified directory

***ls [-options] <directory\_name>***

ls -l               # Detailed list

ls -a               # List all files, including hidden ones

ls -h               # Human-readable sizes

- **cd** – change current working directory to a specified directory

***cd <directory\_name1>***

cd /path/to/directory   # Change to the specified directory

cd ..               # Change to the parent directory

cd                   # Change to the user's home directory

cd -                # Change to the previous directory

- **pwd** - Print the working directory. It displays the current directory's full path.

- **mkdir** – make new directories

***mkdir [-p] <directory\_name1> <directory\_name2>***

*mkdir directory\_name # Create directory*

*mkdir -p path/to/nested/directory # Create nested directories*

- **rmdir** – remove specified directories. Remove an empty directory.  
Note that it can only remove directories that do not contain any files or subdirectories.

***rmdir [-p] <directory\_name1> <directory\_name2>***

*rmdir directory\_name*

- **rm** - Remove files or directories. Use with caution as it is a powerful command that permanently deletes files and directories.

*rm filename*

*rm -r directory\_name # Remove a directory and its contents recursively*



- **cp** - Copy files and directories from one location to another.

***cp [-options] <src\_filename> <dest\_filename>***

*cp file.txt /path/to/destination*

*cp -r directory /path/to/destination # Copy a directory and its contents recursively*

- **mv** - Move or rename files and directories. It can also be used to move files/directories from one location to another.

***mv <src> <dest>***

*mv file.txt newname.txt*


*mv directory /path/to/destination # Move a directory to a different location*

- **find**: Search for files and directories based on various criteria, such as name, size, or modification time.

***Find <path\_list> <selection\_criteria> <action>***

*find /path/to/start/directory -name "filename" # Find files by name*

*find /path/to/start/directory -type d # Find directories*



**tree** - Display the directory structure as a tree. It is not installed by default on most systems, but you can usually install it using package managers like apt, yum, or brew.

***tree /path/to/directory***





# **File Oriented Commands**



- **touch**: Create an empty file or update the modification timestamp of an existing file.  
***touch file\_name***
- **cat**: display contents of specified file, used with redirection operator to create new files. Concatenate and display the content of one or more files.

***cat [-options] <filename1> [<filename2>]***

***cat filename***

***#To view a single file***

***cat file1 file2 file3***

***# Display the content of multiple files***

***cat >filename***

***#Create a file and add content***

***cat file1 > file2***

***# Redirect Contents of a Single***

**File**

***cat file1 file2 > file3***

***# Redirect Contents of Multiple Files***

***tac filename***

***# Display the Contents in***

**wc** – display no. of lines ,words, char stored on the spec. file

**wc** *[-options]* <filename>

**grep** - Search for text within files.

grep "search\_text" file\_name

**more** - Display file contents one screen at a time.

more file\_name

**less** - Similar to more, but allows scrolling both forward and backward.

less file\_name

**head** - Display the first few lines of a file

head -n 10 file\_name (displays the first 10 lines)

**tail** - Display the last few lines of a file

tail -n 10 file\_name (displays the last 10 lines)

**ln** - Create links (hard or symbolic) to files or directories.

ln -s source\_file symbolic\_link



# **File Access oriented Commands**



**chmod** - Change file permissions.

***chmod [options] [mode] [File\_name]***

The chmod command can be used with both **symbolic and numeric notation**.

## **Nu...ric Notation:**

In numeric notation, each permission is represented by a number: Read (4), Write (2), Execute (1). The sum of these numbers represents the desired permission setting.

## **Example :**

- chmod 755 file.txt sets read, write, and execute permissions for the owner, and read and execute permissions for group and others.
- chmod 644 file.txt sets read and write permissions for the owner, and read-only permissions for group and others.

## Symbolic Notation:

In symbolic notation, permissions are represented using letters: Read (r), Write (w), Execute (x). Additionally, you can use symbols like + to add permissions and - to remove permissions.

### Example:

```
chmod u=rw,g=r,o=r file.txt
```

- gives read and write permissions to the owner, read permission to the group and others.
- `chmod a+x script.sh` adds execute permission for all (owner, group, and others) to a script.

### Here are some common symbolic notation examples:

- u: User (owner)
- g: Group
- o: Others (everyone else)
- a: All (equivalent to ugo)





## Modifiers:

+: Add permission

-: Remove permission

=: Set permission exactly as specified

## Examples:

`chmod 755 file.txt`      # Numeric notation

`chmod u=rw,g=rx,o=r file.txt` # Symbolic notation

`chmod a+x script.sh`    # Add execute permission for all

`chmod go-w secret.txt` # Remove write permission for group and others

`chmod ugo+rw file.txt` # Give full permissions to owner, group, and others

`chmod 600 private_file` # Restrict file to read and write only for the owner

**chown** - Change file ownership.

***chown new\_owner:new\_group file\_name***

chown john:users file.txt (Changes the owner to user "john" and the group to "users")

**chgrp**: Change file group ownership.

***chgrp new\_group file\_name***

chgrp developers file.txt (Changes the group ownership to "developers")

**id**: Display user and group information.

id

**umask**: Set the default permission mask for new files and directories.

- umask new\_mask
- Example: umask 022 (Sets the default mask to allow read and write permissions for the owner and read permissions for group and others.)



# **General Purpose Commands**



**date** – displays system date & time

date +<format>

**who** –display users currently logged on to sthe system

**who am i**- tells you who you are.

**man** –displays syntax and detailed usage of the Linux command

man <linux command>

**cal** – display calendar for the month and year specified

cal [<month>] <year>

**lpr** – print one or more files on a printer

lpr [-options] <filename> <filename2> ...

**tee** -send output of a command into standard output as well to spec. file.

command | tee <filename>

Eg. who | tee userlist.txt | wc -l

