1.1	Features of Linux OS
1.2	Components of Linux OS (Hardware, Kernel, Shell, GNU Utilities & Applications)
1.3	Shell in Linux (Bash, Zsh, Dash – Features and Differences)
1.4	Introduction to Files and File Types in Linux (text, binary, special files)
1.5	Linux Directory Structure and File System Hierarchy Standard (FHS)

Introduction

- Linux is one of the oldest operating systems, which is a **multi-user** and **multi-tasking** system.
- The Linux OS is basically written in C programming language.
- Under Linux, the operating system consists of many utilities along with the master control program, called as the ***kernel***.
- The kernel provides services to start and stop programs, handles the file system and other common "low-level" tasks that most programs share, and schedules access to avoid conflicts when programs try to access the same resource or device simultaneously.
- To mediate such access, the kernel has special rights, reflected in the division between user space and kernel space.
- The user interacts with a Linux system through a *command interpreter* known as the ***shell***.

1.1 Features of Linux OS

I. Multi-user capability

- Multi-user feature allows users to use the resources of main computer.
- In other words, at a time same computer resources, such as hard disk, memory, printer etc..., are available to many users.
- In this environment, different terminals are given to users who want to access resources of main computer.
- Here, a terminal consists of input device and the output device i.e. keyboard and monitor respectively.
- All the terminals are connected to the main computer known as host machine.
- The resources of host machine are available to all users.
- Therefore, any user from any of the terminals can use the computer as well as any peripherals attached to main computer e.g. printer, scanner etc.

II. Multitasking capability

- Another main feature of Linux operating system is multitasking.
- This feature enables user to carrying out more than one job at the same time i.e. in Linux, a single user can also run multiple task concurrently.
- In multitasking environment, user can start a job, such as print a file on the printer and then move to another job, such as writing an e-mail, without leaving any of the applications.
- In such a multitasking environment, only one job runs in the foreground while the other jobs run in the background.
- You can switch jobs between background and foreground, suspend, or even terminate them.

III. Inter process communication

- This feature allows user to communicate with users. This feature allows user to pass on data, exchange mail, or program to another users within the networks.

IV. Security

- LINUX provides three types of securities, (i) System level (ii) directory level and (iii) file level.

- **System Level security**: In Linux, every user has been allocated login id and password. When the system administrator opens an account for user, an entry is created in a system password file, called /etc/passwd. So to access the resources of the LINUX system, user has to log in first. Only the authorized user can access the system.

- **Directory level security**: In LINUX, everything is treated as file. Even directories or devices are also considered as file. There are read, write, and execute permissions to each file, which decide who can access a particular file, who can modify it and who can execute it
- **File level security:** Lastly, there is file encryption. Encryption utility encodes your file into an unreadable format, so that even if someone succeeds in opening it, your secret information is safe.

V. Portability

- LINUX system is written in a high level language i.e. in C.
- Moreover, C programs are easily moved from one hardware environment to another.
- This feature is also inherited in LINUX which makes it easier to read, understand, change and move to other machines.
- The code can be changed and compiled on a new machine.

VI. Open system

- LINUX has an open architecture; one can add to the tool kit by simply writing a program and storing the executable in a separate area in the file system.
- A separate device can also be added by creating a file for it.
- Modification of the system is easy because the source code is always available.

VII. Windowing System

- Initially, LINUX has a pretty weak user interface in that it was command-driven.
- Faced with increasing competition from Microsoft windows, it was obvious that sooner or later LINUX had to come up with its own GUI
- The window system provides an environment that allows you to have multiple windows where you can run applications on each window individually.
- It replaces the command line with a screen full of objects in the form of menus, icons, buttons and dialogue boxes.

VIII. System calls and libraries

- LINUX is written in C language.
- There are many commands available in LINUX that handles specialized functions called system calls.
- These calls are built into the kernel, and all library functions and utilities are written using them.

IX. Programming facility

- LINUX is highly programmable; it was designed for programmer, not a casual end user.
- The LINUX shell programming language has all the necessary ingredients like control structures, loops and variables that establish it as a programming language in its own right.

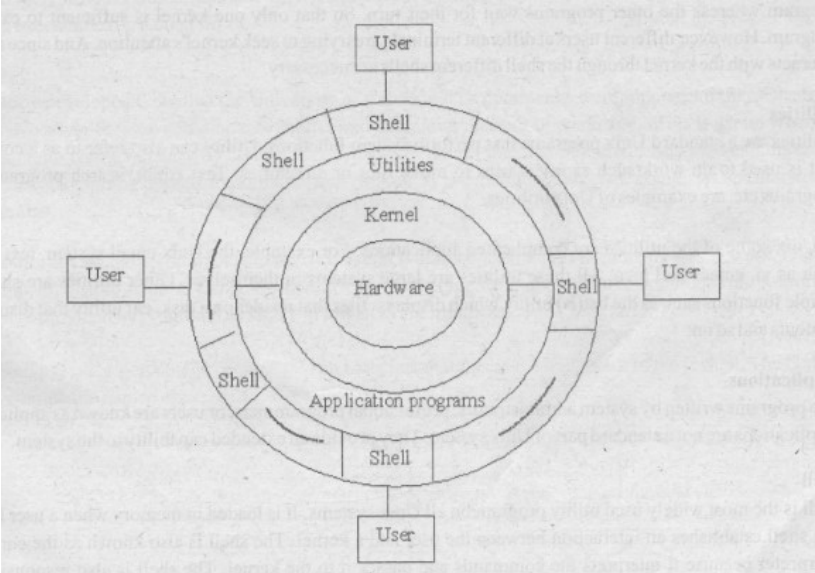
X. Networking

- LINUX was not originally a networking system.
- Networking allows users at one location to log into systems at the other ends and enables user to access the resources of host computer.

1.2 Components of Linux OS (Hardware, Kernel, Shell, GNU Utilities & Applications)

LINUX System Organization

- LINUX is a layered operating system
- There are three major components of LINUX Operating system.
- The function of LINUX is managed in three levels. Fig (1.c) shows the three layers of LINUX OS.



[Fig (1.c): Components of UNIX operating system]

(1) Hardware

- The innermost layer is the hardware that provides the services for the operating system. Hardware includes storage, memory, processor, I/O devices and peripherals etc.

(2) Shell

- The shell is on outer layer. It is a command interpreter which interprets the commands supplied by user at shell prompt.
- Then, shell conveys interpreted command to the kernel which ultimately executes them.
- **It is actually an interface between user and the kernel.**
- When any user logged in successfully to the LINUX system then a shell is allocated to that user.
- Even though there will be only one kernel running, multiple shells will be active – one for each user.
- Shell offers a prompt through which user issue a command to the LINUX system.
- The shell is represented by ***sh***(Bourne Shell)***, csh*** (C Shell)***, ksh*** (Korn shell)***, bash*** (Bash shell).

(3) Kernel

- Kernel is an important component of LINUX OS. It resides between hardware and shell.
- Kernel is a heart of LINUX, which interacts with the actual hardware in machine language.
- The kernel has overall control of everything.
- Kernel is loaded into memory when the system is booted and communicates with the hardware. The application programs access the kernel through a set of functions called as system calls.
- The kernel manages various OS tasks like memory management, process scheduling, deciding job priorities etc.
- Even if none of the user programs are running, kernel will be working in a background.
- The kernel performs various functions such as
- It manages files, carries out all the data transfer between the file system and the hardware.
- It manages memory i.e. assigns memory to each of the programs that are running.
- The kernel is responsible for scheduling of various programs running in memory or allocation of CPU time to all running programs.
- It also handles any interrupts issued, all I/O operations, and all other low-level services and so on.
- The kernel program is usually stored in a file called ‘LINUX’ whereas the shell program is in a file called 'sh'.
- For each user working with LINUX at any time different shell programs are running.
- So there may be several shells running in memory but only one kernel.

- This is because; at any instance LINUX is capable of executing only one program whereas the other programs wait for their turn.
- So that only one kernel is sufficient to execute a program.
- However, different users at different terminals are trying to seek kernel's attention.
- And since the user interacts with the kernel through the shell different shells are necessary.

(4) GNU Utilities

- GNU stands for Gnu's Not Unix, and it is pronounced as “g-noo”. It is a recursive acronym, and it stands for “Gnu's Not Unix”. GNU is a free and open-source operating system that was started in 1984 by Richard Stallman. GNU is based on the Unix operating system, but it has been greatly modified over the years.
- GNU is an extensive collection of free software, which can be used as an operating system or can be used in parts with other operating systems. The use of the completed GNU tools led to the family of operating systems popularly known as Linux.
- GNU utilities, particularly the GNU Core Utilities (coreutils), are a fundamental set of software tools that provide essential functionalities for managing and interacting with Linux systems. These utilities form the backbone of many basic tasks on a GNU/Linux system, including file manipulation, shell operations, and text processing. They are crucial for system administration and user-level operations.

  Core Functionality:
  File Management:
  Commands like ls, cp, mv, rm, mkdir, rmdir, cat, head, tail, chmod, chown, and chgrp are used for listing files, creating directories, copying, moving, removing files and directories, changing permissions, and more.
  Shell Operations:
  Utilities like echo, pwd, cd, export, env, and test (or [ ) are essential for shell interaction, variable handling, and conditional logic.
  Text Processing:
  Commands like sed, awk, grep, sort, uniq, wc, and tr are used for searching, manipulating, and transforming text data

(5) GNU Applications

GNU applications in Linux encompass a wide range of software tools and utilities that are part of the GNU operating system. These include essential components like the Bash shell, the GNU Compiler Collection (GCC), the GNU C library (glibc), and the GNU Core

Utilities (coreutils). GNU applications are foundational to many Linux distributions, providing the basic building blocks for interacting with the system and running other programs.

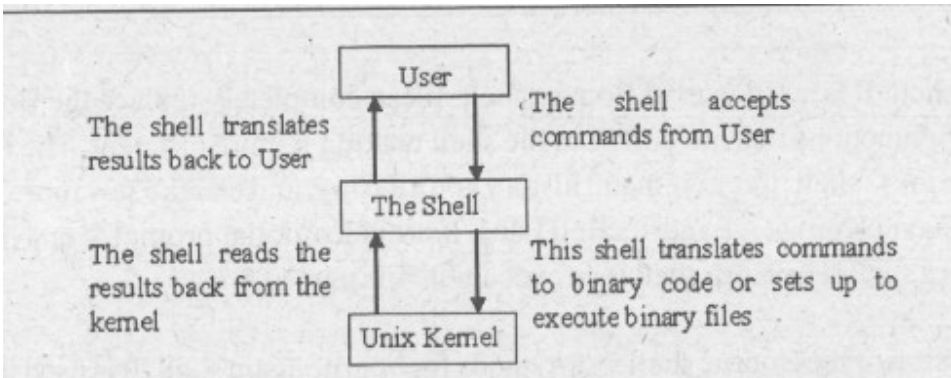
Core GNU Components and Utilities:

Bash Shell: A command-line interpreter that allows users to interact with the operating system.
GCC (GNU Compiler Collection): A powerful compiler suite that supports various programming languages and architectures.
glibc (GNU C Library): A fundamental library providing essential functions for C and C++ programs.
Coreutils: A collection of basic file, shell, and text manipulation utilities.
GDB (GNU Debugger): A powerful tool for debugging programs.
Binutils (GNU Binary Utilities): A collection of tools for working with binary files, including assemblers, linkers, and disassembles.

1.3 Shell in Linux (Bash, Zsh, Dash – Features and Differences)

Shell

- Shell is loaded in memory when a user logs in. The shell establishes an interaction between the user and a kernel.
- The shell is also known as the command Interpreter because it interprets the commands and passes it to the kernel.
- The shell is also responsible for returning the results of the commands to the terminal, a file, or another device. Following figure shows the relationship of the shell to LINUX kernel.



[Fig (1.d): Relationship of the shell to Unix kernel]

- The shell can be considered as a master utility program which enables a user to gain access to all other utilities and resources of the computer.

A Shell performs following functions:

Command Interpretation: The shell reads commands entered by the user (or from a script) and interprets them.

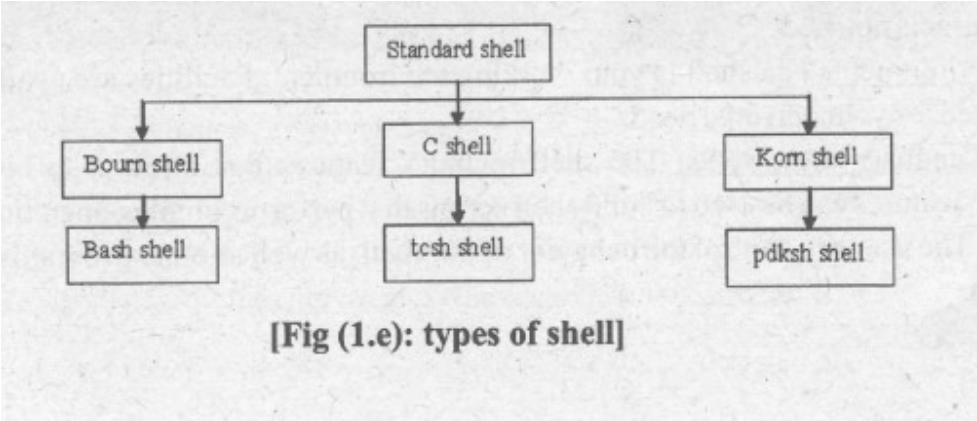
Program Execution: It then executes these commands, often by launching other programs or utilities.

User Interface: The shell provides a text-based interface (command-line interface or CLI) for users to interact with the operating system.

Scripting: Shells also allow users to write scripts (sequences of commands) to automate tasks.

Types of Shells

- The shell runs like any other program under the LINUX system.
- Hence, one shell program can replace the other, or call another shell program as it would call any other program.
- Due to this feature, a number of shells have been developed in response to different needs of users. Some of the popular shells are listed below (see fig (1.e)):



[Fig (1.e): types of shell]

1. **Bourne shell:**
   - This is one of the most widely used shells in the LINUX world.
   - It is the primary LINUX command interpreter and comes along with every LINUX system.
   - It provides 'dollar prompt $' on LINUX installations as the trademark of the Bourne shell. The Bourne shell is an executable **file named *sh.***
2. **bash shell:**
   - It is an enhanced version of the Bourne shell. *bash* stands for bourne again shell.

3. **The Z shell (Zsh):** is a Unix shell that can be used as an interactive login shell and as a command interpreter for shell scripting. Zsh is an extended Bourne shell with many improvements, including some features of Bash, ksh, and tcsh.
4. **Dash** is a lightweight, POSIX-compliant shell available on Linux systems. It's designed to be small and fast, making it a good choice for system scripts and as a replacement for /bin/sh on some distributions.
   Key features of Dash:
   POSIX Compliance: Dash adheres to the POSIX standard, ensuring a degree of compatibility across different Unix-like systems.
   Speed and Small Size: Dash is known for its efficiency and minimal resource usage, which is advantageous for system-level tasks.
   /bin/sh Replacement: On many Debian-based and other systems, /bin/sh is a symbolic link to dash, making it the default command interpreter for scripts that rely on sh.
   Limited Feature Set: Compared to more feature-rich shells like Bash, Dash has a smaller set of built-in commands and features, which contributes to its speed and size.

Difference between Bash, Zsh, Dash

Bash, Zsh, and Dash are all Unix shells, but they differ in their features, customization options, and intended use.

Bash is a widely used, reliable shell, known for its scripting capabilities and POSIX compliance.

Zsh is a more advanced, interactive shell with extensive customization and features like auto-completion and plugin support.

Dash is a minimalist, faster shell, primarily focused on POSIX compliance and performance.

Features of shell

(1) Interactive Environment:

- The shell allows the user to create a dialogue, i.e. communication channel, between the user and the host LINUX system.
- This dialogue terminates until the user ends the session.

(2) Shell scripts:

- It is the shell that has the facility to be 'programmed'.
- Shell script is a file that contains shell commands that perform a useful function.
- In other word, Shell scripts are groups of LINUX commands strung together and executed as individual files.
- It is also known as a shell program.
- The shell is itself a program, except that it is written in C.

(3) Input /Output redirection:

- I/O redirection is a function of the shell that redirects the output from program to a destination other than the screen.
- This way, user can save the output from a command into a file and redirect it to a printer, another terminal on the network or even another program.
- Similarly, the shell can make a program that accepts input from other than the keyboard by redirecting its input from another source.

(4) Piping Mechanism:

- A pipe operator receives its input from standard output and sends it to the next command through standard output i.e. piping mechanism allows the output of one command to be used as input for another LINUX command.
- Using piping, programs that perform simple functions can easily be connected to perform more complex functions, minimizing the need to develop new programs.

(5) Meta-character facilities / filename substitution:

- Shells recognize the *,? or [..] as special characters when reading the arguments from a command line.
- Shell then performs filename expansion on this list before executing the requested program.

e.g. ls s* -- it substitutes all the filenames of working directory whose name begins with 's' at command line and then *ls* command display list of filenames.

(6) Background processing:

- A Multi-tasking facility allows the user to run command in the background.
- This allows the command to be processed while the user can proceed with other tasks.
- When a background task is completed, the user is notified.

(7) Customized Environment:

- The shell is your working environment.
- Facilities are available by which the shell can be customized for your personal needs.

(8) Programming Language Constructs:

- The shell includes features that allow it to be used as programming language.
- These features can be used to build shell scripts that perform complex operations.

(9) Shell variables:

- The user can control the behavior of the shell, as well as other programs and utilities, by storing data in variables.

1.4 Introduction to Files and File Types in Linux (text, binary, special files)

In Linux, files are broadly categorized into text files, binary files, and special files. Text files contain human-readable characters and are often used for storing source code or configuration data. Binary files store data in a format that is not intended for direct human interpretation, such as compiled code or images. Special files represent devices or other system resources.

Text Files:

Definition:

Text files store data in a format that can be directly read and understood by humans, using character encoding like ASCII or UTF-8.

Examples:

Source code files (.c, .py, .java), configuration files (.txt, .ini), and scripts.

Characteristics:

Typically consist of lines of text, each terminated by a newline character.

Can be opened and edited using text editors like nano, vim, or gedit.

Can be easily manipulated by various command-line tools like grep, sed, and awk.

Binary Files:

Definition:

Binary files store data in a format that is not intended for direct human reading. They contain sequences of bytes that are interpreted by specific programs.

Examples:

Executable files (.out, .exe), image files (.jpg, .png), audio files (.mp3, .wav), and compressed files (.zip, .tar.gz).

Characteristics:

Data is not organized into lines.

Can't be reliably opened and edited with a text editor.

Specific programs are required to interpret their contents.

Often contain control codes and other data not meant for human consumption.

Special Files:

Definition:

Special files are not regular files containing data but rather represent devices or other system resources. They are located in the /dev directory.

Examples:

Block special files: Represent block devices like hard drives and partitions (e.g., /dev/sda, /dev/sda1).

Character special files: Represent character devices like terminals and printers (e.g., /dev/tty, /dev/lp0).

FIFO (named pipes): Allow processes to communicate with each other (e.g., /tmp/myfifo).

Characteristics:

Used for input/output operations with hardware or other system resources.

Don't contain regular file data.

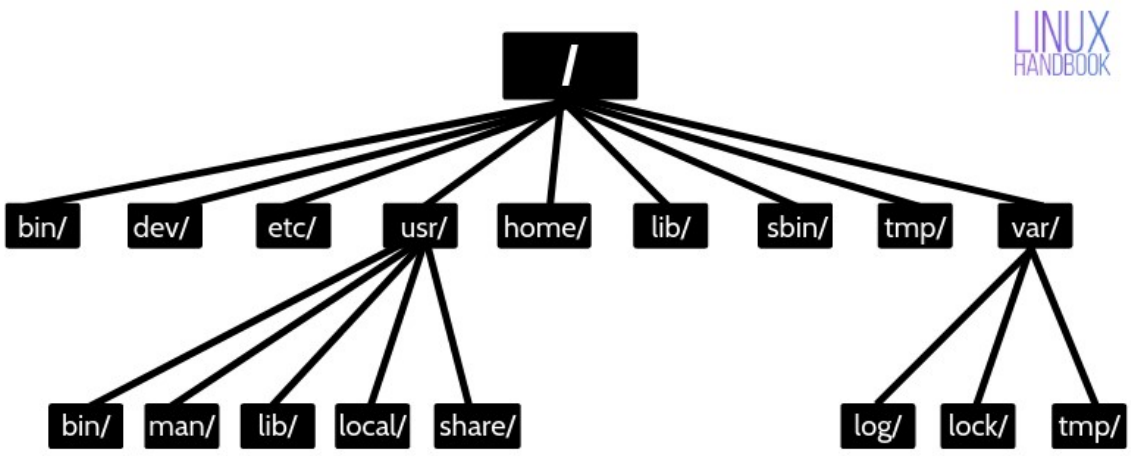
Accessed by the operating system to interact with devices and other resources.

1.5 Linux Directory Structure and File System Hierarchy Standard (FHS)

The Linux directory structure, as defined by the File system Hierarchy Standard (FHS), is a hierarchical system starting from the **root directory represented by a forward slash (/).**

This standard ensures consistency across different Linux distributions, specifying the purpose of each directory and where specific types of files should be stored.

Key directories include /bin (essential command binaries), /etc (system configuration files), /home (user home directories), /var (variable data like logs), and /usr (user programs and libraries).



Key aspects of the FHS and Linux directory structure:

Root Directory (/): The top-level directory from which all other directories and files originate.

Standard Directories:

The FHS defines standard directories like /bin, /etc, /home, /var, /usr, and others, each with a specific purpose.

Purpose of Directories:

/bin: Contains essential command binaries used by all users.

/etc: Stores system-wide configuration files.

/home: Contains individual user's home directories.

/var: Holds variable data, such as log files, databases, and website content.

/usr: Contains user binaries, libraries, documentation, and other user-related programs.