Chapter – 9

UNIX / LINUX OPERATING SYSTEM

What is

Unix / Linux?

- UNIX is a powerful operating system originally developed at AT&T Bell Labs.
- It is very popular among the scientific, engineering, and academic communities due to its
  - multi-user and multi-tasking environment,
  - flexibility and portability,
  - electronic mail and networking capabilities, and
  - the numerous programming, text processing and scientific utilities available.
- The UNIX system is mainly composed of three different parts: the kernel, the file system, and the shell.

- The kernel is “that part of the system which manages the resources of whatever computer system it lives on, to keep track of the disks, tapes, printers, terminals, communication lines and any other devices”.
- The file system is the organizing structure for data. The file system is perhaps the most important part of the Linux operating system.

- “The shell is the command interpreter”. Although the shell is just a utility program, and is not properly a part of the system, it is the part that the user sees.
- The shell listens to your terminal and translates your requests into actions on the part of the kernel and the many utility programs.
- Linux is a freely available, open source, Unix-like operating System.
- Linux now runs on multiple hardware platforms, from the smallest to the largest, and serves a wide variety of needs from servers to movie-making to running businesses to user desktops.

Features of the Unix

1) **Open Source** - Linux source code is freely available and it is community based development project. Multiple teams works in collaboration to enhance the capability of Linux operating system and it is continuously developing.

2) **Portable** - Portability means softwares can work on different types of hardwares in same way. Linux kernel and application programs supports their installation on any kind of hardware platform. It is one of the main reasons for the popularity of the Unix. A portability credit of the UNIX is because of the C language, it written in C language and C language is portable.

3) **Multi-User** - Linux is a multiuser system means multiple users can access system resources like memory/ ram/ application programs at same time.

4) **Multiprogramming** - Linux is a multiprogramming system means multiple applications can run at same time. This is managed by dividing the CPU time between all processes being carrying out.

5) **Shell** - Linux provides a special interpreter program which can be used to execute commands of the operating system. It can be used to do various types of operations, call application programs etc.

6) **Security** - Linux provides user security using authentication features like password protection/ controlled access to specific files/ encryption of data.

7) **Hierarchical File System** - Linux provides a standard file structure in which system files/ user files are arranged.

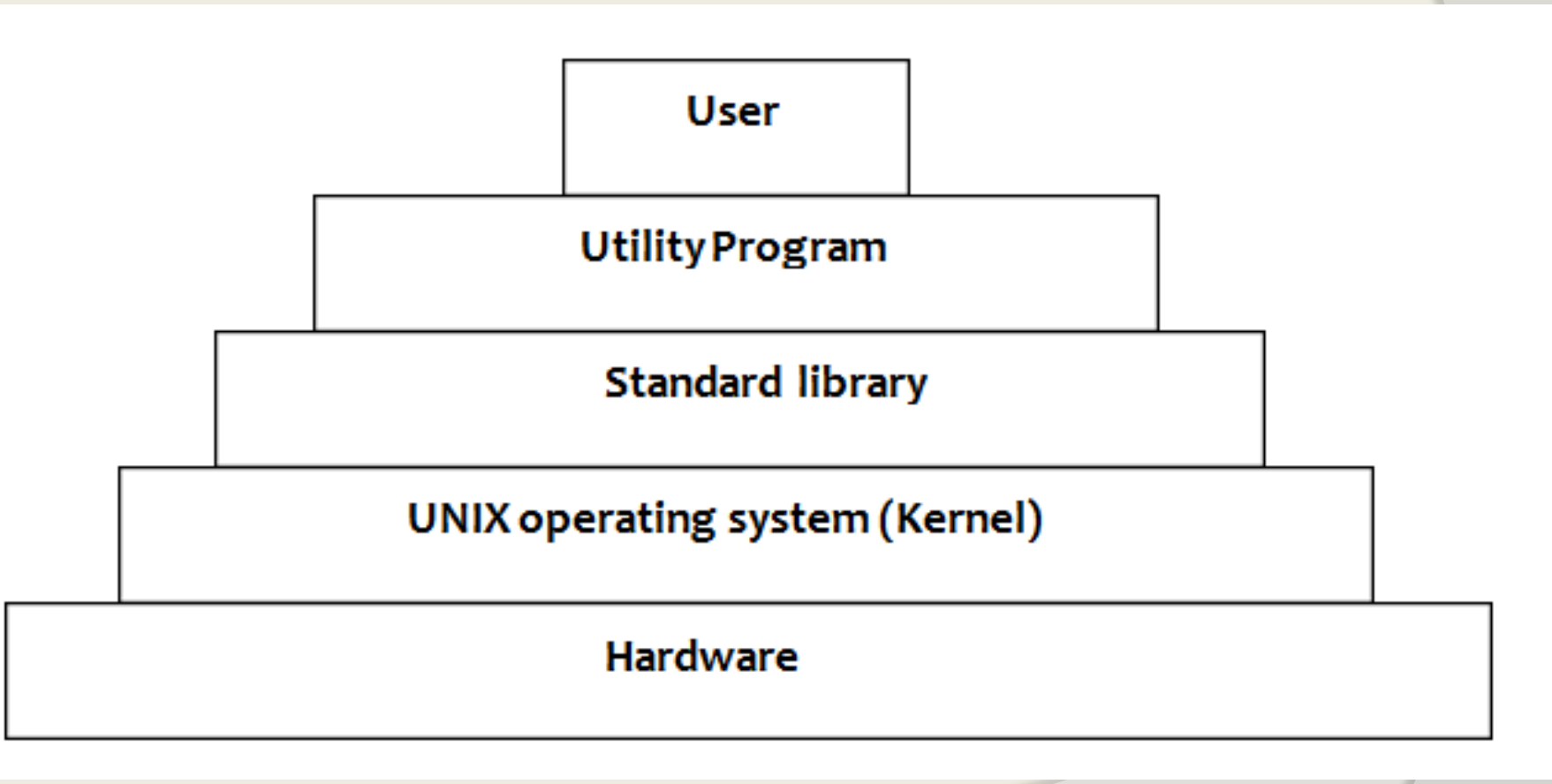
Comparison of Unix and Windows

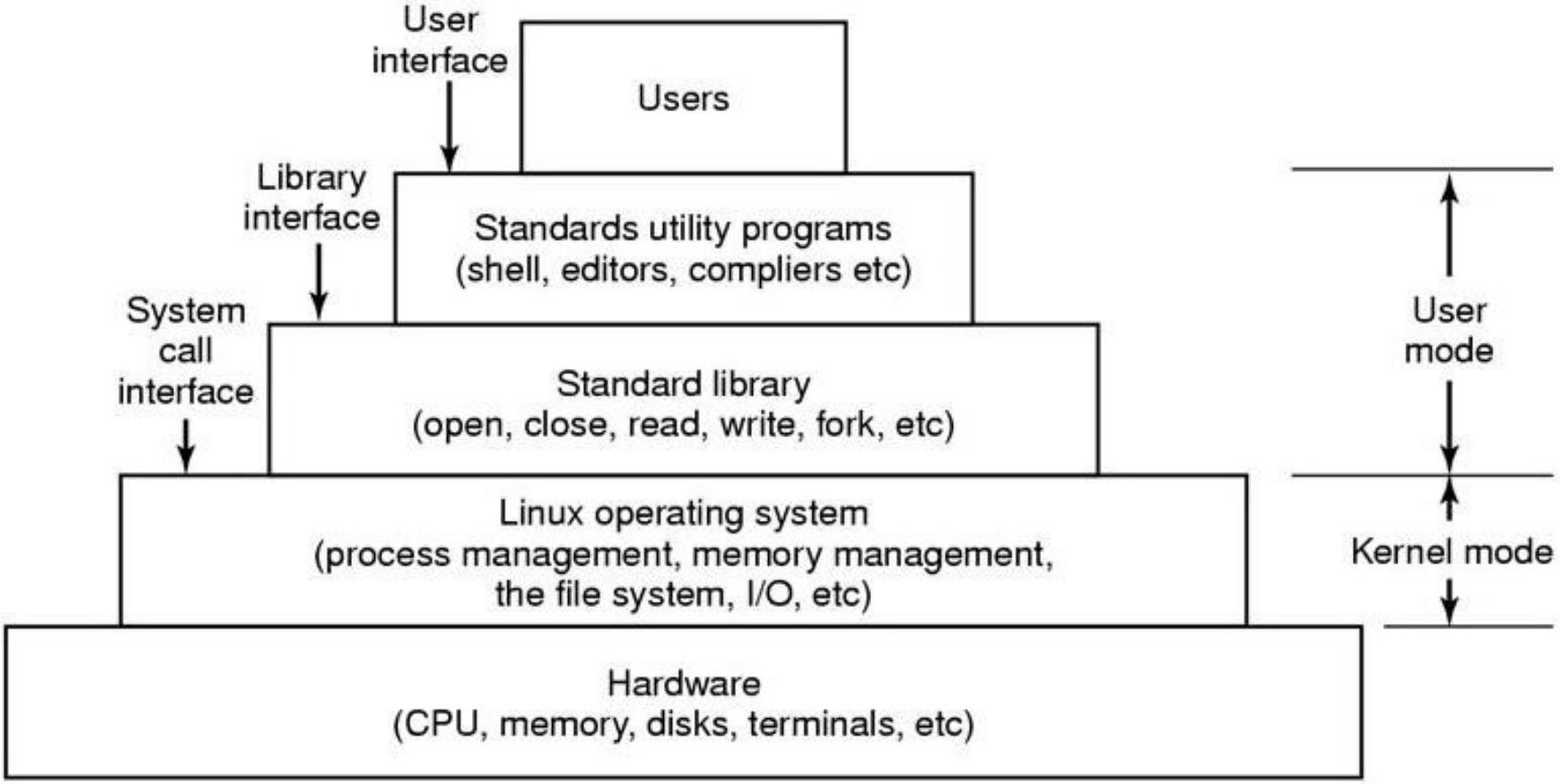
UNIX	Windows
1) UNIX is Programmed in **C**	1) Windows is Programmed in **Assembly Language, C** and **C++.**
2) Its main target is to provide services to **Server Computers** and Personal Computers.	2) Its main target is to provide services to Personal Computers and in **Business**
3) The UNIX Operating System is **Open Source** which means everyone can use it and edit it.	3) Windows is a **closed-source** operating system. It was coded and created by Microsoft. People are not able to edit it, or change its code.
4) UNIX is **CLUI** (Command Line User Interface) operating system, thus it a **command based** operating system, however it does have a GUI like windows.	4) Windows is **GUI** (Graphical User Interface) Operating System. Thus it is purely **GUI-based** (having Menus).
5) Text mode interface :BASH (Bourne Again SHell) is the UNIX default shell. It can support multiple command interpreters.	5) Text mode interface : Windows uses a single command interpreter with dos-like commands.

UNIX	Windows
6) UNIX can boot from either a **primary partition** or a **logical partition** inside an extended partition.	6) Windows must boot from a **primary partition**.
7) UNIX is **less affected by viruses** and malwares and hence more secure compared to windows.	7) Windows is **more affected by viruses** and malwares and hence less secure compared to Linux.
8) Examples of UNIX operating system are: Ubuntu, Fedora, Red Hat, Debian, Archlinux, Androidetc, etc.	8) Examples of windows operating system are: Windows Vista, Windows XP. Windows 7, Windows 8, Windows 8.1, Windows 10, etc.

Linux Architecture

Linux Architecture

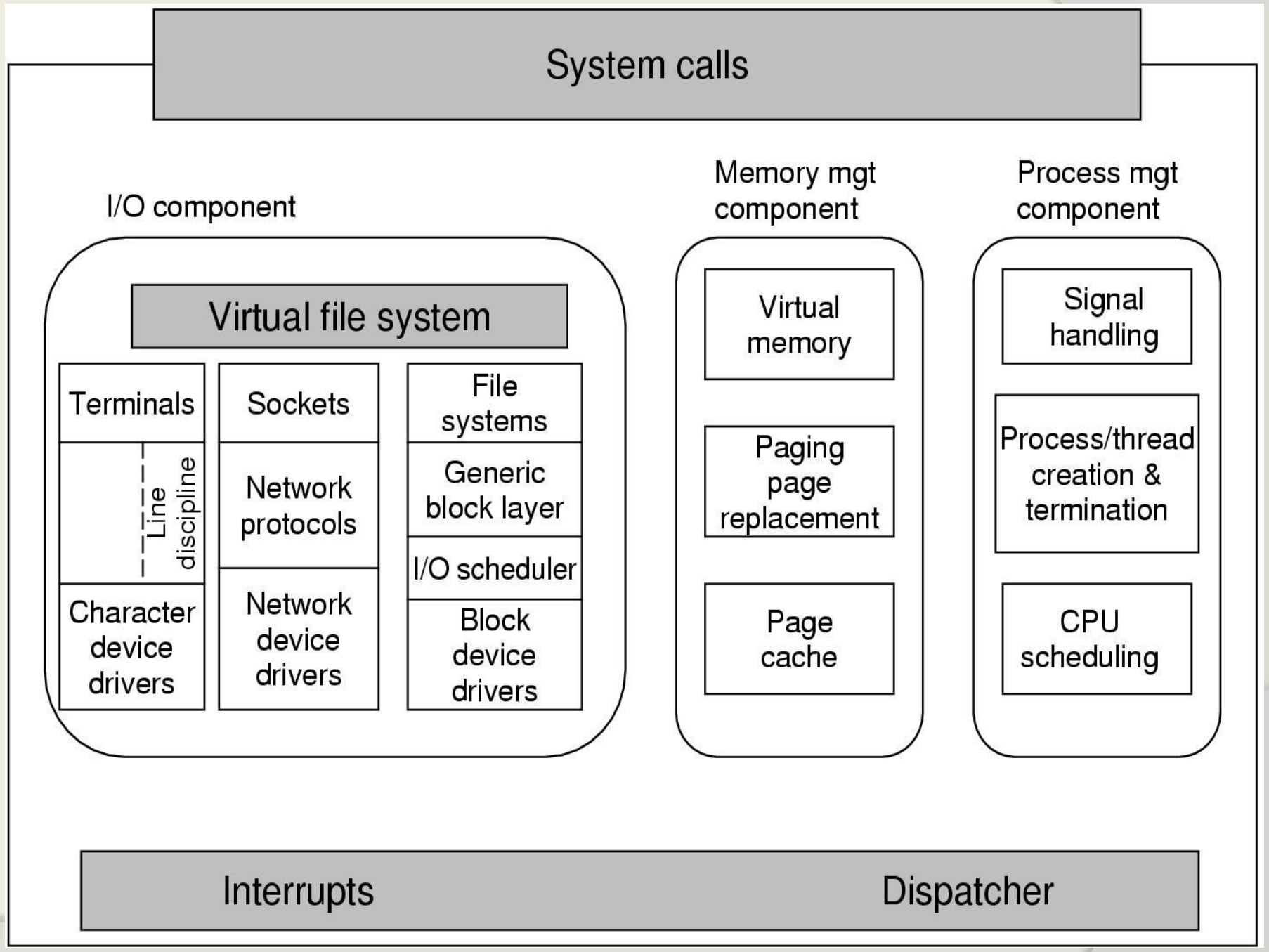



User interface
Users
Library interface
Standards utility programs (shell, editors, compliers etc)
System call interface
Standard library (open, close, read, write, fork, etc)
Linux operating system (process management, memory management, the file system, I/O, etc)
Hardware (CPU, memory, disks, terminals, etc)
User mode
Kernel mode

- **Hardware**
  - The bottom layer is hardware.
  - It contains physical devices of computer like CPU, Memory, Disk, printer etc.
  - UNIX kernel will interface with this hardware.
- **UNIX Kernel**
  - Kernel is program which provides services of OS like memory management, file management and process management.
  - Kernel will provide interface with hardware and user programs.

- **Standard library**
  - It contains set of procedures.
  - This is collection of system level files.
- **Utility program**
  - Utility programs are used to make user programs and make work easier.
  - Utility programs like compilers, assemblers, editors etc.
- **User**
  - Users can directly interact with the user programs. User programs can interact with system.

Linux Kernel Structure
or
Functions of Linux Kernel


System calls
I/O component
Memory mgt component
Process mgt component
Virtual file system
Terminals
Line discipline
Character device drivers
Sockets
Network protocols
Network device drivers
File systems
Generic block layer
I/O scheduler
Block device drivers
Virtual memory
Paging page replacement
Page cache
Signal handling
Process/thread creation & termination
CPU scheduling
Interrupts
Dispatcher

- The kernel sits directly on the hardware and enables interactions with I/O devices and the memory management unit and controls CPU access to them.

1) Interrupt and Dispatcher

- Interrupt handlers are the primary way for interacting with devices
- Dispatching occurs when an interrupt occurs.
- Process dispatching also happens when the kernel completes some operations and it is time to start up a user process again.

2) VFS (Virtual File System)

- At the highest level, the I/O operations are all included under a **VFS** (**Virtual File System**) layer.
- That is, at the top level, performing a read operation on a file, whether it is in memory or on disk, is the same as performing a read operation to retrieve a character from a terminal input.

3) Device Driver

- At the lowest level, all I/O operations pass through some device driver.
- All Linux drivers are classified as either character-device drivers or block-device drivers.

- The main difference between both the devices is that random accesses are allowed on block devices and not on character devices.

4) Network Device Drivers

- Technically, network devices are really character devices, but they are handled somewhat differently, so in the figure it is shown differently.
- Above the device-driver level, the kernel code is different for each device type.
- Networking software is often modular, with different devices and protocols supported.

- The layer above the network drivers handles a kind of routing function, making sure that the right packet goes to the right device or protocol handler.
- Above the router code is the actual protocol stack, including IP and TCP, but also many additional protocols.

5) Sockets

- Above all the network is the socket interface, which allows programs to create sockets for particular networks and protocols.

6) I/O Scheduler

- I/O scheduler is responsible for ordering and issuing disk operation requests.

7) File Systems

- In order to hide the complex architectural differences of various hardware devices from the file system implementation, a generic block-device layer provides an abstraction used by all file systems.

8) Memory Management

- Memory-management tasks include maintaining
  1. virtual to physical-memory mappings,
  2. maintaining a cache of recently accessed pages
  3. implementing a good page-replacement policy, and
  4. on-demand paging

9) Process Management

- The key responsibility of the process-management component is
- creation and termination of processes.
- process scheduling, which chooses which process or thread to run next.
- The Linux kernel treats both processes and threads simply as executable entities, and will schedule them based on a global scheduling policy.

10) System Calls

- Finally, at the very top is the system call interface into the kernel.
- All system calls come here, causing a trap which switches the execution from user mode into protected kernel mode and passes control to one of the kernel components described above.

UNIX Commands and Shell

UNIX Commands and Shell

- “A *command* is an instruction given by a user telling a computer to do something, such as run a single program or a group of linked programs”.
- “Shell is a program that takes your commands from the keyboard and gives them to the operating system to perform specified task”.
- In the old days, it was the only user interface available on a Unix computer.
- Nowadays, we have *graphical user interfaces (GUIs)* in addition to *command line interfaces (CLIs)* such as the shell.

UNIX Commands and Shell

- On most Linux systems a program called bash (which stands for Bourne Again SHell) acts as the default shell program.
- There are several additional shell programs available on a typical Linux system. These include: ksh, tcsh and zsh.
- A shell is an environment in which user can run commands, programs, and shell scripts.
- It gathers input from user and executes programs based on that input, when a program finishes executing; it displays that program's output.

- The shell has the terminal as standard input and standard output.
- It starts out by typing the prompt, a character such as $, which tells user that shell is waiting to accept command.
- For example if user types date command,
- **$ date**
- Sat Mar 11 08:30:19 IST 2017
- The shell creates a child process and runs date program as child.
- While the child process is running, the shell waits for it to terminate.

- When child finishes, the shell types the prompt again and tries to read the next input line.
- Shell is work as an interface, command interpreter and programming language.

- Shell - As interface
  - Shell is interface between user and computer.
  - User can directly interact with shell.
  - Shell provide command prompt to user to execute commands.
- Shell - As command interpreter
  - It reads commands entered by user on prompt.
  - It Interprets the command, so kernel can understand it easily.
- Shell - As programming language
  - Shell is also work as programming language.
  - It provides all features of programming language like variables, control structures and loop structures.