Introduction to OS, Single user, Multi user and Network OS, H/W dependent and independent OS, Important OS components, Kernel, Shell, Shell types

Instructor: Faria Waqar

Lecturer CS department

Primitive Data and Operation

- Processor is brain of computer: controls and coordinate activities of devices. Continuously carries out operations.
- Operation: maps non-empty set: input set into another set output set.
- Elements of these sets are data values or simply data.
- Entity that carries out information processing called execution unit.
- Some operations may be primitive as we know what they do but we not know the rules of operations.

- Execution of an operation is the application or realization of these rules or primitives.
- Depositing any amount in a bank is the operation.
- Depositing 100 rs at 11:00 pm on 17th March 2020 is the operation execution.
- Each operation is assumed to be deterministic: that it produces the same output data value whenever it is applied on the same input set.

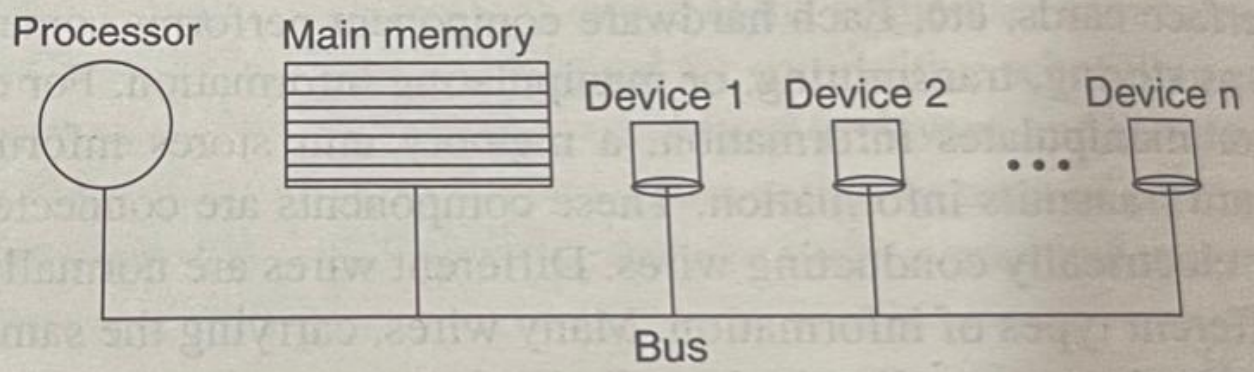


Figure 1.2: *A typical model of computer hardware.*

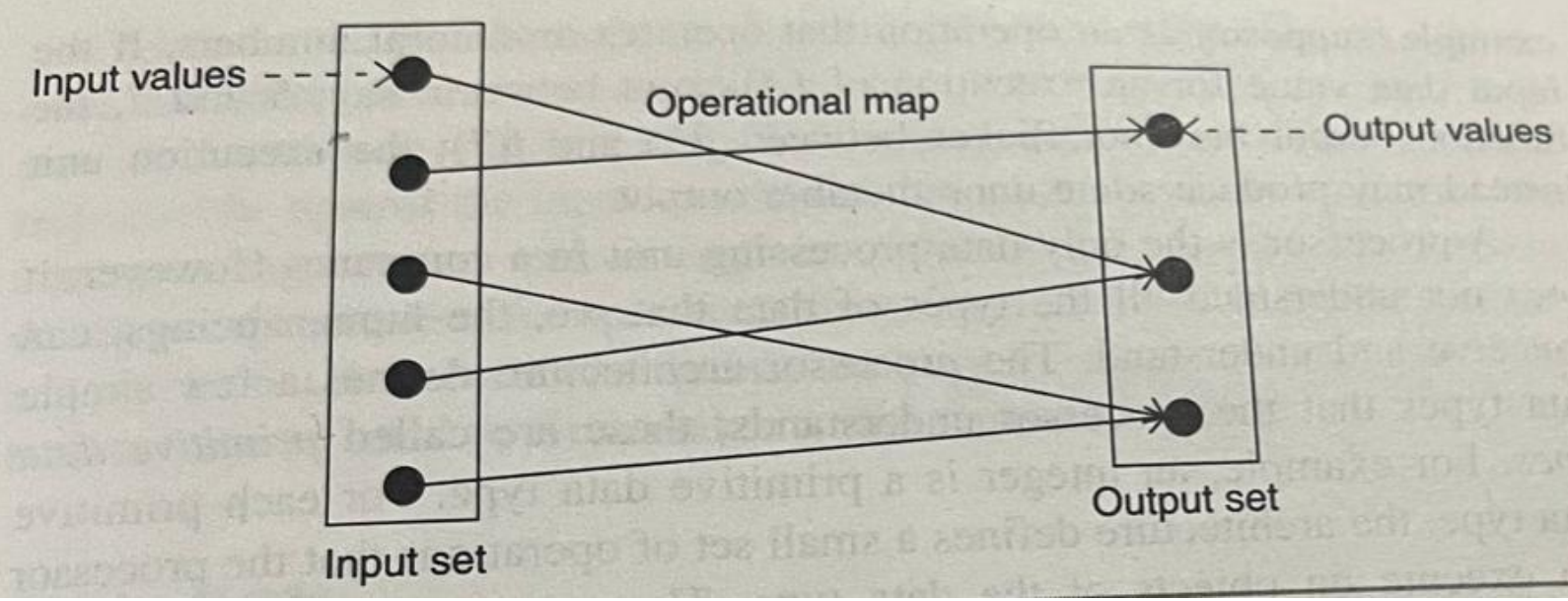
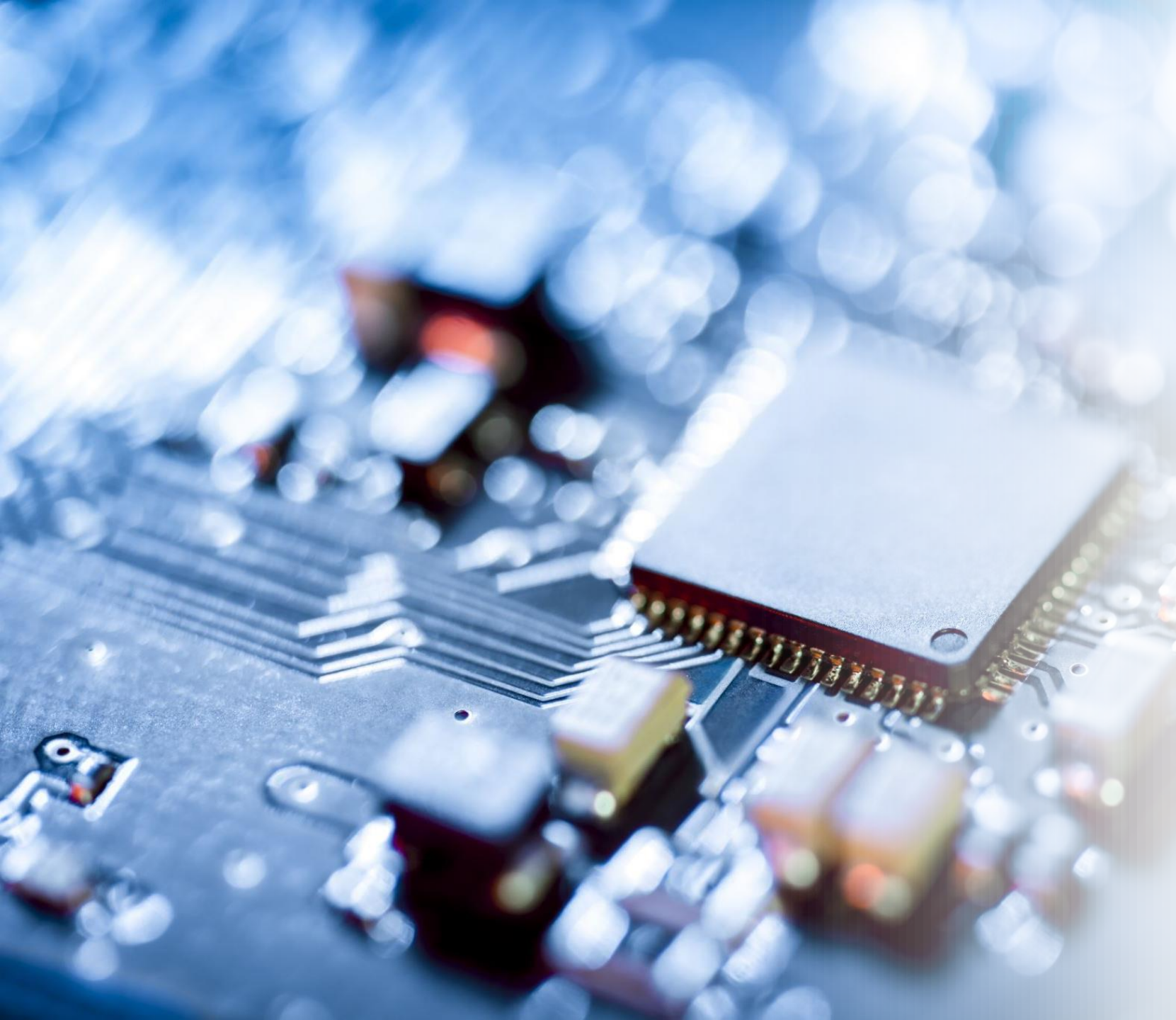


Figure 1.3: *An abstract representation of an operation.*

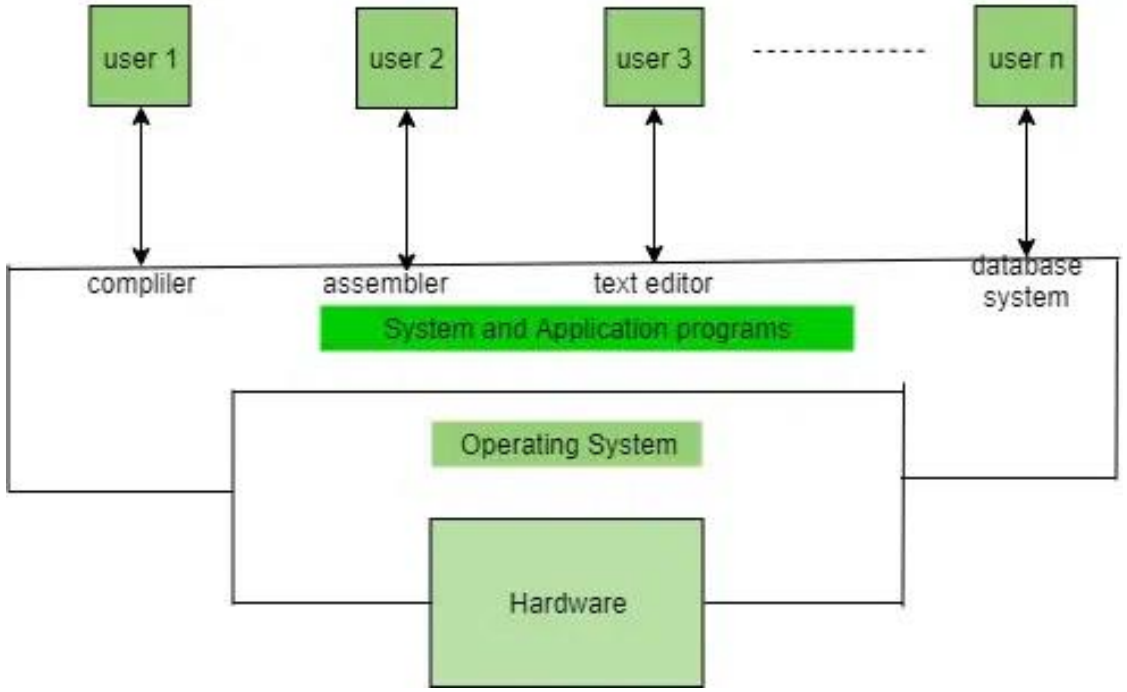


- A processor is only a data processing unit. It can understand the primitive data types. Eg integer is a primitive data type.
- **EXECUTION FLOW:**
- The task of processor is simple. It cyclically fetches, decodes and executes the instructions from memory locations. (eg main memory etc)
- Some instruction executions involve accessing operands form main memory and/or IO Devices.
- The location of next instruction is determined by the outcome of current instruction.
- Processor keeps the flow and by these locations.

- However some events may alter the normal execution: these events are called interrupts or exceptions.
- When interrupt occurs, processor breaks current execution flow. These events originate from I/O devices.
- Exception occur in the processor due to unwanted conditions like dividing by zero or illegal instructions.

What an Operating System does?

- The OS acts as a mediator between the user and the computer's hardware. It communicates with the different parts of the computer, such as the CPU, memory, storage devices, input/output devices, etc., to ensure that they all work together smoothly to execute commands from applications.



Views of OS

- **User View of Operating System:**
  The Operating System is an interface, hides the details which must be performed and present a virtual machine to the user that makes it easier to use. Operating System provides the following services to the user.
- Execution of a program
- Access to I/O devices
- Controlled access to files
- Error detection (Hardware failures, and software errors)

- **Hardware View of Operating System:**
  The Operating System manages the resources efficiently in order to offer the services to the user programs. Operating System acts as a resource manager:
- Allocation of resources
- Controlling the execution of a program
- Control the operations of I/O devices
- Protection of resources
- Monitors the data

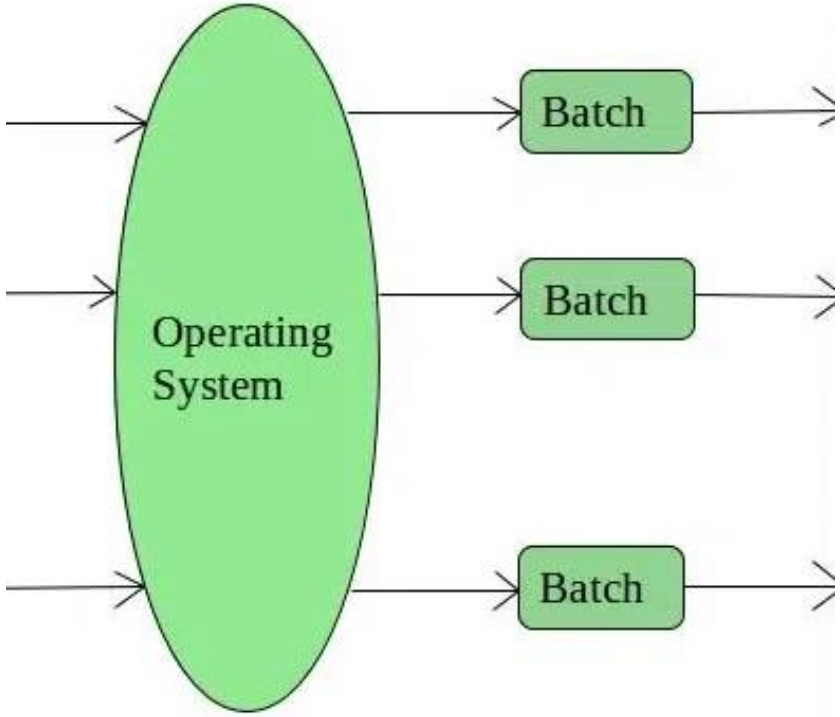
Views of OS

System View of Operating System:
Operating System is a program that functions in the same way as other programs. It is a set of instructions that are executed by the processor. Operating System acts as a program to perform the following.

- Hardware upgrades
- New services
- Fixes the issues of resources
- Controls the user and hardware operations

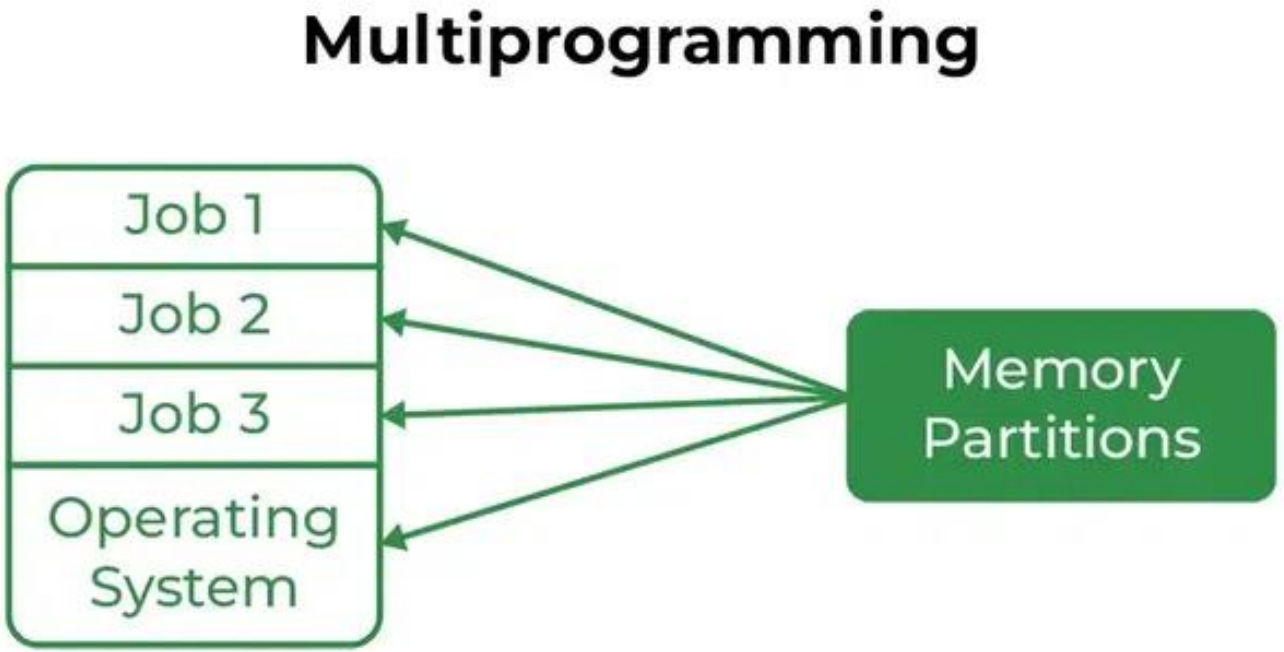
TYPES OF OPERATING SYSTEMS

- **Batch Systems**
- Operating Systems can be categorized according to different criteria like whether an operating system is for mobile devices (examples Android and iOS) or desktop (examples Windows and Linux). In this article, we are going to classify based on functionalities an operating system provides.
- **1. Batch Operating System**
- This type of operating system does not interact with the computer directly. There is an operator which takes similar jobs having the same requirements and groups them into batches. It is the responsibility of the operator to sort jobs with similar needs. Batch Operating System is designed to manage and execute a large number of jobs efficiently by processing them in groups.



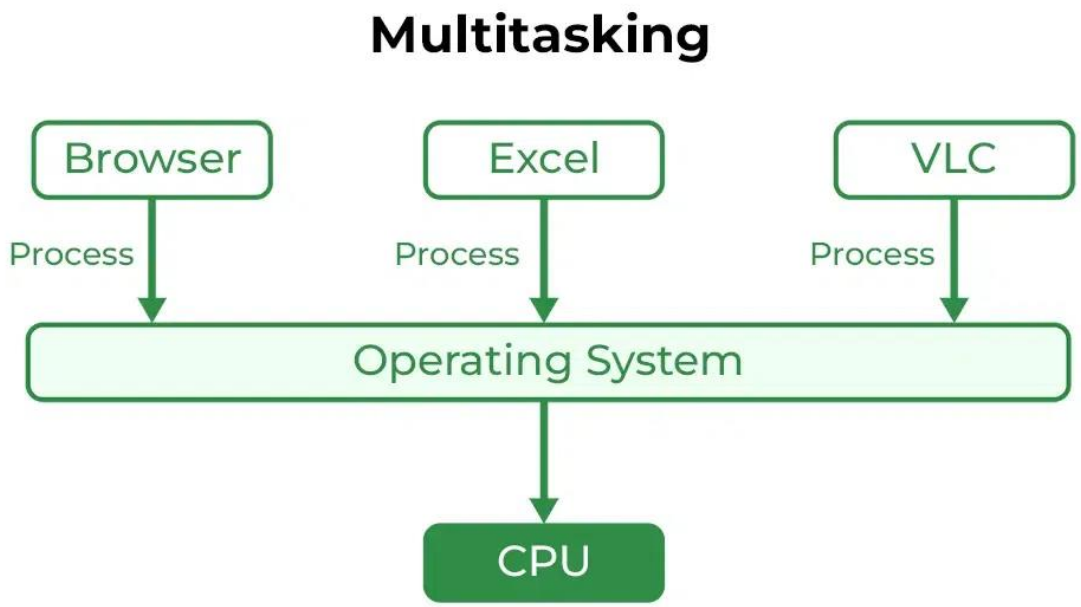
Multi-Programming Operating System

- Multiprogramming Operating Systems can be simply illustrated as more than one program is present in the main memory and any one of them can be kept in execution. This is basically used for better utilization of resources.

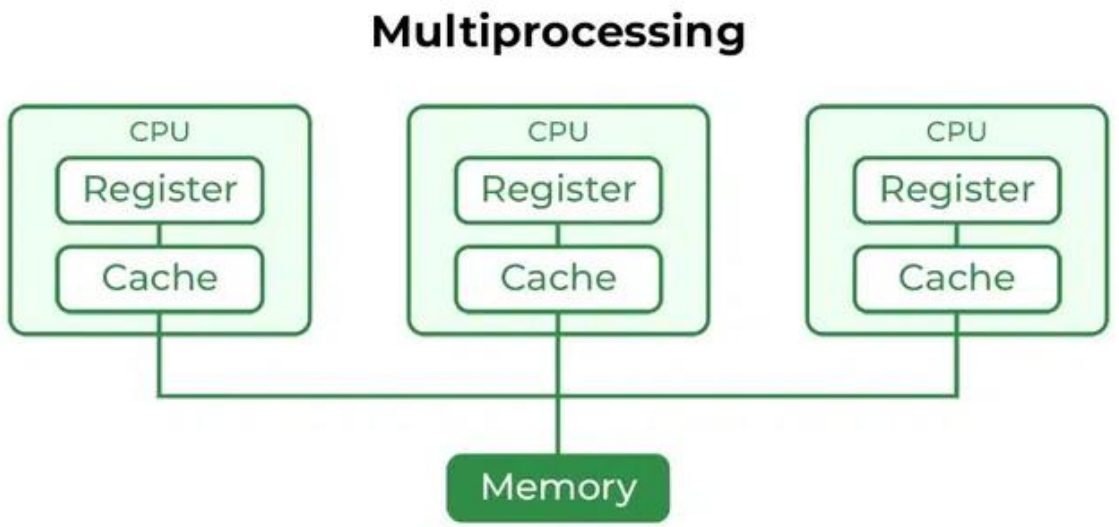


Multi-Tasking/Time-sharing Operating systems

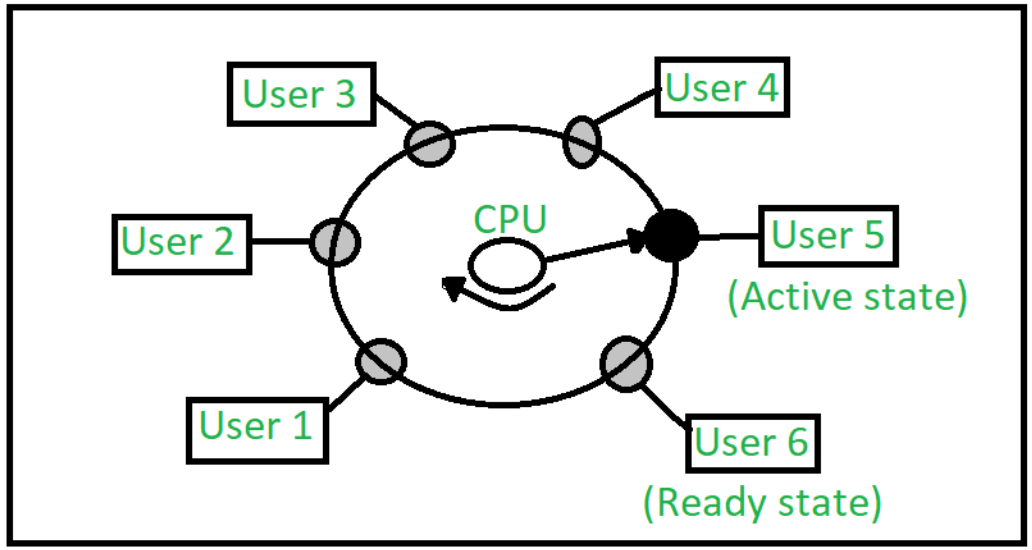
- It is a type of Multiprogramming system with every process running in round robin manner. Each task is given some time to execute so that all the tasks work smoothly. Each user gets the time of the CPU as they use a single system. These systems are also known as Multitasking Systems. The task can be from a single user or different users also. The time that each task gets to execute is called quantum. After this time interval is over OS switches over to the next task.



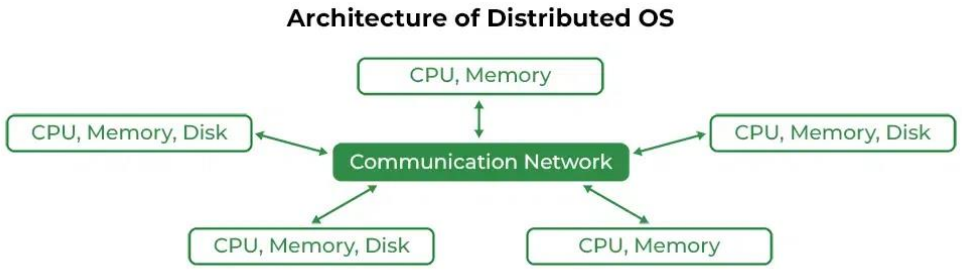
- **3. Multi-Processing Operating System**
- Multi-Processing Operating System is a type of Operating System in which more than one CPU is used for the execution of resources. It betters the throughput of the System.

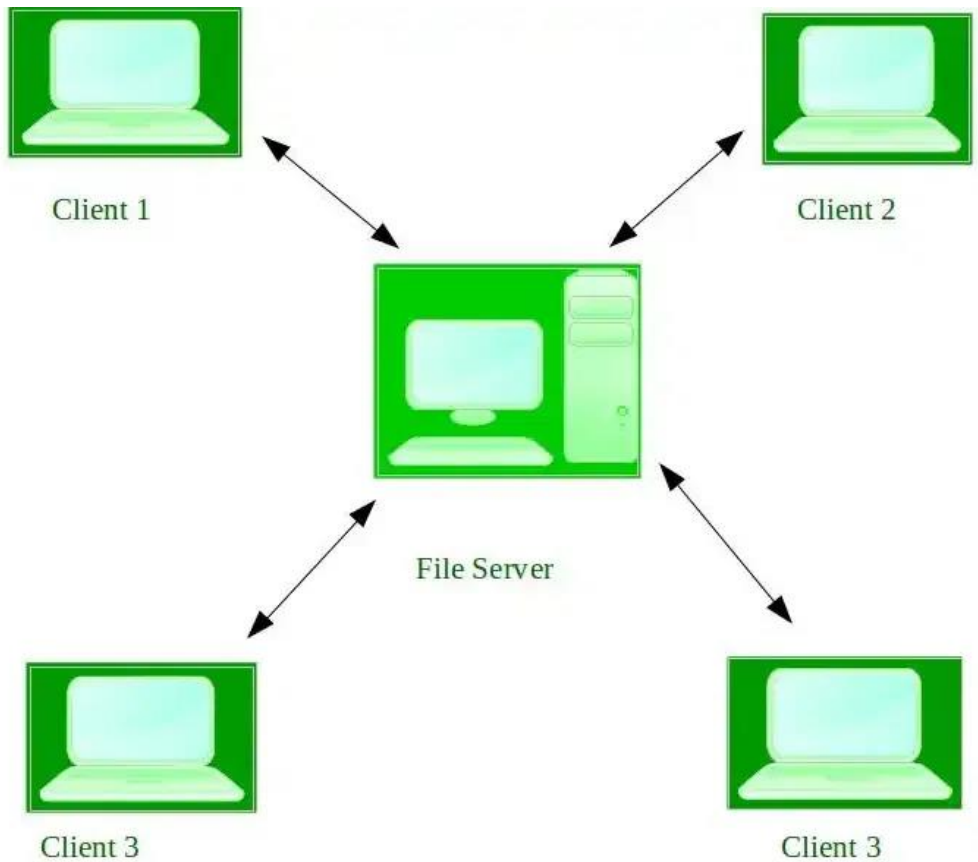


- **4. Multi User Operating Systems**
- These systems allow multiple users to be active at the same time. These system can be either multiprocessor or single processor with interleaving.



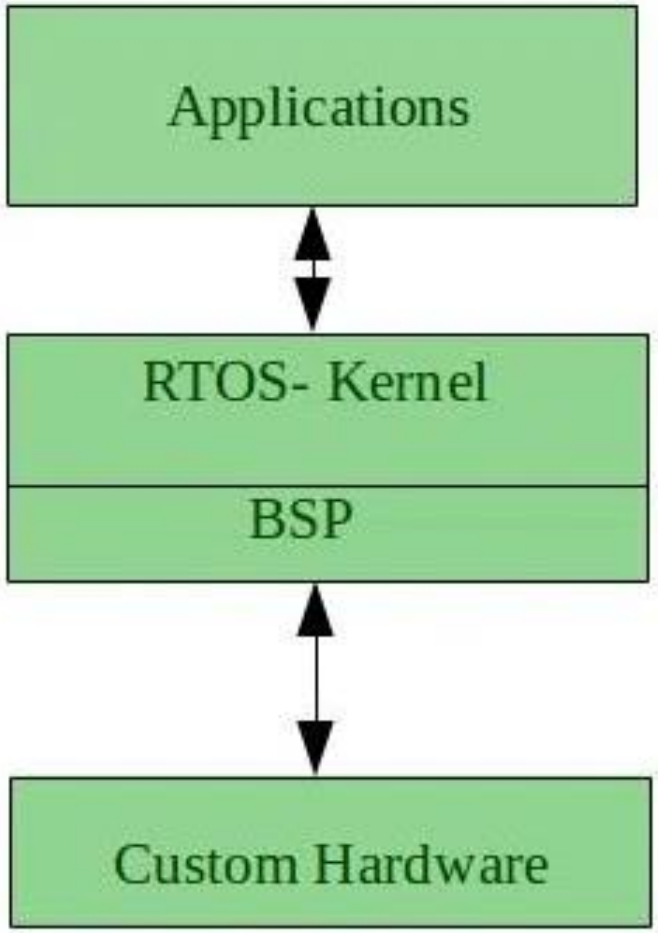
- **5. Distributed Operating System**
- These types of operating system is a recent advancement in the world of computer technology and are being widely accepted all over the world and, that too, at a great pace. Various autonomous interconnected computers communicate with each other using a shared communication network. Independent systems possess their own memory unit and CPU. These are referred to as loosely coupled systems or distributed systems . These systems' processors differ in size and function. The major benefit of working with these types of the operating system is that it is always possible that one user can access the files or software which are not actually present on his system but some other system connected within this network i.e., remote access is enabled within the devices connected in that network.





6. Network Operating System

- These systems run on a server and provide the capability to manage data, users, groups, security, applications, and other networking functions. These types of operating systems allow shared access to files, printers, security, applications, and other networking functions over a small private network. One more important aspect of Network Operating Systems is that all the users are well aware of the underlying configuration, of all other users within the network, their individual connections, etc. and that's why these computers are popularly known as tightly coupled systems .



- **7. Real-Time Operating System**
- These types of OSs serve real-time systems. The time interval required to process and respond to inputs is very small. This time interval is called **response time.Real-time systems** are used when there are time requirements that are very strict like missile systems, air traffic control systems, robots, etc.

- **Types of Real-Time Operating Systems**

- **Hard Real-Time Systems:** Hard Real-Time OSs are meant for applications where time constraints are very strict and even the shortest possible delay is not acceptable. These systems are built for saving life like automatic parachutes or airbags which are required to be readily available in case of an accident. Virtual memory is rarely found in these systems.

- **Soft Real-Time Systems:** These OSs are for applications where time-constraint is less strict.

- **8. Mobile Operating Systems**
- These operating systems are mainly for mobile devices. Examples of such operating systems are Android and iOS.

- **Hardware-Dependent OS:**
- A **hardware-dependent OS** is an operating system that is tightly coupled to the underlying hardware. It is designed to work with specific hardware architectures, and its functions or capabilities are often limited to the hardware features provided by that specific system.
- **Example:** Early versions of **MS-DOS** or OSs like **Apple DOS** were hardware-dependent because they were tailored to run on specific hardware platforms (like IBM PCs or Apple IIs).
- **Characteristics:**
    - Direct interaction with hardware components.
    - Requires hardware-specific drivers.
    - Cannot easily be ported to different hardware without significant modifications.
    - Often optimized for a specific processor, memory architecture, or input/output devices.

Hardware-Independent OS:

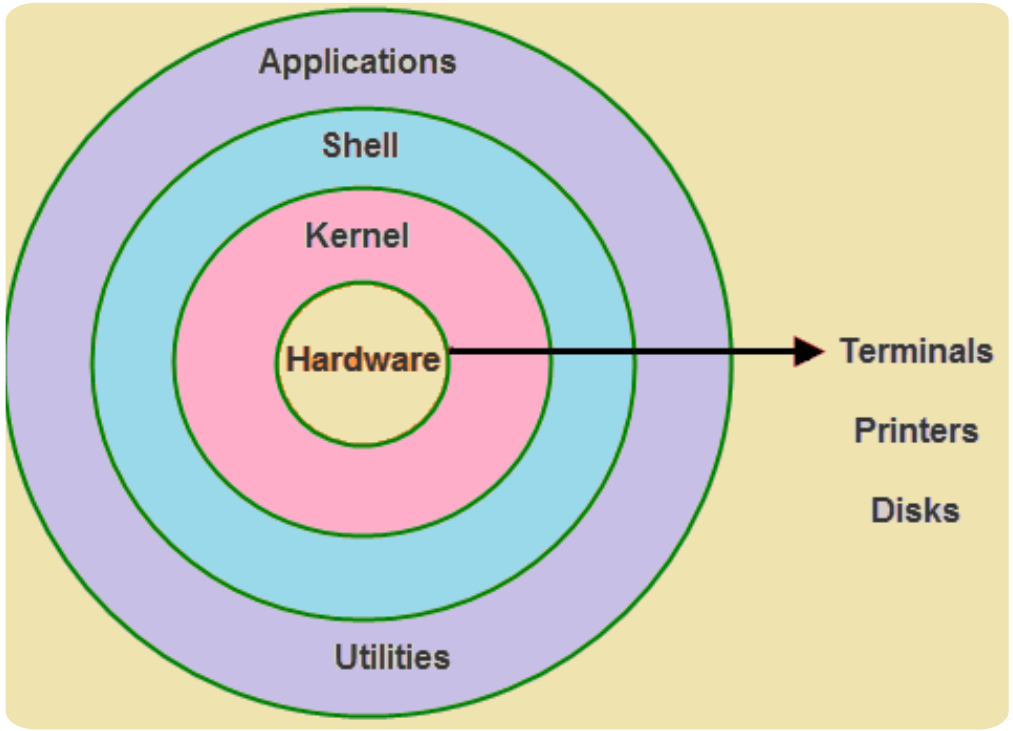
A **hardware-independent OS** is designed to work across different hardware platforms without needing significant changes. This type of OS uses abstracted hardware interfaces, so it can function on a wide variety of hardware, as long as there are appropriate drivers or hardware interfaces available.

•**Example: Linux** and **Windows** are examples of hardware-independent operating systems because they can run on various hardware platforms, including x86, ARM, and others.

•**Characteristics:**

- Hardware abstraction layers allow it to work across different systems.
- The OS itself doesn't need to be modified when moving from one hardware platform to another.
- More flexible and adaptable, with broad hardware support.
- Uses drivers or modules to interface with different hardware.

- **Important Components of the Operating System**
- Process Management
- File Management
- Command Interpreter
- System Calls
- Signals
- Network Management
- Security Management
- I/O Device Management
- Secondary Storage Management
- Main Memory Management



- Two critical components of an operating system are the kernel and the shell. Understanding the relationship between these two components is fundamental to grasping how operating systems function and how users interact with their computers.
- **What is a Shell?**
- The shell is the Linux command line interpreter. It provides an interface between the user and the kernel and executes programs called commands. For example, if a user enters ls then the shell executes the ls command . It acts as an intermediary between the user and the kernel, interpreting commands entered by the user and translating them into instructions that the kernel can execute. The shell also provides various features like command history, tab completion, and scripting capabilities to make it easier for the user to work with the system.

Advantages

•Efficient Command Execution

•Scripting capability

Disadvantages

•Limited Visualization

•Steep Learning Curve

- **What is Kernel?**
- The kernel is the core component of the operating system that manages system resources and provides services to other programs running on the system. It acts as a bridge between the user and the resources of the system by accessing various computer resources like the CPU, I/O devices and other resources. It is responsible for tasks such as memory management, process scheduling, and device drivers. The kernel operates at a lower level than the shell and interacts directly with the hardware of the computer.

- **Advantages**

- Efficient Resource Management
- Process Management
- Hardware Abstraction

Disadvantages

•Limited Flexibility

•Dependency on Hardware

Difference Between Shell and Kernel

Shell allows the users to communicate with the kernel.	Kernel controls all the tasks of the system.
It is the interface between kernel and user.	It is the core of the operating system.
It is a command line interpreter (CLI).	Its a low level program interfacing with the hardware (CPU, RAM, disks) on top of which applications are running.
Its types are – Bourne Shell, C shell, Korn Shell, etc.	Its types are – Monolithic Kernel, Micro kernel, Hybrid kernel, etc.
It carries out commands on a group of files by specifying a pattern to match	It performs memory management.
Shell commands like ls, mkdir and many more can be used to request to complete the specific operation to the OS.	It performs process management.
It is the outer layer of OS.	It is the inner layer of OS.
It interacts with user and interprets to machine understandable language.	Kernel directly interacts with the hardware by accepting machine understandable language from the shell.
Command-line interface that allows user interaction	Core component of the operating system that manages system resources
Interprets and translates user commands	Provides services to other programs running on the system
Acts as an intermediary between the user and the kernel	Operates at a lower level than the shell and interacts with hardware
Provides various features like command history, tab completion, and scripting capabilities	Responsible for tasks such as memory management, process scheduling, and device drivers
Executes commands and programs	Enables user and applications to interact with hardware resources

Types of Shells in Linux:

The C Shell –

Denoted as **csh**

•Bill Joy created it at the University of California at Berkeley. It incorporated features such as aliases and command history. It includes helpful programming features like built-in arithmetic and C-like expression syntax. In C shell:

Command full-path name is /bin/csh, Non-root user default prompt is hostname %, Root user default prompt is hostname #.

The Bourne Shell –

Denoted as **sh**

•It was written by Steve Bourne at AT&T Bell Labs. It is the original UNIX shell. It is faster and more preferred. It lacks features for interactive use like the ability to recall previous commands. It also lacks built-in arithmetic and logical expression handling. It is default shell for Solaris OS. For the Bourne shell the:

Command full-path name is /bin/sh and /sbin/sh, Non-root user default prompt is $, Root user default prompt is #.

The Korn Shell

It is denoted as **ksh**

•It was written by David Korn at AT&T Bell Labs. It is a superset of the Bourne shell. So it supports everything in the Bourne shell.It has interactive features. It includes features like built-in arithmetic and C-like arrays, functions, and string-manipulation facilities. It is faster than C shell. It is compatible with script written for C shell. For the Korn shell the:

Command full-path name is /bin/ksh, Non-root user default prompt is $, Root user default prompt is #.

GNU Bourne-Again Shell –

Denoted as **bash**

•It is compatible to the Bourne shell. It includes features from Korn and Bourne shell. For the GNU Bourne-Again shell the:

Command full-path name is /bin/bash, Default prompt for a non-root user is bash-g.gg$ (g.ggindicates the shell version number like bash-3.50$), Root user default prompt is bash-g.gg#.

T Shell –

Denoted as **tsh**

•It was originally developed for the Plan 9 operating system, but has since been ported to other systems, including Linux, FreeBSD, and macOS.

Command full-path name is /bin/tcsh, Default prompt for a non-root user is abhishekaslk(user):~> Root user default prompt is root@abhishekaslk(user):~#.

Z Shell –

Denoted by **zsh**

•Z Shell (zsh) was created by Paul Falstad in 1990 while he was a student at Princeton University. Z Shell is an extended version of the Bourne-Again Shell (bash), with additional features and capabilities.

Command full-path name is /bin/zsh, Default prompt for a non-root user is abhishekaslk(user)% Root user default prompt is root@abhishekaslk(user):~#