**Operating system**

It is defined as a **Resource Allocator & Control Program** making efficient use of hardware and managing execution of user programs.Operating systems are intermediaries between users and computer hardware. They perform essential functions from managing computer hardware to running applications to enabling users to communicate with computers through a user-friendly interface. It is especially concerned with the operation and control of I/O devices.

**Computer Peripheral**

A peripheral device, or simply peripheral, is an auxiliary (additional) hardware device that a computer uses to transfer information externally. A peripheral is a hardware component that is accessible to and controlled by a computer but is not a core component of the computer. Printers, speakers, and microphones are examples of computer peripherals.

**Computer System Structure**

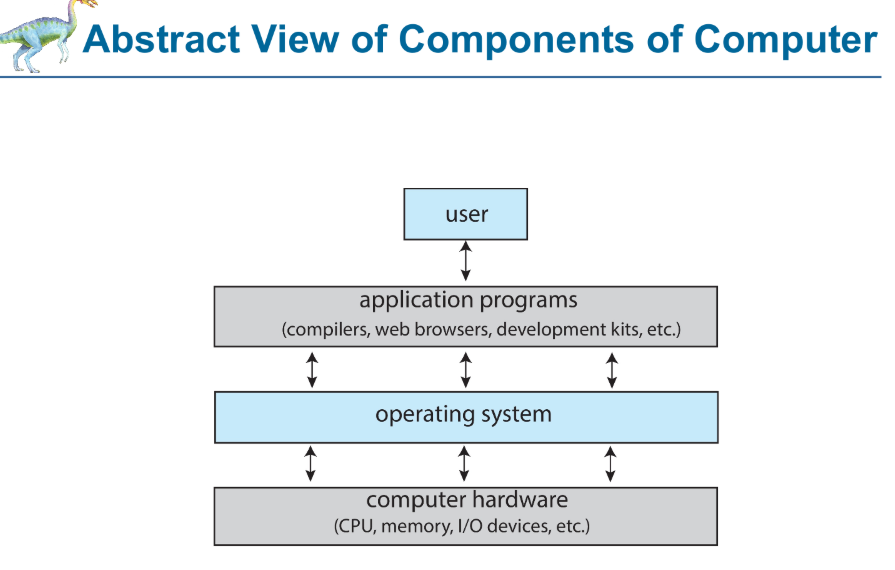
Computer system can be divided into four components:

**Hardware** **–** Provides basic computing resources CPU, memory, I/O devices

**Operating system –** Controls and coordinates use of hardware among various applications and users

**Application programs –** Define the ways in which the system resources are used to solve the computing problems of the users i.e. Word processors, web browsers, video games

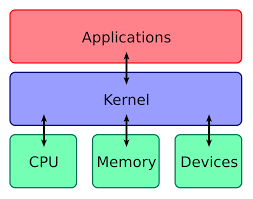
**Users –** People, machines, other computers

****

**Basic Operating System Definition/Terms**

**1. Kernel**

A kernel is a computer program that run all the time on computer. It is the central component (heart) of an operating system (OS). It acts as a bridge between software applications and the computer's hardware, and manages resources like CPU, memory, and devices.



The kernel is the only part of the operating system that has direct access to the hardware. Both system programs and application programs rely on the kernel to perform tasks like input/output operations, process scheduling, and memory management.

**Objectives of Kernel**

* To establish communication between user-level applications and hardware.
* To decide the state of incoming processes.
* To control disk management.
* To control memory management.
* To control task management.

**How does the kernel work?**

* During startup, the BIOS loads the kernel into a protected memory space
* The kernel then loads other OS components to complete the startup
* The kernel provides services to programs, which then request access to hardware
* The kernel forwards the request to device drivers, which control the hardware

**Why is the kernel important?**

* The kernel is the most fundamental part of the OS
* It's responsible for controlling everything that happens on the computer
* It's responsible for security

**Functions of Kernel**

The kernel is responsible for various critical functions that ensure the smooth operation of the computer system. These functions include:

**1. Process Management**

* Scheduling and execution of processes.
* Context switching between processes.
* Process creation and termination.

**2. Memory Management**

* Allocation and deallocation of memory space.
* Managing virtual memory.
* Handling memory protection and sharing.

**3. Device Management**

* Managing input/output devices.
* Providing a unified interface for hardware devices.
* Handling device driver communication.

**4. File System Management**

* Managing file operations and storage.
* Handling file system mounting and unmounting.
* Providing a file system interface to applications.

**5. Resource Management**

* Managing system resources (CPU time, disk space, network bandwidth)
* Allocating and deallocating resources as needed
* Monitoring resource usage and enforcing resource limits

**6. Security and Access Control**

* Enforcing access control policies.
* Managing user permissions and authentication.
* Ensuring system security and integrity.

**7. Inter-Process Communication**

* Facilitating communication between processes.
* Providing mechanisms like message passing and shared memory.

**8. Multitasking Management**

* Sets interrupts for the CPU to enable multitasking

**2. System Program**

System programs are software tools and utilities associated with the operating system, though they are not part of the kernel itself.

System programs are helpers that use the kernel to manage hardware and support other software, but they do not have direct hardware access themselves.

A system program is a type of software designed to help control hardware and provide a platform for running application programs. These programs include operating systems, file managers, device drivers, command line, compilers, and system utilities.

**3. Application Program**

All programs not associated with the operating system. They are designed to help users perform specific tasks such as word processing, browsing the internet, gaming, or managing spreadsheets.

These programs leverage the operating system’s services (provided via the kernel and system programs) to execute their functions.

**4. Middleware**

Middleware is software that lies between an operating system and the applications running on it.

Middleware facilitates communication, data management, and service integration between different software applications.

It is a set of software frameworks that provide additional services to application developers such as databases, multimedia, graphics libraries, and application servers.

**Need for Operating System**

**Platform for Applications:**  
The operating system (OS) acts as a foundation on which application programs run. It provides the necessary interface between the computer hardware and the user, making it easier for users to perform tasks using various software applications.

**Managing Input-Output Units:**  
The OS controls and manages computer resources like the monitor, keyboard, printer, and more. It efficiently allocates these resources to different programs based on their needs.

**Multitasking:**  
By managing memory and scheduling tasks, the OS allows multiple programs to run simultaneously. This multitasking capability improves user experience as different tasks can be performed at the same time.

**Platform for Other Software Applications:**  
The OS manages and coordinates various applications, ensuring they run effectively and interact with the user and each other smoothly.

**Memory Control:**  
The OS controls the computer’s main memory, allocating and deallocating memory for different tasks and applications as needed.

**System File Management:**  
All data on a computer is stored as files. The OS helps organize, manage, and provide easy access to these files, making file interaction straightforward.

**Security:**  
The OS plays a crucial role in maintaining system security by managing user permissions and ensuring that only authorized access is allowed, thus protecting both the system and its applications.

**Without Operating System**

Without an operating system, using a computer would be much more complex because there would be no central software to manage everything.

* **Direct Hardware Access:**  
  Every program would have to control hardware devices (like the keyboard, screen, and printer) on its own. This means each program must write code specifically for each device.
* **No Multitasking:**  
  There would be no built-in way to allocate CPU time or memory among multiple programs. Each program would run one at a time because there’s no OS to switch between them or manage their memory.
* **No Standard Interfaces:**  
  Operating systems provide standardized interfaces (APIs) that allow application programs to interact with the hardware in a consistent way. There wouldn’t be a common way for programs to interact with hardware. Each program would use its own methods, making it hard for software to work together or be reused.
* **Manual File Management:**  
  Saving, reading, or organizing files would be very difficult because there wouldn’t be a built-in system (like a file manager) to handle files and folders.

**Functions of an Operating System**

An operating system has a variety of functions to perform. Some of the prominent functions of an operating system can be broadly outlined:

**1. Processor Management:** This deals with the management of the Central Processing Unit (CPU). The operating system takes care of the allotment of CPU time to different processes. When a process finishes its CPU processing after executing for the allotted time period, this is called scheduling. There is various type of scheduling techniques that are used by the operating systems:

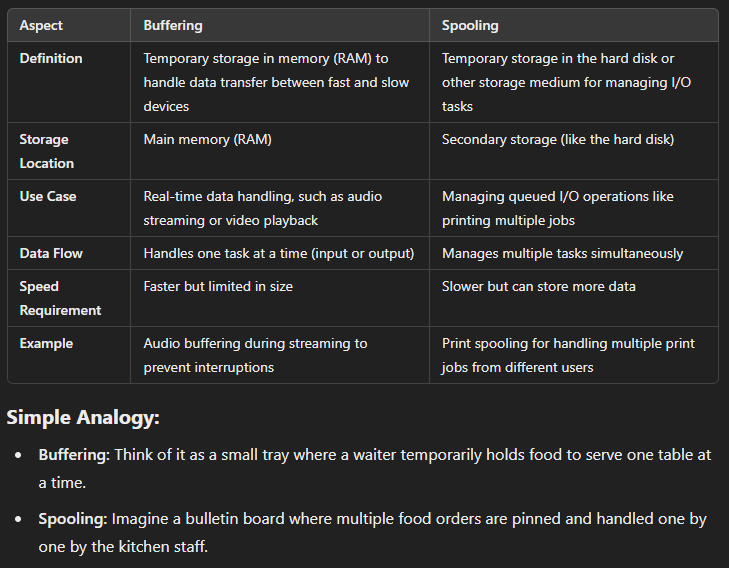
* **Shortest Job First(SJF):** The process which needs the shortest CPU time is scheduled first.
* **Round Robin Scheduling:** Each process is assigned a fixed CPU execution time in a cyclic way until the process is completed.
* **Priority-Based Scheduling (Non-Preemptive):** In this scheduling, processes are scheduled according to their priorities, i.e., the highest priority process is scheduled first. If the priorities of the two processes match, then schedule according to arrival time. **Non-preemptive** means a process cannot be interrupted by another process while it is running.

**2. Buffering:** A temporary storage area, called a buffer, which is a **small, reserved area of memory**, is used to hold data being transferred between a computer's memory and an **input or output device**, and to manage the flow of data between fast and slow devices or processes. Data is collected in chunks (in buffer) and transferred in batches (to CPU) instead of one small piece at a time.

* When data is read from an input device (like a keyboard), it is first stored in a buffer in **non-volatile memory** **(RAM)** before being processed by the CPU. Similarly, when data is sent to an output device (like a printer), it is written to a buffer first, allowing the CPU to continue processing while the output device catches up.
* It helps in matching speed between fast and slow devices or processes.
* Data from a fast source is temporarily placed in a buffe. The receiving device reads data from the buffer at its pace. When the buffer empties or fills up, the process continues.
* **Printers:** Buffer stores print data so the printer can print at its own speed while the computer moves on to other tasks.
* Its primary role is to smooth out data transfers between the CPU and slower I/O devices (like disks, printers, or network interfaces). Other roles include:
* **Network Communications:** Buffers temporarily hold data being sent or received over a network, helping manage variations in transmission speeds and preventing data loss.
* **Streaming Media:** When streaming audio or video, buffers store portions of the media to ensure smooth playback despite fluctuations in data delivery.
* **Inter-process Communication:** Buffers can be used to hold data exchanged between different processes or threads, ensuring that the data is available even if one side is processing it slower than the other.
* **Software Applications:** Many programs use buffers to temporarily store user input, intermediate calculation results, or data read from files before processing.

**3. Spooling:** It refers to a technique where data is temporarily **queued** in a buffer (typically on the hard disk) before being sent to a slow **input/output (I/O) device**, such as a printer. Spooling makes use of the **hard disk (volatile memory)** as a **large buffer**.

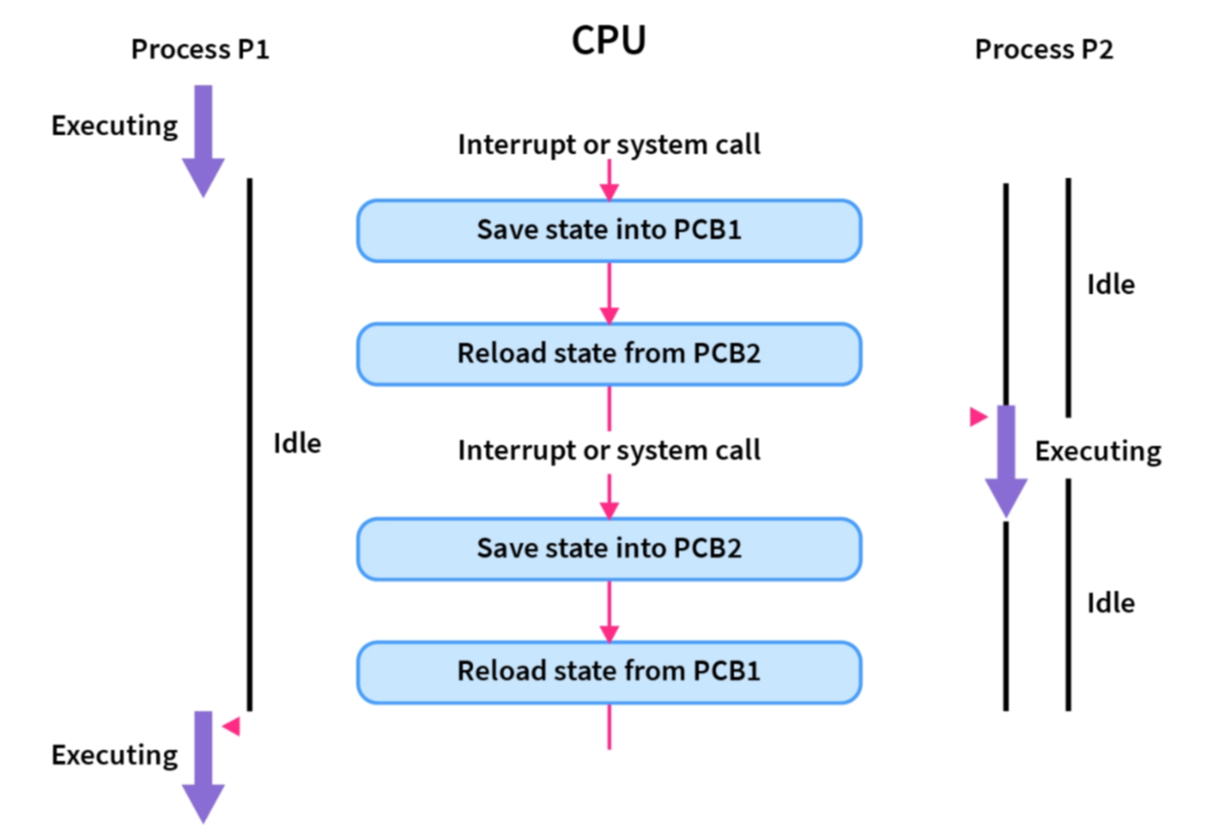
* When multiple tasks need access to the same device (like printing several documents), the operating system saves these tasks in a **queue (FIFO: First In First Out)** on the hard disk.
* The device processes these tasks one by one at its speed while the system continues with other operations.
* Its primary function is to prevent two users from printing on the same page at the same time, resulting in their output being completely mixed. It prevents this because it uses the **FIFO** strategy.



**4. Context/Process Switching:** In most multitasking OSs, context switching is the process of saving the state (or "context") of a currently running process and loading the state of another process. This enables the CPU to switch between multiple processes. This switching happens so fast that we think multiple programs are running at the same time (multitasking), however, CPU only executes one process at a time. It is necessary as the CPU has to continuously execute tasks rather than staying idle, so it must save the current state and load another one.

**It Involves:**

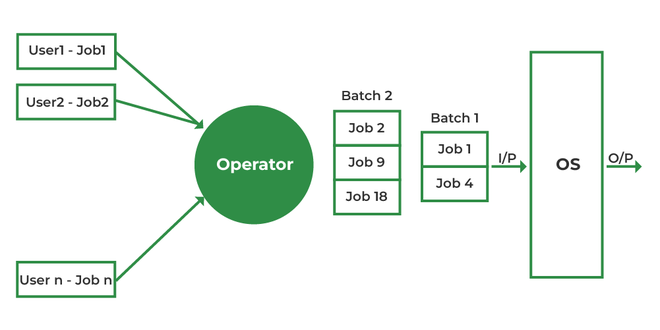
* The operating system saves the state of the current thread
* The operating system loads the state of another thread
* This involves saving and restoring registers, stack pointers, and other context information



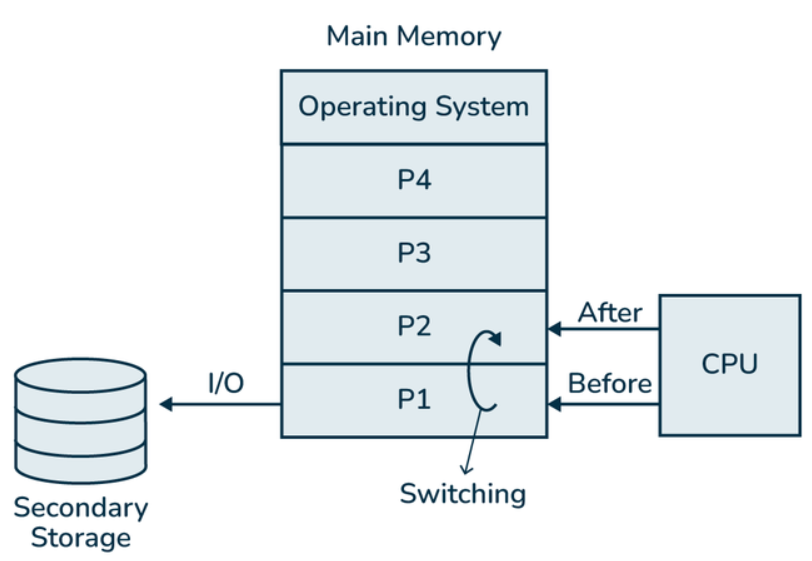
**Context switching overhead** is the delay that occurs when an operating system switches from one thread to another. It's a cost to the system's performance.

**Types of Operating Systems**

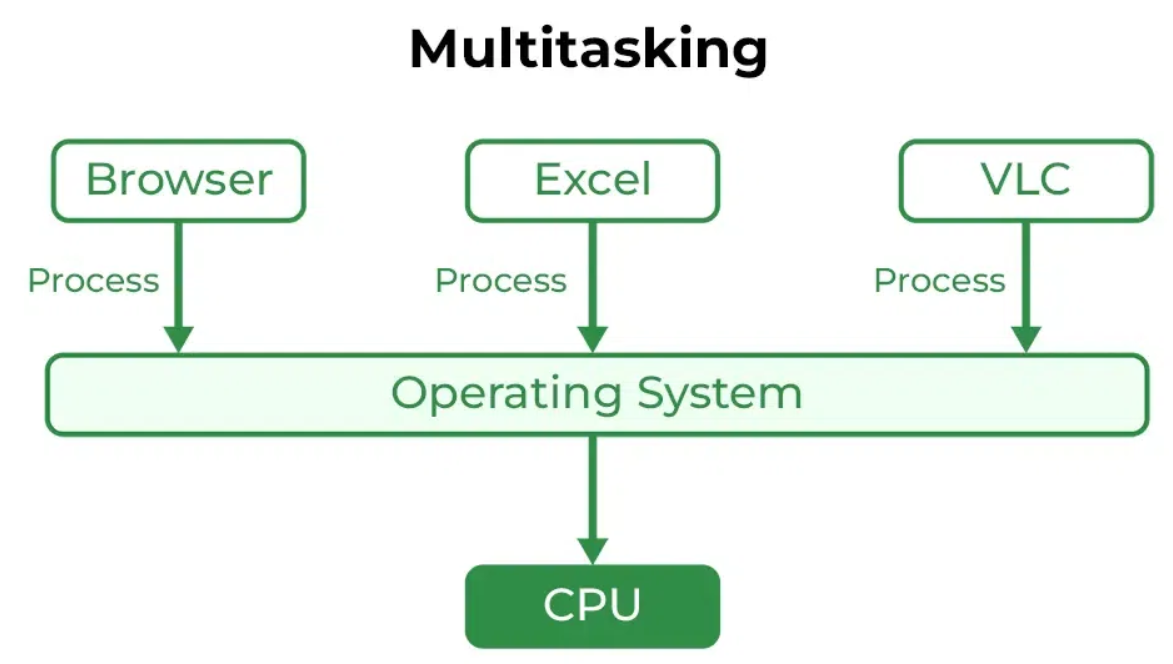
* **Simple batch operating system**
  + It contains a command line interface, and all processes are processed one by one. Users do not interact with the computer directly. Each user prepares their job using an offline device called a punch card and submits it to the computer operator. He then sorts the jobs with similar needs into batches and executes them. i.e. monthly bank statement.



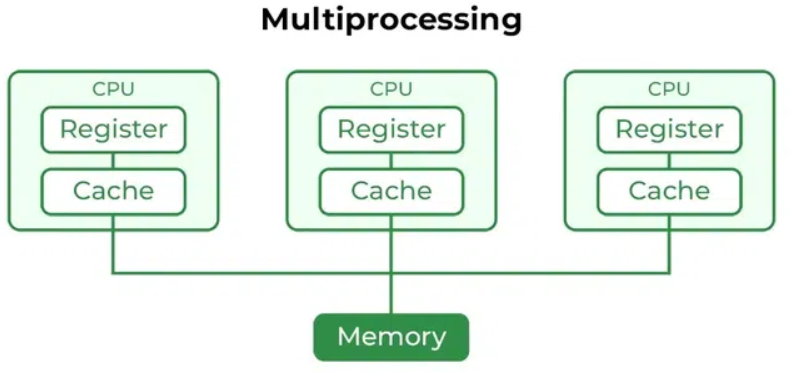
* **Multi-programming batch operating system**
  + It is a type of operating system designed to concurrently load multiple programs in the **main memory** and any one of them can be kept in execution at all times. This allows the CPU to switch between jobs **(Context Switching)** so that if one job is waiting for an input/output (I/O) operation, the CPU can work on another job. This maximizes the use of CPU and reduces idle time. It utilizes **both buffering and spooling** to handle its **I/O operations** efficiently.



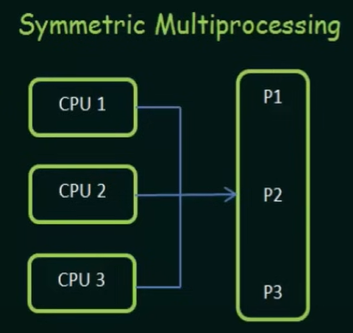
* **Multi-tasking/Time-Sharing operating system**
  + It is a type of Multiprogramming system with every process running in **round robin** manner.
  + **Time-Sharing:** The OS divides CPU time among the various tasks, allotting a small time slice to each process. This ensures that no single process monopolizes the CPU and so the OS manages and executes multiple processes at once by rapidly switching between them based on the time slice, which creates the illusion that they are running simultaneously even on a single-core processor.
  + Time-sharing systems allow multiple users to share hardware resources such as the CPU, memory, and peripherals, reducing the cost of hardware and increasing efficiency.



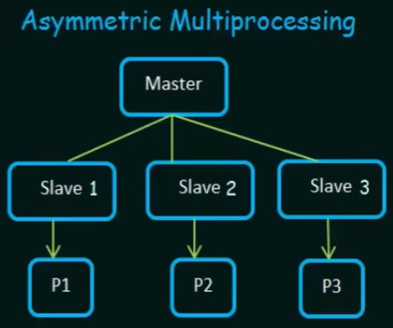
* **Multi-processing operating system**
  + It is a type of Operating System in which more than one CPU is used to run multiple processes concurrently. It increases the throughput (the number of jobs processed per time unit) of the System as processes can be parallelized, which leads to significant performance gains.
  + **Load Balancing:** The OS schedules tasks across the available processors to ensure an even distribution of work. This load balancing helps avoid overloading one processor while others remain idle.



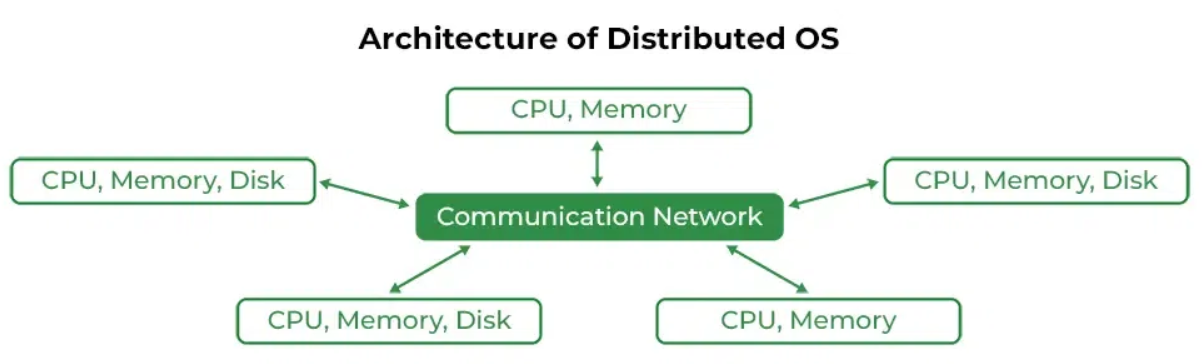
* **Symmetric Multiprocessing (SMP):**  
  All processors are treated equally. Each one can execute any task, and they share a common memory and I/O bus. Most modern systems (like Windows, Linux, and macOS) use SMP.



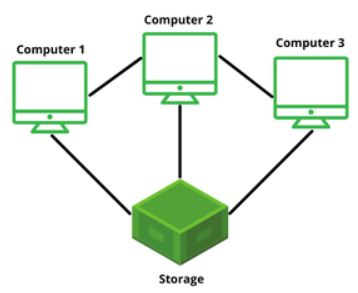
* **Asymmetric Multiprocessing (AMP):**  
  Processors are assigned specific tasks. One processor (the master) may handle the main operating system functions, while others (the slaves) perform specific tasks. This approach is less common in general-purpose systems.



* **Real-time operating system**
  + It is a type of OS used when 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.
* **Distributed operating system**
* The OS allows multiple computers to share hardware and software resources such as memory, processing power, files, and peripheral devices, making the most of the available assets. It is specifically made to manage resources across a collection of independent computers, which can be geographically dispersed.
  + A key component of a distributed OS is the ability to communicate over a network. Efficient messaging, remote procedure calls, or shared memory techniques are used to coordinate tasks between the different computers.
  + Users and applications interact with the distributed system without needing to know which machine is performing a particular task.
  + Distributed operating systems can scale by adding more computers to the network.
  + Because tasks and data are distributed among several machines, the system can continue operating even if one or more machines fail.
  + Distributed systems are commonly seen in cloud computing, where data centers comprise many interconnected servers that work together.

****

* **Clustered operating system (shared-disk system)**
  + A group of computers connected in a local area network is called a Cluster System. A Clustered Operating System manages a group of independent computers (or nodes) that work together as a single unified system, and the workload is spread across several nodes (Load Balancing).
  + It involves a group of tightly coupled computers (nodes) that are usually located in the same physical area. Built specifically to ensure that if one node fails, others can take over seamlessly, minimizing downtime.
  + Basically, they have independent computers but have a common storage and the systems work together.
  + Clustered systems can be expanded by adding more nodes.
  + **Hardware Cluster** helps in enable high-performance disk sharing between systems, while the **Software Cluster** allows all systems to work together.



* **Embedded operating system**
  + An embedded operating system is a specialized operating system (OS) designed to perform a specific task for a device that is not a traditional computer.
  + Embedded OSes are built to perform specific tasks. For example, the operating system in a digital camera or a washing machine is designed solely to manage the functions of that device.
  + They are optimized for low memory usage, limited processing power, and minimal storage, as embedded devices often have these constraints.
* **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.
* **Multi-User Operating Systems**
  + These systems allow multiple users to be active at the same time. This system can be either multiprocessor or single processor with interleaving (alternating or mixing operations).

