Operating System

An operating system (OS) is a software program that manages computer hardware and software resources and provides common services for computer programs. It is the foundation upon which other software applications and user interactions are built.

The primary purpose of an operating system is to facilitate communication and coordination between hardware devices and software applications. It acts as an intermediary between the computer's hardware (such as the processor, memory, storage, and input/output devices) and the software programs running on it.

Functions of OS

Here are some key functions of an operating system :

- **Process management:** The OS manages and schedules processes (or tasks) running on the computer, allocating system resources and ensuring that each process gets its fair share of the CPU time.

- **Memory management:** It handles the allocation and deallocation of memory resources for different processes and manages the virtual memory system.

- **File system management:** The OS provides a hierarchical structure for organizing and storing files and directories, as well as managing file access, permissions, and storage devices.

- **Device management:** It interacts with input and output devices (e.g., keyboards, mice, printers, and disks) by providing device drivers, which act as translators between the hardware and software.

- **User interface:** The OS provides a user interface through which users interact with the computer system. This can be in the form of a command-line interface (CLI) or a graphical user interface (GUI).

- **Security and protection:** The OS enforces security measures to protect the system and user data from unauthorized access, viruses, and other threats.

Examples of popular operating systems include Microsoft Windows, macOS, Linux, and Android. Each operating system has its own features, design principles, and target devices, but they all serve as the foundation for running software applications and managing computer resources efficiently.

Need of operating system

There are several reasons why an operating system is necessary for a computer system. Here are some key reasons:

- **Hardware abstraction:** The operating system provides a layer of abstraction between the hardware components of a computer and the software applications. It hides the complexities of the underlying hardware, allowing software developers to write programs without having to worry about the specific details of each hardware device. This abstraction makes it easier to develop software that can run on different hardware configurations.

- **Resource management:** An operating system manages computer resources such as the central processing unit (CPU), memory, storage devices, and input/output devices. It allocates these resources to different software programs and ensures that they are used efficiently and fairly. Without an operating system, software applications would need to directly manage and compete for system resources, which would be highly inefficient and prone to conflicts.

- **Process scheduling:** The operating system is responsible for scheduling and managing processes (or tasks) running on the computer. It decides which process gets to use the CPU and for how long, ensuring that all running programs get their fair share of processing time. This scheduling allows for multitasking, where multiple programs can run concurrently and appear to be executing simultaneously.

- **Memory management:** The operating system handles the allocation and deallocation of memory resources for software programs. It provides virtual memory, which allows programs to use more memory than physically available by storing parts of the program in secondary storage (such as the hard disk) when not immediately needed. The OS also ensures that programs do not interfere with each other's memory space, preventing crashes and data corruption.

- **File system management:** Operating systems provide a file system that organizes and stores files and directories on storage devices. It handles file creation, deletion, and access permissions, and provides a hierarchical structure for organizing data. The file system allows users and programs to store, retrieve, and manage data efficiently.

- **User interface:** The operating system provides a user interface that allows users to interact with the computer system. This interface can be a command-line interface (CLI) or a graphical user interface (GUI) with icons, menus, and windows. The user interface makes it easier for users to run applications, manage files, and perform various tasks on the computer.

- **Security and protection:** Operating systems incorporate security measures to protect the computer system and user data from unauthorized access, viruses, and other threats. They enforce user authentication, access control, and encryption mechanisms to safeguard data and ensure system integrity.

Overall, an operating system plays a crucial role in managing computer resources, providing a user-friendly interface, and ensuring the efficient and secure operation of a computer system. It acts as the intermediary between hardware and software, enabling the execution of various applications and facilitating user interactions with the system.

Types of OS

Here are the types of operating systems you mentioned :

1. **Batch Operating System:** A batch operating system is designed to execute jobs or tasks in batches without user interaction. It processes a series of similar jobs without human intervention, such as running a sequence of commands or programs.

2. **Multiprogrammed Operating System:** A multiprogrammed operating system allows multiple programs to reside in the main memory simultaneously. It enables efficient utilization of the CPU by rapidly switching between programs, providing the illusion of simultaneous execution.

4. **Multitasking Operating System:** A multitasking operating system allows multiple tasks or processes to run concurrently. It provides the ability to execute multiple programs or tasks simultaneously, allowing users to switch between them quickly and share system resources.

5. **Real-Time Operating System (RTOS):** A real-time operating system (RTOS) is an operating system designed to serve real-time applications, which require precise and predictable timing and response characteristics. It is primarily used in systems where timely and deterministic execution of tasks is crucial.

RTOS Types

Hard Real-Time Operating System:

A hard real-time operating system guarantees that tasks are executed within their specified time constraints. It prioritizes time-critical tasks and ensures that they are completed on time, even if it means delaying or aborting lower-priority tasks. Hard real-time systems are commonly found in critical applications such as aerospace, industrial automation, medical devices, and automotive systems. Examples of hard real-time operating systems include VxWorks, QNX, and FreeRTOS.

Soft Real-Time Operating System:

A soft real-time operating system aims to provide timely execution of tasks, but it does not provide absolute guarantees on meeting every deadline. Soft real-time systems prioritize tasks based on their urgency and attempt to meet the deadlines as much as possible. If a deadline is missed, the system can still recover and execute the task, but it might impact system performance. Soft real-time operating systems are used in applications such as multimedia streaming, telecommunication systems, and video games. Examples of soft real-time operating systems include Linux with real-time extensions (such as RTAI and Xenomai) and Windows Embedded Compact.

Clustered Operating System: A clustered operating system is designed to provide high availability and fault tolerance by combining multiple computers, called nodes, into a cluster. Each node in the cluster has its own processor, memory, and storage. However, they work together as a single system to perform tasks and provide services. Think of it like a team of people working together on a project. Each person has their own skills and resources, but they collaborate and coordinate their efforts to achieve a common goal. Similarly, in a clustered operating system, each node contributes its resources to handle tasks and ensure that if one node fails, another can take over seamlessly.

Distributed Operating System: A distributed operating system, on the other hand, extends the concept of a clustered operating system by connecting computers that are geographically dispersed into a network. These computers, often called nodes or hosts, communicate and coordinate with each other to provide a unified computing environment. Imagine a group of friends spread across different locations, but they still want to collaborate and work together. They can communicate through phone calls, video chats, or messaging apps to share information, divide tasks, and accomplish their objectives. In a distributed operating system, the computers act in a similar way, sharing resources, exchanging data, and coordinating their activities to provide a seamless and efficient computing environment.

Embedded Operating System: An embedded operating system, or embedded OS, is a specialized type of operating system that is designed to run on small, resource-constrained devices. Unlike traditional operating systems like Windows or macOS that run on desktop computers or laptops, embedded OS is used in a wide range of devices such as smartphones, smartwatches, digital cameras, home appliances, and even industrial machinery. In simple terms, an embedded OS is like the brain of a small device. It manages the device's resources and enables it to perform its intended tasks, whether it's making phone calls, taking pictures, or controlling the temperature of your home. It's specifically designed to work efficiently on these devices and make the most of their limited capabilities.

Multiprogramming & Multitasking

Multiprogramming	***Multitasking***
In a multiprogramming OS, multiple programs are loaded into the memory simultaneously.	Multitasking takes the idea of multiprogramming a step further by allowing multiple programs to run simultaneously.
The CPU executes one program at a time until it encounters an input/output operation that requires waiting.	In a multitasking OS, each program is assigned a small time slice or a portion of CPU time to execute.
While one program is waiting, the CPU switches to another program and executes it. This process continues, allowing multiple programs to make progress concurrently.	The CPU rapidly switches between the programs, giving the illusion that they are running concurrently.
The goal of multiprogramming is to maximize CPU utilization and keep the computer busy, even when individual programs have to wait for input/output operations.	The purpose of multitasking is to provide users with the ability to work on multiple tasks or programs simultaneously, improving overall system responsiveness.

In summary, multiprogramming focuses on maximizing CPU utilization by running multiple programs sequentially, while multitasking allows multiple programs to run concurrently, giving users the perception of simultaneous execution.

Process & Process Scheduling

In an operating system, a process refers to a program in execution. It is a unit of work that consists of a set of instructions, data, and resources needed to complete a task. Process scheduling, on the other hand, is the method by which the operating system manages the execution of multiple processes on a computer's CPU.

Process scheduling involves deciding which process to run at a given time, allocating CPU time to each process, and switching between processes efficiently. The goal is to maximize CPU utilization, minimize response time, and ensure fairness among processes.

Think of process scheduling as a traffic controller that determines the order in which processes get to use the CPU. It helps ensure that different tasks are executed fairly and that the computer operates smoothly, even with multiple programs running simultaneously.

Process States

In an operating system, process states represent the different stages a process goes through during its execution. These states help the operating system manage and control the execution of multiple processes efficiently. Here are the commonly recognized process states in an operating system :

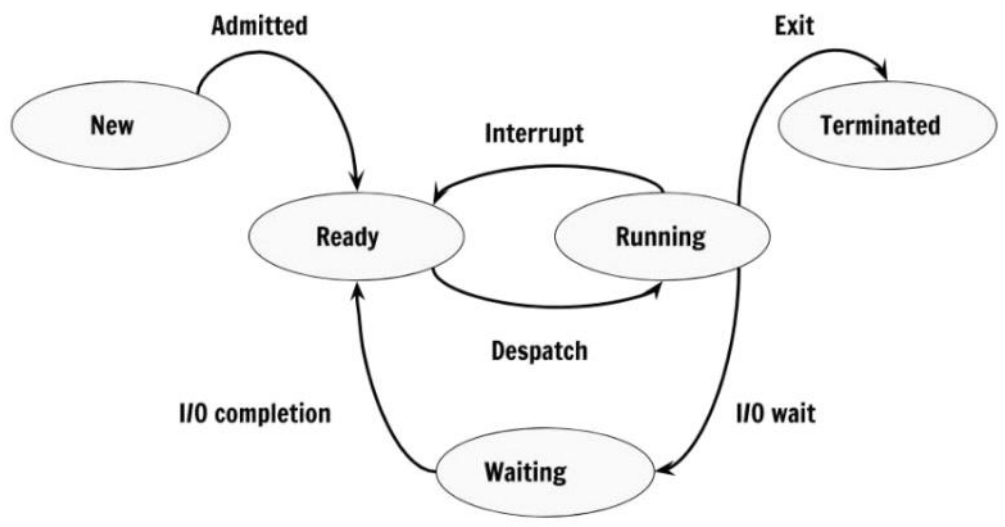
1. **New:** When a process is first created, it is in the new state. In this state, the process is being initialized, and the necessary resources are being allocated to it.

2. **Ready:** Once the initialization is complete, the process enters the ready state. In this state, the process is loaded into the main memory, and it is waiting to be assigned to a processor for execution. Multiple processes in the ready state are maintained in a queue or a similar data structure, and the operating system scheduler determines which process should be executed next.

3. **Running:** When a process is assigned to a processor, it enters the running state. In this state, the process's instructions are executed by the CPU, and it continues to consume CPU time until it either completes its execution or gets interrupted.

4. **Blocked (or Waiting):** A process enters the blocked state when it cannot proceed further without the occurrence of a specific event or the availability of a particular resource. For example, if a process needs to read data from the disk and the data is not currently available, it will enter the blocked state and wait for the data to be ready. While in the blocked state, the process does not consume CPU time and remains idle.

5. **Terminated (or Exit):** When a process finishes its execution, it enters the terminated state. In this state, the process releases any resources it was using, and its process control block (PCB) is removed from the system. The terminated process may still retain some information for the operating system, such as its exit status, which can be used by other processes.


Admitted
Exit
New
Terminated
Interrupt
Ready
Running
Despatch
I/O completion
I/O wait
Waiting