OPERATING SYSTEM:

DEFINITION:

It acts as an intermediary between the user and the computer hardware, enabling users to interact with the system EXAMPLE: WINDOWS,LINUX,MACOS,ANDROID,IOS

:

PURPOSE:

The primary purpose of an operating system is to provide an environment in which applications can run smoothly and utilize the computer's resources effectively

Can I switch from one operating system to another?

Yes, it is possible to switch from one operating system to another. However, it may require reinstalling the new operating system and ensuring compatibility with your hardware and software requirements.

Can I install multiple operating systems on a single computer?

Yes, it is possible to install multiple operating systems on a single computer using techniques such as dual booting or virtualization.

Different Types of Operating Systems

Different kinds of operating systems are available some of the important ones are –

- Batch System:

  Batch Operating system groups jobs that perform similar type of functions. These groups are treated as a batch and are executed at the same time without requiring constant user intervention.

Can batch operating systems handle real-time processing?

Batch systems are not designed for real-time processing, as they prioritize efficient execution of tasks in batches rather than immediate responsiveness.

Are batch operating systems still in use today?

Yes, batch operating systems are still utilized in specific industries that require large-scale data processing and efficient execution of repetitive tasks.

Characteristics and Features of Batch Operating Systems

- Job Sequencing: Batch operating systems execute tasks in a sequential order, known as job sequencing, without requiring constant user intervention.
- Minimal User Interaction: Batch systems minimize the need for user interaction by automating the execution of tasks, allowing users to submit jobs and let the system handle their processing.
- Queue-Based Processing: Batch systems typically employ a queue-based mechanism for job scheduling and resource allocation, ensuring efficient utilization of system resources.
- Priority-Based Execution: Jobs in a batch operating system can be assigned priority levels, allowing for the execution of higher-priority jobs before lower-priority ones.

- Distributed Operating System:

  A distributed operating system refers to an operating system that runs on a network of computers and enables them to work together as a single entity. The primary purpose of a distributed operating system is to achieve efficient resource sharing, fault tolerance, and scalability.

Component	**Purpose**	**Example**
Network Protocols	Communicate between machines	TCP/IP, UDP, RPC
Resource Management	Share CPU, memory, and files without conflict	Distributed memory
Distributed File Systems	Store/access files across machines	GFS, HDFS

Process Coordination	Ensure tasks run and interact correctly	Message passing, RPC

- Time Sharing System

Each user or process is given a fair and equal amount of time to execute their tasks. The time slices are typically very small, ranging from a few milliseconds to a few microseconds, and the context switch between tasks happens so quickly that it creates an illusion of concurrent execution.

Characteristics and Features of Time Sharing Operating Systems

- Interactive and Concurrent Access

One of the primary characteristics of a time sharing system is its ability to provide interactive and concurrent access to multiple users. Each user can initiate their tasks, execute commands, and receive prompt responses from the system. Users can work concurrently without significant delays or interruptions, enhancing productivity and collaboration.

- Fairness and Equal Time Allocation

  Fairness is a crucial aspect of a time sharing system. It ensures that each user or process receives an equal share of the computing resources. Time slices, also known as quanta, are allocated to each user or process, ensuring a fair distribution of CPU time. This fairness prevents any user or process from monopolizing the system resources and promotes equitable resource utilization.

- Responsiveness and Real-Time Interaction

  Time sharing systems prioritize responsiveness to provide users with a real-time interaction experience. Users can input commands and receive immediate feedback or results. The system quickly switches between tasks, giving the impression of simultaneous execution. This responsiveness creates a smooth and interactive user experience, contributing to user satisfaction and productivity.

- Process Scheduling

  Process scheduling is a critical feature of a time sharing system. It involves determining the order in which processes or tasks are executed and allocating time slices to each process. Scheduling algorithms, such as Round Robin, Priority Scheduling, and Multilevel Queue Scheduling, are employed to ensure fairness, optimize resource utilization, and meet specific system requirements.

- Context Switching

  Context switching is a key mechanism in a time sharing system. It enables the system to switch rapidly between different processes or tasks. During a context switch, the system saves the current execution context of a task and restores the context of the next task to be executed. Efficient context switching minimizes overhead and facilitates seamless transitions between tasks, contributing to overall system performance.

- Desktop System

A desktop system refers to a personal computer setup that is typically used on a desk or table. It consists of various hardware components and an operating system that enables users to perform a wide range of tasks such as document editing, web browsing, gaming, multimedia consumption, and more.

- Multiprocessor System

When two or more central processing units operate within a single computer system, it is referred as a multiprocessor system. In such computers, these multiple CPU's have a close connection and communication to share data and programs. These multiple processors also share operating system resources such as memory, buses, printers, and other peripheral devices.

Multiprocessor systems are used when users need extremely high processing speed to process a high volume of data. In most of the cases, such operating systems are used to carry out scientific calculations or operations such as satellite processing, deep data analysis, and weather forecasting.

Which operating systems provide support for multiprocessor systems?

Examples of operating systems that support multiprocessor systems include Linux, Windows Server, and Solaris.

- nhanced Performance and Throughput: Multiprocessor operating systems leverage parallelism to enhance system performance and increase throughput. By distributing tasks among multiple processors, these systems can execute multiple instructions simultaneously, leading to faster execution times and improved overall system responsiveness.
- Improved Reliability and Fault Tolerance: Multiprocessor systems provide built-in redundancy, which enhances system reliability and fault tolerance. If one processor fails, the remaining processors can continue to execute tasks without any disruption. This fault-tolerant capability is especially critical in mission-critical applications where system downtime is unacceptable.
- Scalability and Load Balancing:
  Multiprocessor operating systems offer excellent scalability by allowing additional processors to be added to the system as needed. This scalability enables organizations to accommodate increasing workloads without replacing the entire system. Furthermore, load balancing algorithms ensure that tasks are evenly distributed among processors, preventing bottlenecks and maximizing resource utilization.

- Clustered System

Feature	Clustered System	Distributed System
Locati	Same physical location	Geographically dispersed

on		
Connectivity	Tightly coupled	Loosely coupled
Shared Storage	Yes (typically shared storage like SAN/NAS)	Usually not shared (each node has its own)
Communication	High-speed, local communication (LAN)	Over a network (LAN/WAN/Internet)

Management	Managed as a single system	Each node is managed independently
Scalability	Limited (hardware constraints)	Highly scalable (add more machines easily)
Fault Tolerance	High (failover support)	Very high (redundancy and replication)
Example Use Cases	Load balancing, failover	Cloud computing, distributed databases, big data

Types of Clustered Systems

Clustered systems come in various forms, each tailored to specific requirements. Let’s delve into the three common types of clustered systems:

High-Availability Clusters High-availability clusters focus on providing continuous operation even in the face of hardware or software failures. They ensure that critical services remain accessible by automatically switching to a redundant node when a failure occurs. This type of clustering is widely used in mission-critical applications where downtime can have severe consequences.

Load-Balancing Clusters Load-balancing clusters distribute incoming workload evenly across multiple nodes, optimizing resource utilization and preventing bottlenecks. By intelligently allocating tasks, these clusters improve performance and handle traffic spikes gracefully. Load-balancing clusters are commonly employed in web servers, where high traffic volumes need to be handled efficiently.

Failover Clusters Failover clusters are designed to provide seamless failover in case of node failures. When a node becomes unavailable, another standby node takes over its responsibilities to ensure uninterrupted operation. Failover clusters are commonly used in database systems and critical applications that require uninterrupted service.

- Realtime Operating System

  A **Real-Time Operating System (RTOS)** is **not like your usual OS** (like Windows or Linux). It is specially designed for **time-sensitive** applications where responses must occur **within a strict deadline**.

Types of Real-Time Systems

Let’s go over each type one by one:

✅ 1. Hard Real-Time Systems

- **Definition:** Systems where **missing even one deadline** is **unacceptable** and could be **dangerous or catastrophic**.
- **Examples:**
  - 🚀 Spacecraft control
  - 🏥 Medical equipment (e.g., pacemakers)
  - ⚙️ Industrial robots

💥 If they're late even once — it can cause accidents!

🟡 2. Soft Real-Time Systems

- **Definition:** Deadlines are **important**, but missing them **occasionally** won't cause a disaster — just **reduced performance**.

- **Examples:**
    - 🎮 Video games
    - 📺 Video streaming
    - 🎤 Voice assistants

🔄 Slight delays are acceptable but still not ideal.

🟠 3. Firm Real-Time Systems

- **Definition:** These systems work within **timing constraints**, but **missing a few deadlines** may not break the system — though the **results might become useless**.
- **Examples:**
    - 📷 Visual inspection in automated factories

- 📹 Video conferencing

⚠️ Missing a deadline doesn't crash the system but can cause poor results.

🎯 What Makes RTOS Special Compared to Regular OS?

Feature	**General-Purpose OS (Windows/Linux)**	**Real-Time OS (RTOS)**
Priority	Fairness, multitasking	Timing & predictability
Response Time	Not guaranteed	Guaranteed & fast
Task Scheduling	Round-robin, priority	Deadline-driven

Used In	PCs, servers	Embedded systems, robotics
Example	Linux, Windows	FreeRTOS, VxWorks, RTEMS

- Handheld System: A **Handheld System** is a **small, portable computing device** that fits in your hand and typically runs on battery power. These systems are designed to be **lightweight**, **efficient**, and able to perform specific tasks.

CPU SCHEDULING:

- New: This is the initial state when a process is being created.
- Ready: The process is loaded into main memory and waiting to be assigned to a processor.
- Running: The process is being executed by a processor.
- Blocked: The process is unable to execute further until a certain event occurs, such as waiting for user input or completion of I/O operations.
- Terminated: The process has completed its execution or has been forcefully terminated.

A process can be in any of the following states –

- New state :Process it submitted to the process queue, it in turns acknowledges submission.
- Once submission is acknowledged, the process is given new status.

- Ready state
- It then goes to Ready State, at this moment the process is waiting to be assigned a processor by the OS

- Running state
- Once the Processor is assigned, the process is being executed and turns in Running State.

- Waiting state
- Terminated