Unit-1 Introduction to Operating System

Lecture – 5

Types of Operating System

Multi programming OS

- A Multiprogramming Operating System is designed to run multiple programs simultaneously by managing and sharing CPU time among them. It improves system efficiency by maximizing CPU utilization.
- The concept of multiprogramming is shown in the following figure.

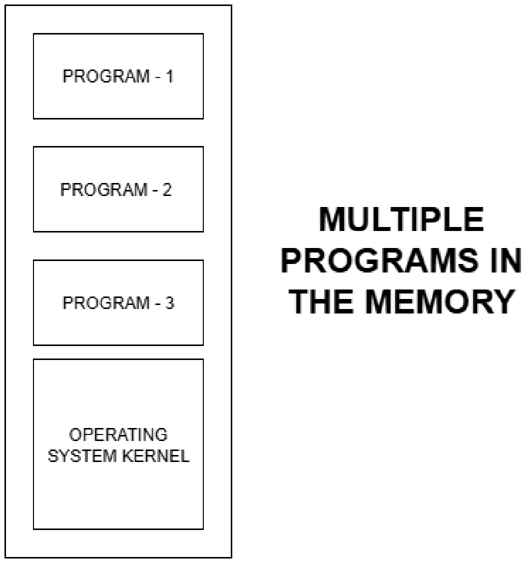


Figure – 1 The Concept of Multiprogramming

- **Key Features:**
    - Efficient CPU Utilization – The CPU never sits idle unless all processes are waiting.
    - Better Throughput – More jobs are completed in less time.
    - Job Scheduling – The OS selects which job to run next based on job scheduling algorithms.
    - Memory Management – The OS keeps multiple jobs in memory simultaneously and switches between them.

- **Example Scenario:**
    - Imagine a system running three programs:
        - Program A is doing calculations,
        - Program B is reading from a file,
        - Program C is waiting for user input.
    - If Program A finishes and B is still doing I/O, the CPU switches to Program C.
      So the CPU is always working, increasing performance.

- **Challenges:**
    - Memory management is complex because several jobs need to stay in memory.
    - CPU scheduling must be efficient to avoid starvation or deadlocks.
    - It also needs protection mechanisms to ensure one job doesn't affect another.

Time sharing OS

- A Time-Sharing Operating System allows multiple users or tasks to access a computer system simultaneously by sharing CPU time. It gives each user or processes a small time slice (called a time quantum) and rapidly switches between them, giving the illusion that all are running at the same time.
- It is also called a multitasking system.
- The concept of time sharing is shown in the following figure.

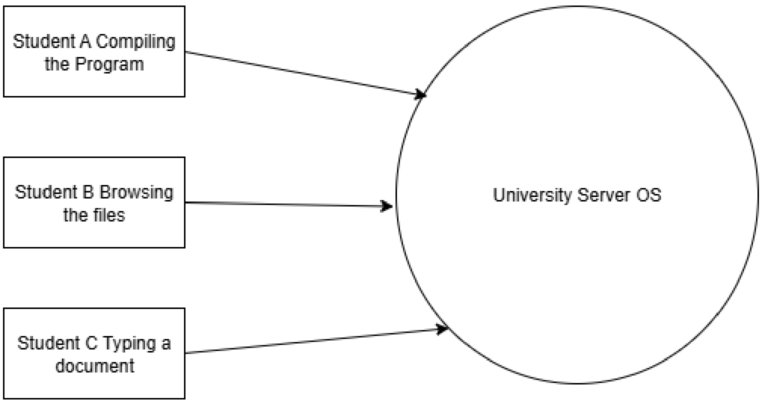


Figure – 1 The Concept of Time Sharing

- ***Key Features:***
    - **Interactive Processing** – Users can interact directly with the system in real time.
    - **Time Slicing** – Each process gets a small, fixed time to execute; the CPU switches rapidly between processes.
    - **Preemptive Scheduling** – If a process exceeds its time slice, the CPU preempts and moves to the next process.
    - **Multiple Users Supported** – Several users can access the system at once through different terminals.
    - **Efficient Resource Sharing** – CPU, memory, and I/O devices are shared efficiently among users.

- ***Example Scenario:***
    - Suppose multiple users are logged into a university server:
    - Student A is compiling a program,
    - Student B is browsing files,
    - Student C is typing a document.

  - The OS allocates a few milliseconds of CPU time to each user. Due to rapid switching, it appears that each student has exclusive access, but in reality, the CPU is being shared.

- ***Challenges:***
  - Response Time must be fast to maintain the feel of interactivity.
  - Process Isolation is essential so that one user's process doesn't interfere with another's.
  - Efficient Scheduling is required to avoid lag or unfair CPU allocation.
  - Security becomes critical in multi-user environments.

Real Time OS

- A Real-Time Operating System is designed to process data and respond to inputs or events within a guaranteed, fixed time frame.
- It is used where timely and predictable responses are critical — often in embedded systems, industrial control, robotics, missile control, etc.
- ***Key Features:***
  - **Deterministic Behavior** – Guarantees that critical tasks are completed within a defined time.
  - **Minimal Latency** – Very fast response to external events (in microseconds or milliseconds).
  - **Task Prioritization** – High-priority tasks preempt lower-priority ones.
  - **Resource Control** – Manages CPU, memory, and I/O devices tightly for consistency and speed.
- ***Types:***
  - **Hard real-time:**
    - Missing a deadline leads to failure. Timing is strict.
    - Used in: Pacemakers, Flight control systems, Rocket launching system.
  - **Soft real-time:**
    - Occasional deadline misses are tolerated, but performance degrades.
    - Used in: Multimedia streaming, Online games
  - **Firm real-time:**
    - Missing deadlines is acceptable but data becomes useless.
    - Used in: Banking transaction systems, Video surveillance
- ***Challenges:***
  - **Timing Guarantees** – Must consistently meet strict deadlines.
  - **Complex Scheduling** – Must balance CPU time across tasks with varying urgency.
  - **Limited Resources** – Often used in resource-constrained embedded systems.
  - **System Validation** – Must be tested thoroughly for reliability.

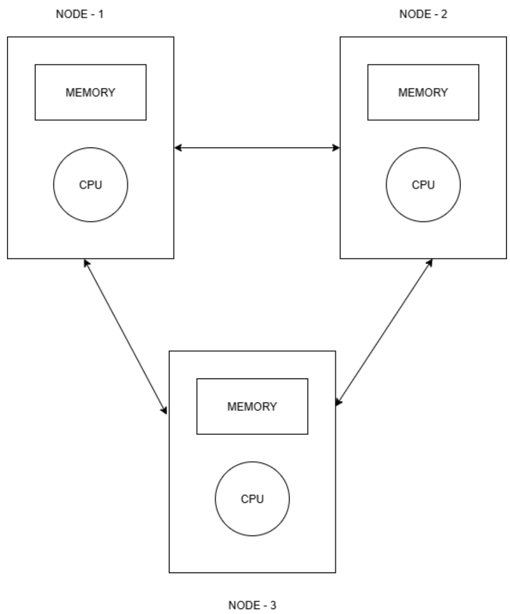
Multithreading OS

- A Multithreading Operating System allows a single process to have multiple threads of execution that run concurrently.
- Each thread represents a separate path of execution, but all threads share the same memory space of the process.
- This helps in improving the performance of applications that perform multiple tasks simultaneously, such as web browsers, video players, or servers.
- ***Key Features:***
  - **Concurrent Execution** – Multiple threads of a process run independently but simultaneously.
  - **Shared Memory** – All threads of a process share code, data, and file system resources.
  - **Faster Context Switching** – Switching between threads is faster than between full processes.
  - **Resource Efficiency** – Reduces memory and resource usage compared to multiple processes.
  - **Improved Responsiveness** – In interactive apps, one thread can handle user input while others process data.
- ***Types of Threading Models***
  - **User-Level Thread (ULT)**
    - Managed by user-level libraries, not known to the OS.
  - **Kernel-Level Thread (KLT)**
    - Managed by the OS kernel.
  - **Hybrid Threads**
    - Combines ULT and KLT.
- ***Example Scenario***
  - **Word Processor(Ex. MS Word)**
    - One thread checks spelling in the background,
    - Another autosaves your document,
    - Another waits for user input.
  - All happen concurrently using threads within one application.
- ***Challenges:***
  - **Thread Synchronization** – Threads share memory, so proper use of locks and semaphores is required to avoid data corruption.
  - **Deadlocks and Race Conditions** – Poor thread management can lead to blocked execution or unpredictable behavior.
  - **Debugging Difficulty** – Bugs are harder to reproduce and debug in multi-threaded environments.
  - **Overhead** – Too many threads can increase context switching overhead.

<u>Distributed OS</u>

- It is a type of operating system in which multiple independent computers are connected through a single communication channel i.e. Network.

- Each of these computers have separate CPUs & Memory
- It is also called loosely coupled systems
- In a Distributed OS, users and applications can access resources (files, printers, processors, memory) across multiple machines as if they are local.
- A Distributed OS ensures that tasks are distributed, load-balanced, and responses are delivered efficiently.
- The concept of Distributed System is shown in the following figure.



Distributed System

- ***Key Features:***
    - **Transparency** – Provides different types of transparency (location, access, replication, concurrency, failure) to hide the distributed nature from users.
    - **Resource Sharing** – CPU, memory, data, and devices are shared between multiple systems.
    - **Fault Tolerance** – If one node fails, others continue to function — improves reliability.
    - **Scalability** – Easy to add more machines to increase power and performance.
    - **Concurrency** – Multiple processes can run concurrently across different machines.
- **Example:**
    - **Google Search or YouTube Backend:**
        - When you search something or stream a video, thousands of servers work together to deliver results.
- **Challenges:**
    - **Network Dependency** – Performance heavily depends on network speed and reliability.
    - **Security** – More exposure over network increases the chance of attacks.
    - **Synchronization** – Difficult to maintain synchronized operations across systems.

- **Complex Design** – Coordination, communication, and failure handling are complex to implement.

Embedded OS

- An Embedded Operating System is a specialized OS designed to run on embedded systems, which are dedicated hardware devices built to perform a specific function.
- Unlike general-purpose OS, Embedded OSs are lightweight, highly reliable
- It is mostly used for controlling tasks in consumer electronics, vehicles, appliances, industrial machines, etc.
- ***Key Features:***
  - **Minimal Resource Usage** – Designed to run with very limited CPU, memory, and storage.
  - **Dedicated Functionality** – Built for specific, repetitive tasks with consistent performance.
  - **Fast Boot Time** – Starts quickly, often within milliseconds or seconds.
  - **High Reliability** – Runs continuously for long periods without failure.
  - **Small Footprint** – Minimal codebase with low overhead.
- ***Types:***
  - **Standalone Embedded OS**
    - Runs directly on hardware without a host OS.
  - **Smart Embedded OS**
    - More complex, includes UI, networking, multitasking.
  - **Real-time Embedded OS**
    - Supports hard or soft real-time operations.
- ***Example:***
  - **Washing Machine Controller:**
    - Controls motor speed, timers, and water levels using sensor inputs.
    - Needs predictable, time-bound operations.
- ***Challenges:***
  - **Resource Limitations** – Low RAM, processing power, and storage.
  - **Firmware Updates** – OS changes are hard once deployed.
  - **Security** – Usually lacks advanced protection unless explicitly added.
  - **Debugging** – Often lacks user interface, making testing difficult.

References:

- Operating System Concepts: Abraham Silberschatz, Peter Baer Galvin, Greg Gagne, Ninth Edition, Wiley India.
- Andrew S. Tanenbaum - Modern Operating Systems, Fourth Edition.
- https://unacademy.com/content/bank-exam/study-material/computer-knowledge/time-sharing-operating-system/

- https://www.scaler.com/topics/time-sharing-operating-system/
- https://www.scaler.com/topics/distributed-operating-system/
- https://www.scaler.com/topics/multithreading-in-os/
- **https://www.geeksforgeeks.org/operating-systems/types-of-operating-systems/**
- https://www.tutorialspoint.com/operating_system/os_types.htm