OS Revision Notes

Lecture 1: Introduction to Operating Systems & System Architecture

Definition and Functions of OS

An Operating System (OS) is system software that acts as an intermediary between the user applications and the computer hardware. It manages hardware resources and provides services for computer programs.

 Process Management – creating and deleting processes, process synchronization, scheduling. Memory Management – tracking memory usage and allocation.
 File System Management – handling files and directories.
 Device Management – coordinating and assigning devices.
 Security and Protection – ensuring authorized access to data.
 User Interface – CLI or GUI for interaction.

Types of Operating Systems

Type Description

Batch Jobs with similar needs are grouped and executed without interaction. Time-

Sharing Allows multiple users to use the system interactively at the same time.

Distributed Manages multiple computers and makes them appear as a single system.

Real-Time	Provides instant response, used in embedded and critical systems.

Kernel, Shell, and System Calls

 Kernel – Core component that directly interacts with hardware.

- Shell – Interface for users to communicate with the OS.
- System Calls – Interface for user programs to request OS services (e.g., open, read, write).

System Architecture

Architecture Description

Monolithic All OS services run in kernel space as one large process. Microkernel

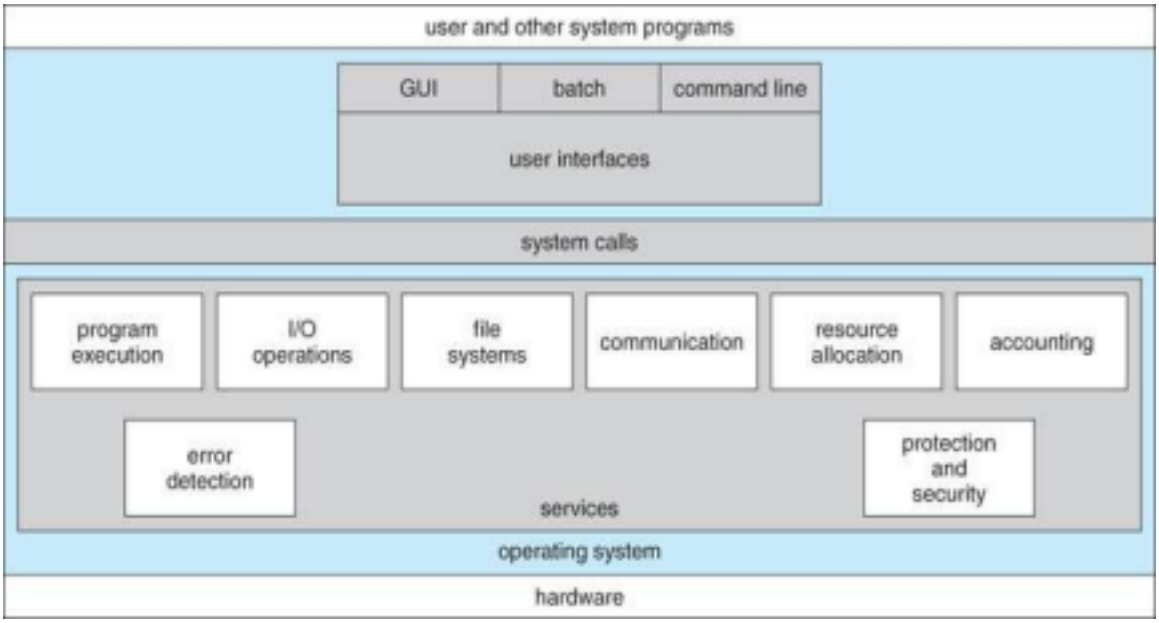
Minimal core in kernel space; services in user space for modularity.

Layered	Hierarchical structure where each layer builds upon the lower one.

Modern OS Overview

- Windows – GUI-based, used widely in personal and enterprise desktops.
- Linux – Open-source, used in servers and development environments.
- macOS – Apple's Unix-based OS known for polished UI and performance.
- Android – Linux-based mobile OS used in smartphones and tablets.

Operating System Services



Lecture 2: Process Management & CPU Scheduling

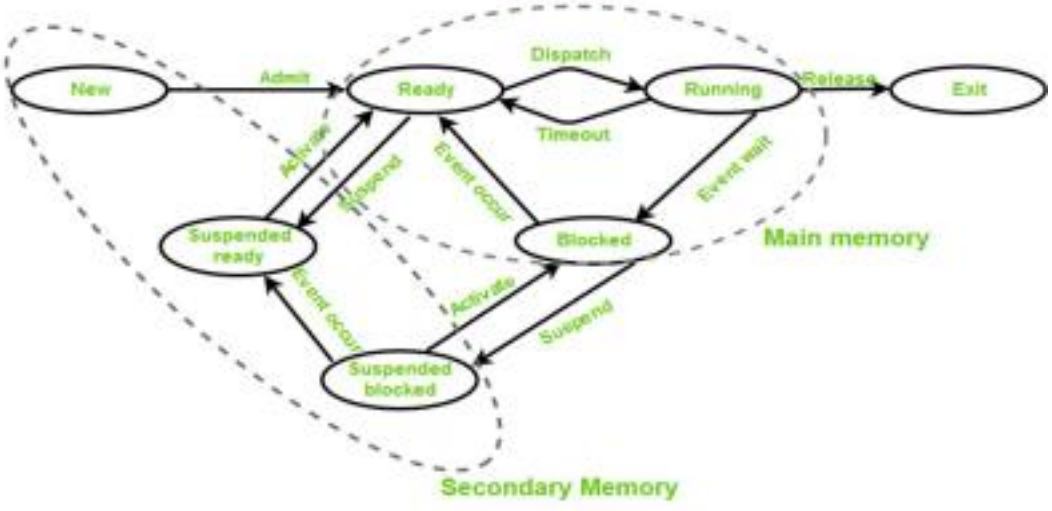
Process Lifecycle and Process Control Block (PCB)

Program in execution is called process.

What is the relationship between process & Program?

It is the same beast with different name or when this beast is sleeping (not executing) called program and when it is executing becomes process.
A process goes through different states in its lifetime:

- New – The process is being created.
- Ready – The process is waiting to be assigned to a processor.
- Running – Instructions are being executed.
- Waiting – The process is waiting for some event to occur (like I/O).
- Terminated – The process has finished execution.



PCB is a data structure maintained by the OS for every process and contains:

- Process ID
- Process state
- Program counter
- CPU registers
- Memory management info
- Accounting and I/O status

Context Switching

Context switching is the process of storing and restoring the state (context) of a CPU so that multiple processes can share a single CPU resource. This enables multitasking.

Threads

A thread is a single sequence stream within in a process. Bcz threads have some of the properties of processes, they are sometime called light weight processes.

In a process, threads allow multiple executions of streams. In many respect, threads are popular way to improve application through parallelism.
The CPU switches rapidly back and forth among the threads giving illusion that the

threads are running in parallel.

Type Description

User-Level Threads Managed by user libraries, not visible to OS.

Kernel-Level Threads	Managed directly by the operating system.

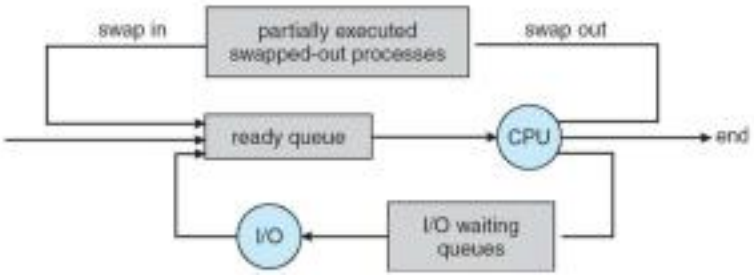
Threads share code, data, and files, but each has its own register and stack.

Schedulers

- **Short-term scheduler** (or **CPU scheduler**) – selects which process should be executed next and allocates CPU
  - Sometimes the only scheduler in a system
  - Short-term scheduler is invoked frequently (milliseconds) ⇒ (must be fast)
- **Long-term scheduler** (or **job scheduler**) – selects which processes should be brought into the ready queue
  - Long-term scheduler is invoked infrequently (seconds, minutes) ⇒ (may be slow)
  - The long-term scheduler controls the **degree of multiprogramming**
- Processes can be described as either:
  - **I/O-bound process** – spends more time doing I/O than computations, many short CPU bursts
  - **CPU-bound process** – spends more time doing computations; few very long CPU bursts
- Long-term scheduler strives for good ***process mix***

Addition of Medium Term Scheduling

- **Medium-term scheduler** can be added if degree of multiple programming needs to decrease
  - Remove process from memory, store on disk, bring back in from disk to continue execution: **swapping**



Scheduling Criteria CPU scheduling terminology

- **CPU utilization** – keep the CPU as busy as possible
- **Throughput** – # of processes that complete their execution per time unit
- **Burst time/ execution time/ running time:** is the time process require for running on CPU
- **Waiting time**: time spend by a process in ready state waiting for CPU
- **Arrival time**: when a process enter the ready state
- **Exit time**: when process completes execution and exit from system
- **Turn around time**: total time spend by a process in the system
- TAT = ET – AT or BT + WT
- **Response time**: Time between a process enter ready queue and get scheduled on the CPU for the first time

CPU Scheduling Algorithms

Algorithm Description

FCFS Non-preemptive, first process in the queue is served first. Processes Suffer from convoy effects

SJF/ SRTF Non-preemptive/preemptive, process with shortest burst time is chosen. Processes suffer from starvation.

Priority CPU is allocated to the process which process the highest priority (Number may be higher or lower). Processes suffer from starvation. Solution Aging

Round Robin Each process gets a fixed time quantum.

Multilevel Queue	Processes are grouped by priority/type into queues.

Multithreading Models

Model Explanation

Many-to-One Many user-level threads mapped to one kernel thread.

One-to-One Each user thread maps to a unique kernel thread.

Many-to-Many	Many user threads mapped to fewer or equal kernel threads.

Scheduling in Multi-Core Processors

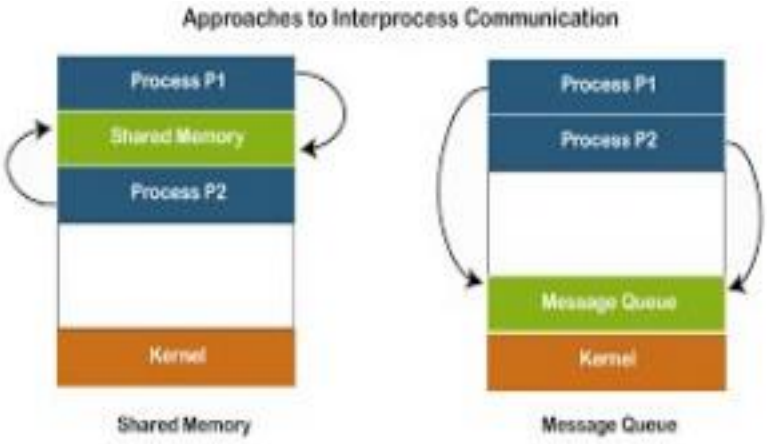
In modern OS, multi-core scheduling ensures efficient use of all CPU cores using:
 Load Balancing – Distributes processes evenly across cores.
 Processor Affinity – Keeps a process on the same CPU to maximize cache performance. Symmetric Multiprocessing (SMP) – All processors are treated equally.

Lecture 3: Inter-Process Communication (IPC) & Synchronization

IPC Mechanisms

IPC allows processes to communicate and synchronize their actions. Two primary methods are:

Method	Description
Shared Memory	Processes share a common memory space. Requires synchronization mechanisms.
Message Passing	Processes communicate via messages (send/receive). Easier but slower due to OS overhead.



Critical Section: When more than one processes access a same code segment that segment is known as critical section. Critical section contains shared variables or resources which are needed to be synchronized to maintain consistency of data variable.

Critical Section Problem & Race Conditions

A critical section is a segment of code where shared resources are accessed. Race conditions occur when the outcome depends on the sequence of process execution.

 To avoid race conditions, we need mutual exclusion in critical sections.
 Solutions require three conditions: Mutual Exclusion, Progress, and Bounded Waiting.

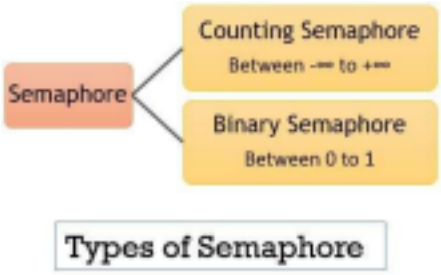
Synchronization Tools

Tool	Description

Semaphores Integer variable used for signaling. Can be binary or counting.

Mutex Mutual Exclusion lock used to prevent race conditions.

Monitors	High-level abstraction that encapsulates shared variables and operations.



Types of Semaphore

```
P(Semaphore s){
   while(S == 0);   /* wait until s=0 */
  s=s-1;
}

V(Semaphore s){
        s=s+1;
}
```

Note that there is Semicolon after while. The code gets stuck Here while s is 0.

Classical Synchronization Problems

 Producer-Consumer Problem – Synchronizing producer/consumer using buffer. Readers-Writers Problem – Ensuring multiple readers but only one writer.

 Dining Philosophers – Managing access to shared forks among philosophers.

Real-World IPC Examples

 Pipes – Unidirectional data flow between related processes.

 Sockets – Bi-directional communication, used over networks.

 Message Queues – Queue-like structure in OS to send messages between processes.

Lecture 4: Deadlock & Resource Management

Deadlock Definition and Conditions

A deadlock is a situation in which a set of processes are blocked because each process is holding a resource and waiting for another resource held by another process.

 Mutual Exclusion – Only one process can use a resource at a time.

 Hold and Wait – A process holding a resource is waiting for more.

 No Preemption – Resources cannot be forcibly taken.

 Circular Wait – A closed chain of processes exists where each holds at least one resource needed by the next.

Deadlock Handling Strategies

Strategy Explanation

Deadlock Prevention Design system to eliminate one of the conditions (e.g., avoid hold and wait).

Deadlock Avoidance Requires information about future resource requests (e.g., Banker's Algorithm).

Deadlock Detection Allow deadlock, detect it with algorithm, then recover.

Deadlock Recovery Abort or preempt processes/resources to resolve deadlock.

Banker's Algorithm

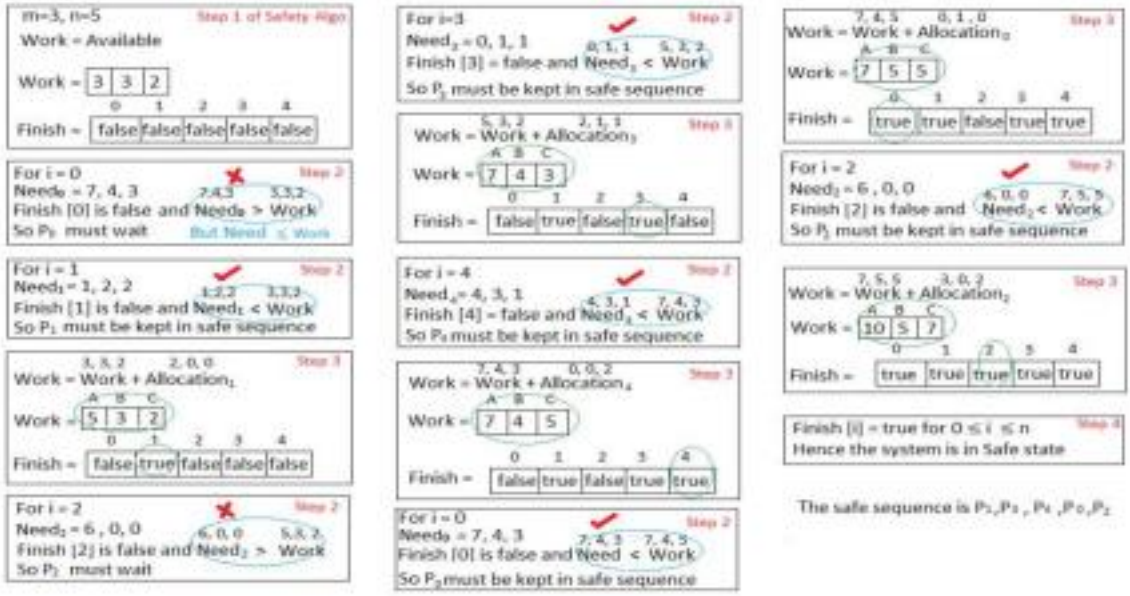
It is used to avoid deadlock by ensuring a safe sequence of resource allocation.

- Maintain matrices: Allocation, Max, Available, Need.
- Check if processes can finish with available resources.
- If all processes can finish in some order, system is in a safe state.

Considering a system with five processes P0 through P4 and three resources of type A, B, C. Resource type A has 10 instances, B has 5 instances and type C has 7 instances. Suppose at time t0 following snapshot of the system has been taken:

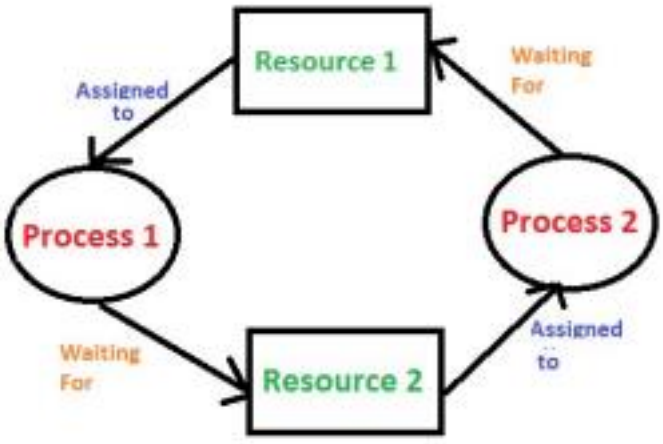
Process	Need		
	A	B	C
P_0	7	4	3
P_1	1	2	2
P_2	6	0	0
P_3	0	1	1
P_4	4	3	1

Process	Allocation	Max	Available
	A B C	A B C	A B C
P_0	0 1 0	7 5 3	3 3 2
P_1	2 0 0	3 2 2	
P_2	3 0 2	9 0 2	
P_3	2 1 1	2 2 2	
P_4	0 0 2	4 3 3	



Resource Allocation Graph (RAG)

RAG is a graphical representation showing resource allocation among processes. A cycle in the graph indicates the possibility of deadlock.



Other Issues: Starvation and Livelock

Issue Description

Starvation A process waits indefinitely due to low priority or unfair scheduling.

Livelock	Processes continually change states without making progress.

Lecture 5: Memory Management

Logical vs Physical Address Space

Logical Address is the address generated by the CPU, whereas Physical Address is

the actual location in memory. OS maps logical to physical addresses using memory management unit (MMU).

Fixed & Variable-Sized Allocation

- How to satisfy a request of size ***n*** from a list of free holes?
- **First-fit**: Allocate the ***first*** hole that is big enough
- **Best-fit**: Allocate the ***smallest*** hole that is big enough; must search entire list, unless ordered by size
  - Produces the smallest leftover hole
- **Worst-fit**: Allocate the ***largest*** hole; must also search entire list
  - Produces the largest leftover hole

➢First-fit and best-fit better than worst-fit in terms of speed and storage utilization

Memory Allocation Techniques

Technique	Explanation
Contiguous Allocation	Each process is allocated a single continuous block of memory. Simple but can cause fragmentation.
Paging	Divides memory into fixed-size pages and frames. Eliminates external fragmentation.

Segmentation	Divides memory based on variable-sized logical units or segments.

Virtual Memory

Virtual Memory allows processes to execute even if they are not completely in main memory. Achieved through demand paging.

 Demand Paging – Pages are loaded into memory only when required. Page Table – Keeps track of mapping between logical pages and physical frames. Page Fault – Occurs when a required page is not in memory.

Page Replacement Algorithms

Algorithm Description

FIFO Replaces the oldest page in memory.

LRU	Replaces the page that hasn't been used for the longest time.

Optimal	Replaces the page that will not be used for the longest time in future (theoretical best).

Page replacement Policies Comparison

Additional Concepts

 TLB (Translation Lookaside Buffer) – Cache that stores recent address translations. Thrashing – Excessive paging leading to low CPU utilization.
 Working Set Model – Defines the number of pages a process is currently using.

Lecture 6: File & I/O Management + OS Security

File System Interface and Implementation

The file system provides a way to store and retrieve data and programs. It includes file attributes, operations, and the logical structure of files.
 Attributes – Name, Type, Size, Permissions, Timestamps.
 Operations – Create, Read, Write, Delete, Seek.
 File Types – Text, Binary, Directory, etc.

Directory Structure

Structure Description

Single-Level All files in the same directory.

Two-Level Separate directory for each user.

Tree Hierarchical structure with directories and subdirectories. Acyclic

Graph Allows shared subdirectories or files.

General Graph	Includes links and cycles (requires cycle detection).

File Allocation Methods

Method Explanation

Contiguous Each file occupies a set of contiguous blocks on disk.

Linked Each block contains a pointer to the next block.

Indexed	Uses an index block to keep pointers to all file blocks.

Disk Scheduling Algorithms

Algorithm Description

FCFS Serve requests in order of arrival.

SSTF Selects the request closest to current head position. SCAN Moves head from one end to other, servicing in one direction. C-SCAN Only services requests in one direction then resets.

LOOK/C-LOOK	Like SCAN but only goes as far as the last request.

- Seek time: Time required to move the R/W head on the desired track.
- Disk Scheduling: In case of multiple I/O request, disk scheduling algorithm must decide which request must be executed first.

OS Security

- Authentication – Verifies identity of users (e.g., password, biometrics).
- Access Control – Determines access permissions for resources.
- Threats – Malware, unauthorized access, privilege escalation, etc.

Security techniques include encryption, firewalls, user permissions, and system hardening.

File System Examples

File System Features

FAT Simple structure, used in USB drives and earlier Windows systems.

NTFS Supports permissions, encryption, journaling. Used in modern Windows.

ext4	Linux file system with journaling, extents, large volume support.