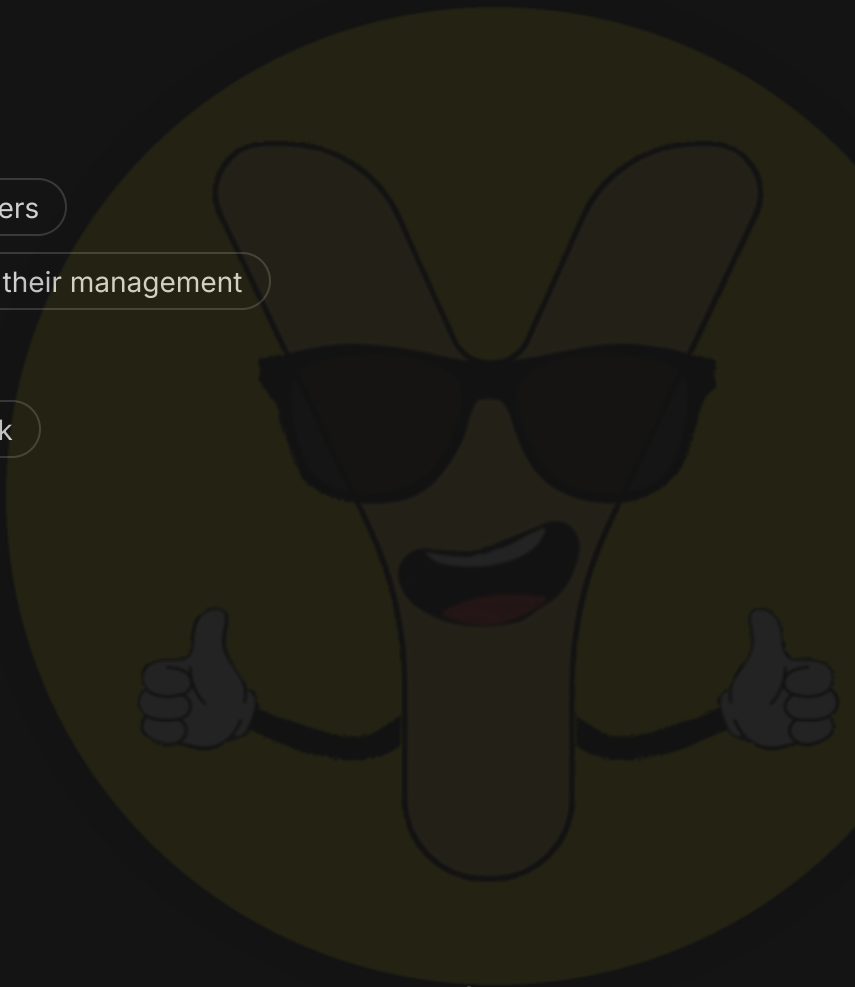
AKTU · 2ND YEAR · SEMESTER 4 · BCS401

Operating System

Unit 3 — CPU Scheduling and Deadlock

- Scheduling Concepts
- Performance Criteria
- Process States
- Process Transition Diagram
- Schedulers
- Process Control Block (PCB)
- Process address space
- Process identification information
- Threads and their management
- Scheduling Algorithms
- Multiprocessor Scheduling
- Deadlock: System model, Deadlock characterization, Prevention, Avoidance and detection, Recovery from deadlock



DECODED BY

Chirag Jain

Founder, Ycotes · Gold Medalist in Artificial Intelligence, IIT Madras · taught 10,000+ students

CPU Scheduling Concepts

Fundamental concepts governing how the operating system assigns CPU execution time to processes.

- Multiprogramming maximizes CPU utilization by having multiple processes loaded in memory simultaneously.
- CPU-I/O Burst Cycle: Process execution consists of a cycle of CPU execution (burst) and I/O wait.
- CPU-bound processes have long CPU bursts; I/O-bound processes have short CPU bursts.
- The CPU scheduler selects a process from the ready queue and allocates the CPU to it.
- Non-preemptive scheduling: Once CPU is allocated, the process keeps it until termination or I/O request.
- Preemptive scheduling: The CPU can be forcibly taken away from a running process (e.g., time quantum expiration).

CPU Scheduling Performance Criteria

Quantitative metrics used to evaluate and compare the efficiency of different scheduling algorithms.

- CPU Utilization: Percentage of time the CPU is busy (0% to 100%).
- Throughput: Number of processes completed per time unit.
- Turnaround Time (T_t): Total time from process submission to completion ($T_{\text{completion}} - T_{\text{arrival}}$).
- Waiting Time (T_w): Total time spent waiting in the ready queue ($T_t - T_{\text{burst}}$).
- Response Time (T_r): Time from submission until the first response is produced (crucial for interactive systems).
- Optimization goals: Maximize utilization and throughput; minimize turnaround, waiting, and response time.

Process States

PROCESS STATE

The distinct condition or status of a process during its lifecycle from creation to termination.

- New: The process is currently being created.
- Ready: The process is waiting to be assigned to a processor.
- Running: Instructions are actively being executed by the CPU.
- Waiting / Blocked: The process is waiting for some event to occur (such as I/O completion).
- Terminated: The process has finished execution.

Process Transition Diagram



State transitions of a process in the operating system.

System Schedulers

Different schedulers manage processes at different levels of the operating system hierarchy.

- Long-Term Scheduler (Job Scheduler): Selects processes from secondary storage (disk pool) and loads them into RAM.
- Short-Term Scheduler (CPU Scheduler): Selects from ready queue and allocates CPU. Executes very frequently.
- Medium-Term Scheduler: Handles swapping processes in and out of main memory to manage multiprogramming degree.
- Degree of Multiprogramming is primarily controlled by the long-term scheduler.
- I/O-bound vs CPU-bound mix is vital for long-term scheduler decisions.

Process Control Block (PCB)

PCB (TASK CONTROL BLOCK)

A data structure maintained by the operating system containing all information associated with a specific process.

- Process State: Current state (New, Ready, Running, etc.).
- Program Counter: Address of the next instruction to execute for this process.
- CPU Registers: Accumulators, index registers, stack pointers, and condition codes.
- CPU Scheduling Information: Process priority, pointers to scheduling queues.
- Memory-Management Information: Page tables, base/limit registers.
- Accounting and I/O status information: Amount of CPU used, list of allocated I/O devices.

Process Address Space & Identification

Memory layout and unique identifiers assigned to every active process.

- Process Address Space consists of: Text (code), Data (global variables), Heap (dynamic memory), and Stack (local variables/function calls).
- Process ID (PID): A unique integer assigned by the OS to identify every active process.
- Parent/Child PIDs track process lineage (e.g., using fork() system call).
- User ID (UID) and Group ID (GID) define process security permissions.
- Context Switch: Saving context (PCB) of old process and loading context of new process via CPU.

Threads and Their Management

A thread is a basic unit of CPU utilization, also known as a lightweight process (LWP).

- Components of a thread: Thread ID, Program Counter, register set, and stack.
- Threads share with peer threads: Code section, data section, and OS resources (like open files).
- User-Level Threads (ULT): Managed by user-space libraries; fast creation, but kernel blocks all if one blocks.
- Kernel-Level Threads (KLT): Managed directly by OS; heavier, but true parallel execution on multi-core.
- Multithreading models: Many-to-One, One-to-One, and Many-to-Many.

Scheduling Algorithms: FCFS and SJF

First-Come, First-Served and Shortest Job First scheduling policies.

- FCFS: Non-preemptive. First process in ready queue gets the CPU. Suffers from the Convoy Effect (short processes wait behind long ones).
- SJF (Shortest Job First): Associates each process with the length of its next CPU burst. Optimal average waiting time.
- SJF can be Non-preemptive or Preemptive (Shortest Remaining Time First - SRTF).
- Challenge in SJF: Determining or predicting the length of the next CPU burst (using exponential smoothing).

Scheduling Algorithms: Priority and Round Robin

Priority-based selection and time-sliced rotational scheduling.

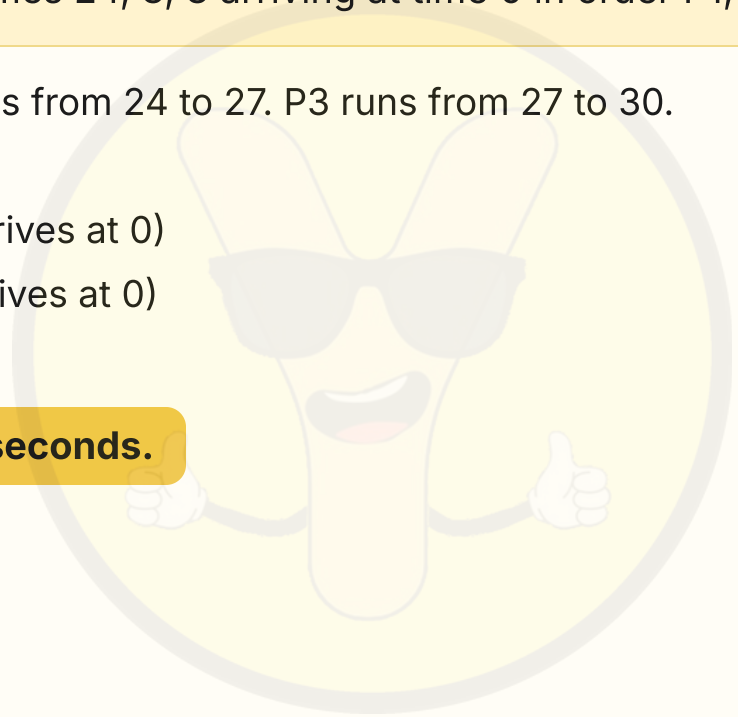
- Priority Scheduling: CPU allocated to the process with highest priority (smallest integer = highest priority typically).
- Major issue with Priority: Starvation (indefinite blocking). Solution: Aging (gradually increasing priority of waiting processes).
- Round Robin (RR): Preemptive FCFS with a small unit of time called a time quantum (q).
- RR Performance depends heavily on quantum size: too large becomes FCFS; too small causes excessive context switch overhead.

Worked Example: FCFS Scheduling

Given processes P1, P2, P3 with burst times 24, 3, 3 arriving at time 0 in order P1, P2, P3. Find average waiting time.

1. Gantt Chart: P1 runs from 0 to 24. P2 runs from 24 to 27. P3 runs from 27 to 30.
2. Waiting time for P1 = 0
3. Waiting time for P2 = 24 (starts at 24, arrives at 0)
4. Waiting time for P3 = 27 (starts at 27, arrives at 0)
5. Total Waiting Time = 0 + 24 + 27 = 51

Average Waiting Time = 51 / 3 = 17 milliseconds.



Comparison of CPU Scheduling Algorithms

Algorithm	Preemptive?	Advantages	Disadvantages
FCFS	No	Simple, easy to implement	Convoy effect, high average wait
SJF	Both	Minimum average waiting time	Hard to predict burst length, starvation
Priority	Both	Supports urgent/important tasks	Starvation for low priority tasks
Round Robin	Yes	Fair, good response time	High overhead if quantum is too small

Multiprocessor Scheduling

Scheduling tasks across systems with multiple CPU cores.

- Asymmetric Multiprocessing: One master processor handles system data and scheduling; slaves execute user code.
- Symmetric Multiprocessing (SMP): Each processor is self-scheduling, examining the ready queue independently.
- Processor Affinity: Process prefers to run on the same CPU to leverage cache memory (Soft vs Hard affinity).
- Load Balancing: Keeping workload evenly distributed across all processors (Push migration vs Pull migration).
- Multi-core processors and Hyper-threading present hardware threads sharing execution resources.

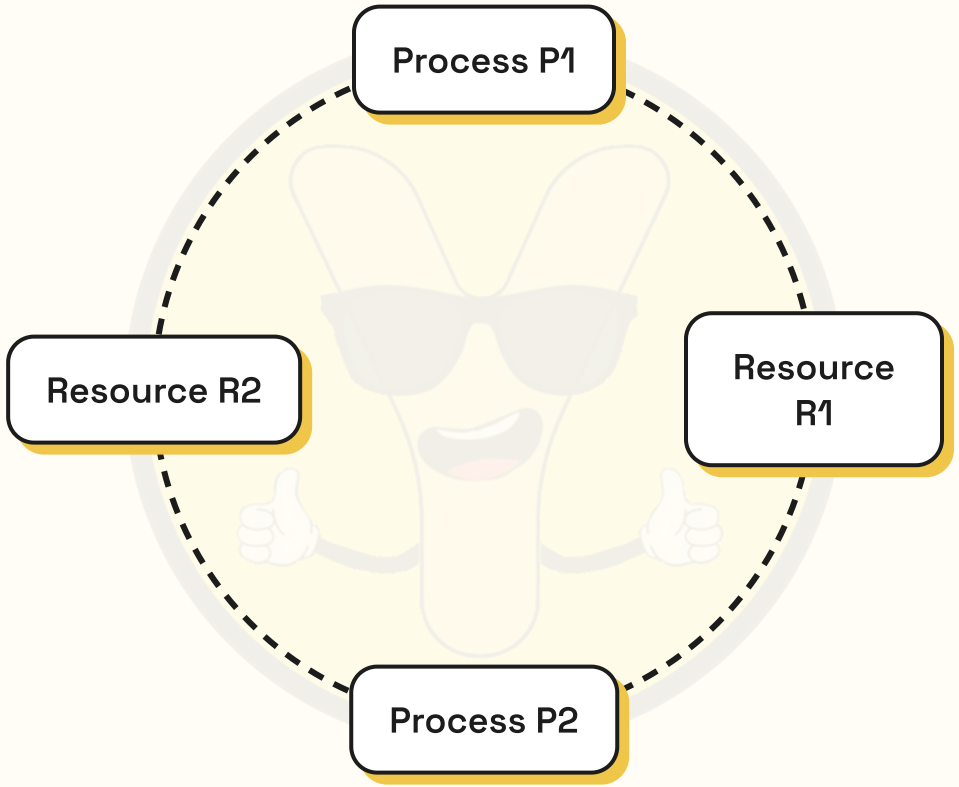
Deadlock System Model

DEADLOCK

A situation where a set of blocked processes each holding a resource and waiting to acquire a resource held by another process in the set.

- System resources include CPU cycles, memory space, files, and I/O devices.
- Normal resource usage protocol: Request → Use → Release.
- Coffman Conditions (All four must hold simultaneously for deadlock):
- 1. Mutual Exclusion: At least one resource must be held in a non-shareable mode.
- 2. Hold and Wait: Process holding at least one resource is waiting to acquire additional resources.
- 3. No Preemption: Resources cannot be forcibly taken from a process.
- 4. Circular Wait: A closed chain of processes exists where each waits for a resource held by the next.

Resource Allocation Graph (RAG)



A directed graph representing resource allocation and requests containing a cycle.

Deadlock Handling Strategies

Methods used by operating systems to manage deadlock scenarios.

- Ignore the problem (Ostrich Algorithm): Pretend deadlocks never occur. Used by most operating systems including UNIX and Windows.
- Deadlock Prevention: Ensure at least one of the four Coffman conditions never holds.
- Deadlock Avoidance: Operating system requires a priori information about resource requests to ensure system never enters unsafe state.
- Deadlock Detection & Recovery: Allow deadlocks to happen, detect them, and recover via preemption or termination.

Deadlock Prevention

Restricting how resource requests can be made to invalidate Coffman conditions.

- Mutual Exclusion: Make resources shareable (e.g., read-only files). Hard for exclusive resources like printers.
- Hold and Wait: Require processes to request and be allocated all resources at once before beginning execution (low resource utilization).
- No Preemption: If a process holding resources requests another that cannot be immediately allocated, release all currently held resources.
- Circular Wait: Impose a total ordering of all resource types; require requests to be made in strictly increasing sequence of enumeration.

Deadlock Avoidance (Banker's Algorithm)

Ensuring the system always remains in a safe state.

- Safe State: There exists a safe sequence $\langle P_1, P_2, \ldots, P_n angle$ such that for each P_i, resource requests can be satisfied by available resources plus resources held by preceding processes.
- Unsafe state does not necessarily mean deadlock, but it can lead to deadlock.
- Banker's Algorithm Data Structures: Available vector, Max matrix, Allocation matrix, Need matrix ($\mathrm{Need} = \mathrm{Max} - \mathrm{Allocation}$).
- Safety Algorithm: Finds if a safe execution sequence exists.
- Resource-Request Algorithm: Evaluates if a specific request can be safely granted immediately.

Deadlock Detection and Recovery

Identifying active deadlocks and taking corrective actions.

- Detection for single instance of each resource type: Resource Allocation Graph with cycle detection ($O(n^2)$).
- Detection for multiple instances: Matrix-based algorithm similar to Banker's algorithm using Available, Allocation, and Request matrices.
- Recovery - Process Termination: Abort all deadlocked processes OR abort one process at a time until the deadlock cycle is eliminated.
- Recovery - Resource Preemption: Selecting a victim process, preempting resources, and rolling back the victim to a safe state.

Exam corner

- **Q1** Explain the process states and draw the complete process transition diagram with all state transitions.
- **Q2** Define CPU scheduling. Explain FCFS, SJF, and Round Robin algorithms with numerical examples.
- **Q3** What is a Process Control Block (PCB)? Explain its various fields in detail.
- **Q4** State and explain the four necessary conditions for deadlock to occur.
- **Q5** Explain the Banker's Algorithm for deadlock avoidance with necessary data structures and safety algorithm steps.
- **Q6** Differentiate between Preemptive and Non-preemptive scheduling. Also explain the Convoy Effect.
- **Q7** Discuss deadlock prevention techniques by invalidating the Coffman conditions.

Unit 3 done. You are exam-ready.

Revise the Exam corner once more, then open Unit 4 on theycotes.com. Every AKTU subject, unit by unit, free. Notes by Chirag Jain, Ycotes.

