OPERATING SYSTEM (315319)

CHAPTER 3

CPU SCHEDULING

Chapter Scope

- Pre-emptive and non-pre-emptive scheduling
- Scheduling criteria and CPU/I/O burst cycle
- FCFS, SJF, SRTF, Priority, Round Robin and Multilevel Queue scheduling
- Waiting time and turnaround time calculations
- Deadlock system model, necessary conditions, prevention and avoidance
- Banker's Algorithm

Question Bank Coverage

- The uploaded source contains 40 Unit III questions spanning Recall, Understanding and Applied levels.

3.1 Learning Outcomes

Outcome	Expected capability
3.1	Define CPU scheduling and explain pre-emptive and non-pre-emptive scheduling.
3.2	Explain scheduling criteria and CPU/I/O burst cycle.
3.3	Explain and apply FCFS, SJF, SRTF, Priority, Round Robin and Multilevel Queue scheduling.
3.4	Calculate waiting time and turnaround time.
3.5	Explain deadlock and Banker's Algorithm.

3.2 Pre-emptive and Non-pre-emptive Scheduling

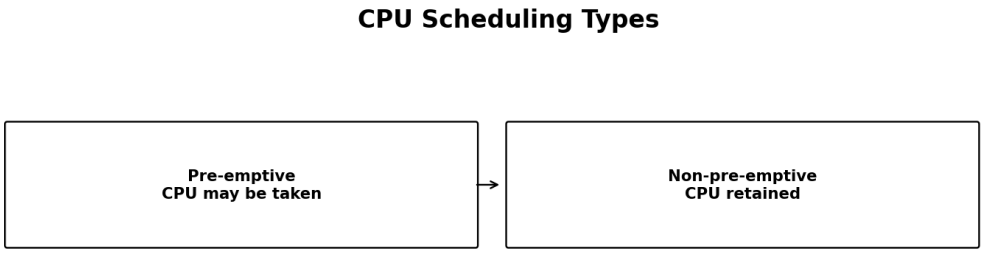

Fig. 3.1 Scheduling types

Feature	Pre-emptive	Non-pre-emptive
CPU control	Running process may be interrupted.	Process keeps CPU until release/termination/wait.
Typical triggers	Running→Ready; Waiting→Ready.	Running→Waiting; termination.
Examples in source	SJF/SRTF, Priority, Round Robin	FCFS
Main idea	Higher-priority/qualifying process may take CPU.	Current process continues.

Exam Focus Define both terms and differentiate them with examples.

3.3 Scheduling Criteria

Criterion	Meaning
CPU Utilization	Percentage of time CPU is busy.
Throughput	Number of processes completed per unit time.
Turnaround Time	Time from submission to completion.
Waiting Time	Total time spent in the ready queue.
Response Time	Time from request submission to first response.
Deadline	Time limit for completing process execution.

Important Formulas

- TAT = CT − AT
- WT = TAT − BT
- Average WT = Sum of WT / Number of processes
- Average TAT = Sum of TAT / Number of processes

3.4 CPU and I/O Burst Cycle

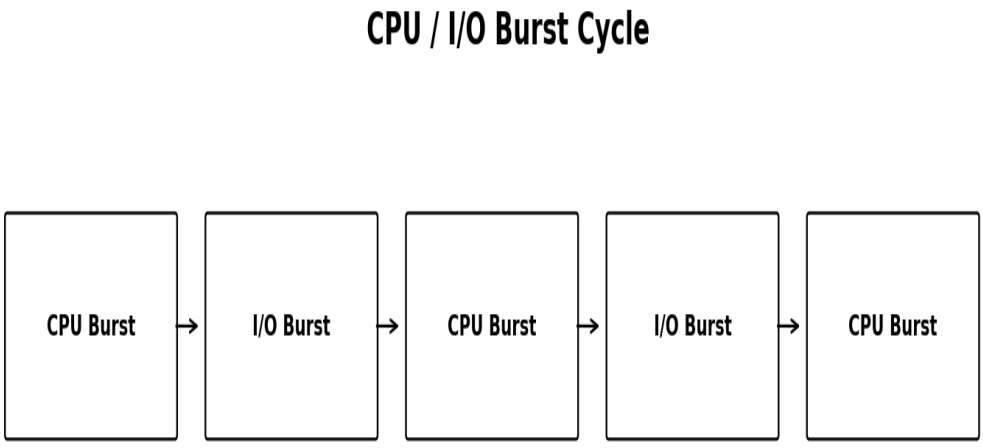

Fig. 3.2 CPU/I/O burst cycle

- Processes alternate between CPU and I/O bursts.
- CPU-bound processes generally have longer CPU bursts.
- I/O-bound processes spend more time in I/O operations.

3.5 FCFS (First Come First Serve)

- FCFS is non-pre-emptive.
- The process that requests the CPU first is allocated first.
- It is implemented using a FIFO ready queue.
- The source notes that average waiting time can be long.

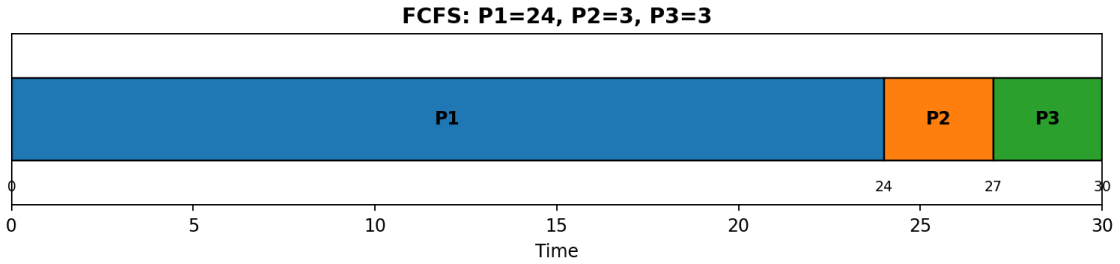

Fig. 3.3 FCFS example

Process	BT	WT
P1	24	0
P2	3	24
P3	3	27

- Average WT = (0 + 24 + 27) / 3 = 17 units.

3.6 SJF (Shortest Job First)

- CPU is assigned to the process with the shortest next CPU burst.
- Equal burst times are resolved using FCFS.
- SJF can reduce average waiting time but requires burst-time information in advance.

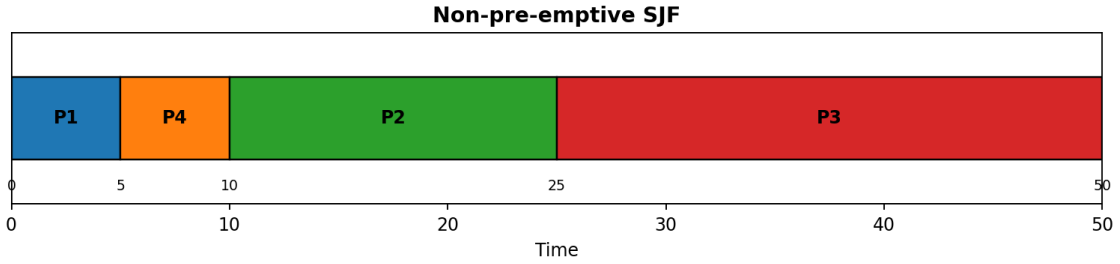

Fig. 3.4 Non-pre-emptive SJF

Process	BT	WT
P1	5	0
P4	5	5
P2	15	10
P3	25	25

- Source example: Average WT = 10 ms.

3.7 Pre-emptive SJF / SRTF

- SRTF is the pre-emptive form of SJF.
- A running process can be pre-empted when a newly arrived process has shorter remaining time.
- The shortest remaining burst is selected at each decision point.

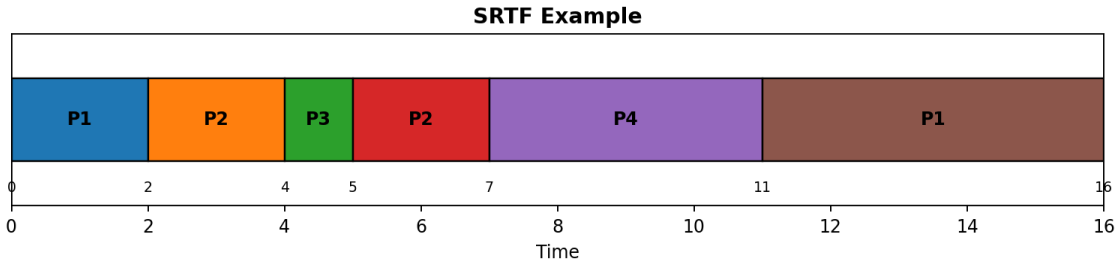

Fig. 3.5 SRTF example

Feature	Non-pre-emptive SJF	SRTF
Pre-emption	No	Yes
Response time	Higher	Lower
Complexity	Easier	More complex
Starvation	Possible	Possible

3.8 Priority Scheduling

- Each process is associated with an integer priority.
- In the source, smaller integer means higher priority.
- Priority scheduling can be pre-emptive or non-pre-emptive.
- A major problem is starvation/indefinite blocking.
- Aging gradually increases the priority of long-waiting processes.

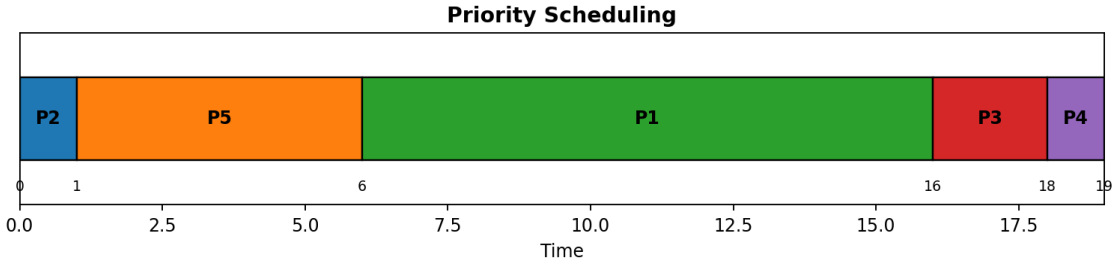

Fig. 3.6 Priority scheduling example

Process	BT	Priority
P1	10	3
P2	1	1
P3	2	4
P4	1	5
P5	5	2

Advantages	Disadvantages
Simple	Starvation / indefinite blocking
Supports priority-based requirements	Low-priority processes may wait indefinitely
Useful for varying time/resource requirements	Unfinished low-priority work may be lost if system fails

- Source example: Average WT = 8.2 ms.

3.9 Round Robin (RR)

- RR is pre-emptive.

- Each process receives a fixed time quantum.
- An unfinished process is moved to the end of the ready queue.
- Large quantum makes RR behave like FCFS; very small quantum causes many context switches.

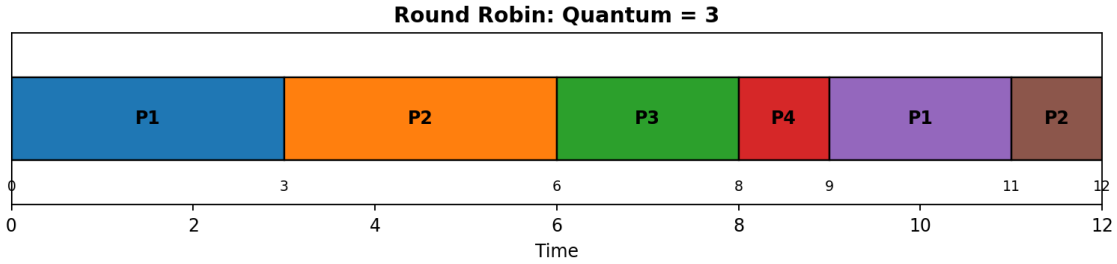

Fig. 3.7 Round Robin, quantum = 3

Process	AT	BT	CT	TAT	WT
P1	0	5	11	11	6
P2	1	4	12	11	7
P3	2	2	8	6	4
P4	3	1	9	6	5

- Average TAT = 8.5 units; Average WT = 5.5 units.

3.10 Multilevel Queue Scheduling

- Processes are permanently assigned to separate queues.
- Each queue has its own scheduling algorithm.
- Scheduling occurs between queues and within each queue.
- The source examples include interactive, system and batch process classes.

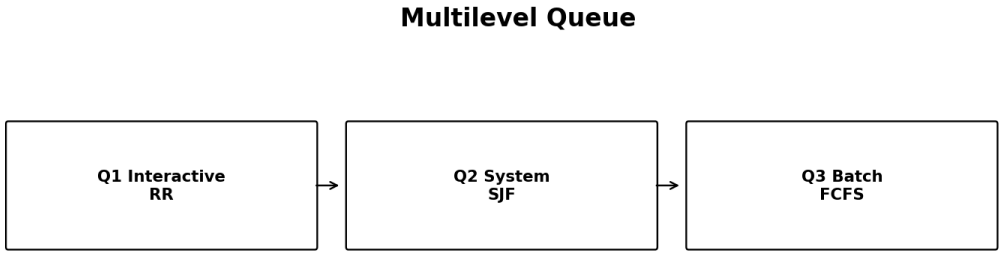

Fig. 3.8 Multilevel Queue structure

3.11 Multilevel Queue – Worked Example

Process	Queue	AT	BT
P1	Q1 RR(q=4)	0	6
P2	Q2 SJF	1	4
P3	Q1 RR(q=4)	2	5
P4	Q3 FCFS	3	7

- Queue priority: Q1 > Q2 > Q3.

1. P1 runs 0–4; remaining BT = 2.
2. P3 runs 4–8; remaining BT = 1.
3. P1 runs 8–10 and completes.
4. P3 runs 10–11 and completes.
5. P2 runs 11–15 and completes.
6. P4 runs 15–22 and completes.

Advantages	Disadvantages
Simple classification	Rigid/static assignment
Differentiates process types	Lower queues may starve
Useful for defined categories	Less flexible

3.12 Deadlock

- A deadlock can occur when processes compete for a finite set of resources.
- A process follows Request → Use → Release.
- A set of processes is deadlocked when each waits for an event/resource dependent on another process in the same set.

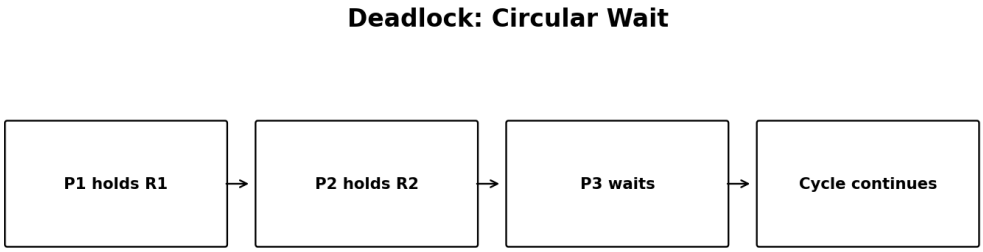

Fig. 3.9 Circular wait leading to deadlock

3.13 Necessary Conditions for Deadlock

Condition	Meaning
Mutual Exclusion	At least one resource is non-sharable.
Hold and Wait	A process holds allocated resources while waiting for additional resources.
No Pre-emption	Allocated resources cannot be forcibly taken away.
Circular Wait	Processes form a circular resource-wait dependency.

3.14 Deadlock Handling

- Prevention: ensure at least one necessary condition cannot hold.
- Avoidance: grant requests only when the resulting state remains safe.
- Detection and recovery: detect deadlock after it occurs and recover.
- Ignorance: the system may choose to ignore deadlock.

3.15 Deadlock Prevention

Condition	Prevention approach from source
Mutual Exclusion	Use sharable resources where possible; intrinsically non-sharable resources remain a limitation.
Hold and Wait	Request all resources before execution, or request more only when holding none.
No Pre-emption	In selected cases take resources back, recognizing possible consistency/work-loss issues.
Circular Wait	Impose an ordering on resources to prevent cycles.

3.16 Deadlock Avoidance

- Avoidance requires advance information about resource needs.
- For each request, the OS checks whether granting it keeps the system safe.
- An unsafe request is delayed.

3.17 Banker's Algorithm

Structure	Meaning
Available	Available instances of each resource type.
Max	Maximum demand matrix.
Allocation	Currently allocated resources.
Need	Remaining resource needs.

Key Formula $Need[i][j] = Max[i][j] - Allocation[i][j]$

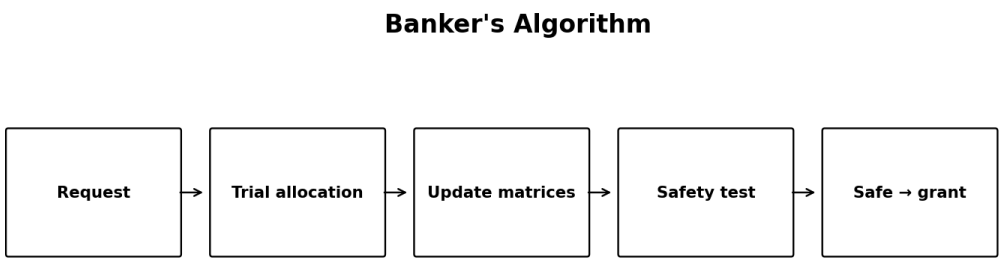

Fig. 3.10 Banker's Algorithm safety-check flow

Algorithm Steps

- Consider the request using a trial allocation.
- Update Available, Allocation and Need in a temporary/work area.
- Compare Available with each Need row.
- If no process can finish with the current Available resources, the state is unsafe.
- If a process can finish, assume it receives its remaining resources and then releases them.

- Repeat until all processes can complete or no progress is possible.
- Commit the allocation only when the state is safe.

3.18 Banker's Algorithm – Source Example

- Five processes P0–P4 and resource types A=10, B=5, C=7.

Process	Allocation	Max	Need
P0	0 1 0	7 5 3	7 4 3
P1	2 0 0	3 2 2	1 2 2
P2	3 0 2	9 0 2	6 0 0
P3	2 1 1	2 2 2	0 1 1
P4	0 0 2	4 3 3	4 3 1

Safe sequence The uploaded source gives <P1, P3, P4, P2, P0> as a safe sequence.

3.19 Quick Revision Sheet

Topic	Remember
FCFS	Non-pre-emptive; FIFO.
SJF	Shortest next CPU burst.
SRTF	Pre-emptive SJF; shortest remaining time.
Priority	Smaller integer = higher priority in source; starvation possible; aging helps.
RR	Pre-emptive; fixed quantum; cyclic FIFO.
MLQ	Fixed queues; each queue has its own policy.
TAT	CT − AT.
WT	TAT − BT.
Deadlock conditions	Mutual exclusion, hold and wait, no pre-emption, circular wait.
Banker's	Trial allocation + safety test; Need = Max − Allocation.

Exam-Oriented Checklist

- Define CPU scheduling and pre-emptive/non-pre-emptive scheduling.
- Explain scheduling criteria and CPU/I/O burst cycle.
- Explain FCFS, SJF and SRTF with Gantt charts.
- Explain Priority scheduling, starvation and aging.
- Explain Round Robin and calculate TAT/WT.
- Discuss Multilevel Queue scheduling.
- Explain deadlock and its four necessary conditions.
- Explain deadlock prevention and avoidance.
- Explain Banker's Algorithm and safe/unsafe states.