Operating Systems

An operating system manages hardware and provides services to programs.

OS functions

- Process management
- Memory management
- File management
- I/O management
- Security
- Resource allocation

Process vs Program

Program: passive file stored on disk.
Process: active program in execution.

Process states

Common states:

- New
- Ready
- Running
- Waiting/Blocked
- Terminated

Flow:

```
New -> Ready -> Running -> Terminated
              Running -> Waiting -> Ready
```

Process vs Thread

Process	Thread
Independent execution unit	Lightweight execution unit
Separate address space	Shares process memory
Context switch is heavier	Context switch is lighter
More isolation	Less isolation

Context switching

Context switching is saving the current process/thread state and loading another state.

It adds overhead because CPU time is spent switching instead of executing user work.

CPU scheduling

FCFS

First Come First Served.

- Simple.
- Can cause convoy effect.

SJF

Shortest Job First.

- Gives minimum average waiting time.
- Needs knowledge of burst time.

Round Robin

Each process gets fixed time quantum.

- Good for time-sharing systems.
- Too small quantum causes many context switches.
- Too large quantum behaves like FCFS.

Priority scheduling

Higher priority process runs first.

Problem: starvation of low-priority processes.
Solution: aging.

Deadlock

Deadlock occurs when processes wait forever for resources.

Four necessary conditions:

1. Mutual exclusion
2. Hold and wait
3. No preemption
4. Circular wait

If any one condition is removed, deadlock can be prevented.

Semaphore

Semaphore is a synchronization variable.

Types:

- Binary semaphore: 0 or 1
- Counting semaphore: integer count

Operations:

- wait/P/down
- signal/V/up

Mutex

Mutex is a lock that ensures only one thread enters a critical section.

Difference:

- Mutex has ownership.
- Semaphore signals resource count.

Critical section

Part of code accessing shared resource.

Requirements:

- mutual exclusion,
- progress,
- bounded waiting.

Memory management

Paging

Memory divided into fixed-size pages and frames.

Advantages:

- avoids external fragmentation.

Disadvantage:

- can have internal fragmentation.

Segmentation

Memory divided into logical variable-size segments.

Examples:

- code segment,
- data segment,
- stack segment.

Disadvantage:

- external fragmentation.

Virtual memory

Virtual memory lets a program use more memory than physical RAM by using disk.

Page fault:

> Required page is not in RAM and must be loaded from disk.

Too many page faults cause thrashing.

File system

File system manages file storage, directories, permissions, and metadata.

Common MCQ traps

- Deadlock prevention removes at least one necessary condition.
- Starvation is indefinite waiting, not always deadlock.
- Round Robin is preemptive.
- Paging uses fixed-size blocks.
- Segmentation uses variable-size logical blocks.
- Threads share code/data but have separate stack and registers.