



VIT[®]
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering

Fall Semester 2024-25

CAT I

SLOT: B2+TB2

Programme Name & Branch: B.Tech & BCB, BCE, BCI, BCT, BDS, BKT

Course Name & Code: BCSE303L & Operating Systems

ANSWER KEY

1. (i) Comparison of Layered approach and Micro-kernel approach: (5 Marks)

Features	Monolithic	Layered Operating System
Definition	It is one in which the complete operating system operates in the kernel space.	A layered operating system that has divided into multiple layers, and each layer serves as a specific task.
Number of levels	There are mainly three layers in the monolithic operating system.	There are multiple layers in layered operating systems.

Monolithic Operating System:

Advantages:

1. The monolithic kernel runs quickly because of memory management, file management, process scheduling, etc. These are all implemented in the same address space.
2. Its structures are easy and simple. The kernel contains all of the components required for processing.
3. All of the components may interact directly with each other's and also with the kernel.
4. It works better for performing smaller tasks because it may handle limited resources.

Disadvantages:

1. Monolithic OS has more tendency to generate errors and bugs. It's because user programs use the same address spaces as the kernel.
2. It isn't easy to port code written in the monolithic operating system.
3. It is very difficult to add and remove features from a monolithic operating system. All of the code must be modified and recompiled to add or remove a feature.

Layered operating system:

Advantages:

1. Easy Debugging

It is very simple to debug because the layers are discrete. If an error happens in the CPU scheduling layer, the developer may only debug that layer.

2. Modularity

This design supports modularity because each layer only executes tasks it is scheduled to perform.

3. Abstraction

Each layer is concerned with its own set of functions. As a result, the functions and implementations of the other layer are abstract to it.

4. Easy update

A modification in one layer does not affect the other layers.

Disadvantages

1. Complex and better implementation

Layer layout is important because a layer can utilize the services of the layers below it. For example, the backup storage layer uses the memory management layer's services, so it must be stored beneath the memory management layer.

2. Slower in execution

When one layer wishes to interact with another, it sends a request that must traverse all layers between the two layers to be fulfilled. It enhances response time, which is faster than the Monolithic system. As a result, increasing the number of layers may lead to a very inefficient design.

1. (ii) Design Issues: (5 Marks)

- Efficiency
- Robustness
- Flexibility
- Portability
- Compatibility
- Memory Management
 - Memory Protection: Ensure that processes cannot interfere with each other's memory space to maintain system stability and security.
 - Allocation and Deallocation of Memory
- Concurrency and Synchronization
 - Concurrency Control
 - Synchronization
- Security and Protection
 - User Authentication and Authorization
 - Data Security
- Resource Management
 - Resource Allocation

- User Interface
 - Command-Line Interface (CLI)
 - Graphical User Interface (GUI)
- Error Handling and Recovery
 - Fault Tolerance
 - Crash Recovery

2. Multithreading: (10 Marks)

Definition – Multithreading: (2 Marks)

Multithreading allows the application to divide its task into individual threads. In multi-threads, the same process or task can be done by the number of threads. Multithreading is a feature in operating systems that allows a program to do several tasks at the same time.

Advantages of multi-threaded process over single threaded process: (2 Marks)

- **Concurrency:** Ability to perform multiple tasks simultaneously.
- **Responsiveness:** Keeps applications responsive, especially with user interfaces, by handling background tasks without freezing.
- **Resource Sharing:** Threads share the same memory space, making communication and data sharing more efficient.
- **Improved Performance:** Can lead to performance gains for parallelizable tasks by utilizing multiple CPU cores.
- **Simplified Program Structure:** Easier design for concurrent tasks compared to managing multiple processes.
- **Better Resource Utilization:** Makes more efficient use of CPU by continuing work while other threads wait for I/O operations.
- **Scalability:** Can leverage additional CPU cores and hardware improvements more effectively

Multi-threading is related to concurrency and parallelism – Justification: (2 Marks)

- **Concurrency:** Multi-threading allows a single process to manage multiple tasks at once by interleaving their execution. This means threads can start, run, and complete in overlapping time periods, even if not simultaneously. It improves the responsiveness and structure of applications by enabling tasks to progress independently.
- **Parallelism:** Multi-threading enables parallelism by executing multiple threads simultaneously on multiple CPU cores. This means threads can perform computations at the same time, leading to faster execution of tasks that can be divided and executed concurrently.

Different Multi-threading Models: (4 Marks)

- **User-Level Threads:** Managed entirely by user-level libraries or runtime environments, without direct support from the operating system.
- **Kernel-Level Threads:** Managed and scheduled by the operating system kernel.
-  **Many-to-One Model:** Multiple user-level threads are mapped to a single kernel thread.
-  **One-to-One Model:** Each user-level thread is mapped to a separate kernel thread.
-  **Many-to-Many Model:** Multiple user-level threads are mapped to multiple kernel threads.

3. (i) Multi-user and Multi-programming OS – Security threats and Measures: (5 Marks)

Security Threats:

- Unauthorized Access
- Malware
- Data Theft or Loss
- Viruses
- Privilege Escalation

Security Measures:

- User Mode and Kernel mode of execution
- Access Control Policies
- Authentication
- Authorization
- Logging
- Backup and Recovery

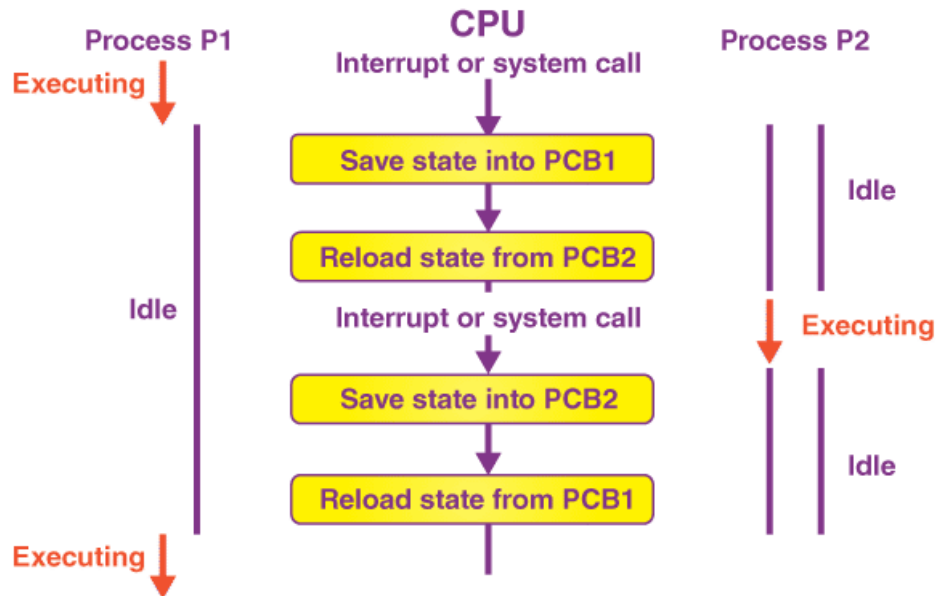
3. (ii) Process Control Block: (5 Marks)

The **Process Control Block (PCB)** is used by an operating system to manage and track information about processes. In brief, its key uses are:

1. **Process Identification:** Uniquely identifies each process with a Process ID (PID) and other identifiers.
2. **Process State Management:** Tracks the current state of the process (e.g., running, waiting, ready).
3. **CPU Scheduling:** Stores CPU register values, the Program Counter (PC), and scheduling priorities needed for context switching and scheduling.
4. **Memory Management:** Manages memory allocation details, including base and limit registers or page tables.
5. **Process Control Information:** Handles process control tasks, including state transitions and inter-process communication.
6. **I/O Management:** Tracks I/O resources allocated to the process and their status.
7. **Accounting Information:** Records resource usage and process execution times for accounting purposes.

Role of PCB in Context Switching:

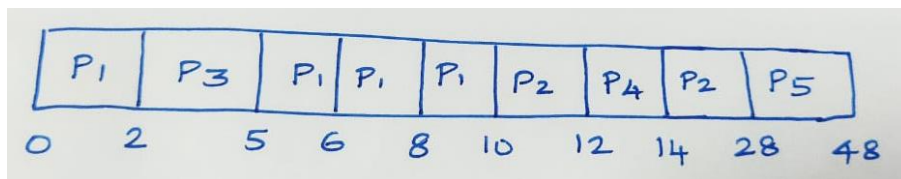
- **Saving State:** During a context switch, the operating system saves the current process's state (CPU registers, Program Counter, etc.) into its PCB. This ensures that the process can resume execution from the exact point it was interrupted.
- **Loading State:** The PCB of the next process to be executed is accessed to load its saved state (CPU registers, Program Counter, etc.) into the CPU. This prepares the CPU to resume execution of that process.



4. CPU Scheduling - Solution: (10 Marks)

Preemptive SJF: (3 Marks)

Gantt Chart:



Waiting Time:

P1=3

P2=6

P3=0

P4=0

P5=20

Average Waiting Time = 5.8 ms

Turnaround Time:

P1=10

P2=22

P3=3

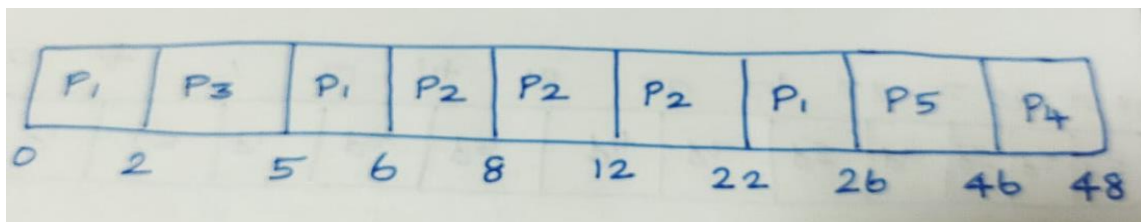
P4=2

P5=40

Average Waiting Time = 15.4 ms

Preemptive Priority: (3 Marks)

Gantt Chart:



Waiting Time:

P1=19

P2=0

P3=0

P4=34

P5=18

Average Waiting Time = 14.2 ms

Turnaround Time:

P1=26

P2=16

P3=3

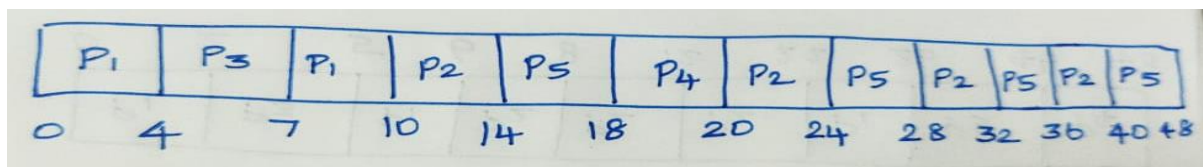
P4=36

P5=38

Average Waiting Time = 23.8 ms

Round Robin: (4 Marks)

Gantt chart:



Waiting Time:

P1=3

P2=18

P3=2

P4=6

P5=20

Average Waiting Time = 9.8 ms

Turnaround Time:

P1=10

P2=34

P3=5

P4=8

P5=40

Average Waiting Time = 19.4 ms

5. CPU Scheduling - Solution: (10 Marks)**CPU Burst Time Calculation: (2 Marks)**

Process	Arrival Time (ms)	Total Execution Time (ms)	CPU Burst Time (ms) 60 % of TET
P0	4	20	12
P1	5	10	6
P2	6	30	18
P3	2	8	4.8
P4	3	12	7.2
P5	10	40	24

FCFS: (4 Marks)

The diagram shows a Gantt chart for FCFS scheduling. The processes are executed in the order of their arrival times: P3 (at 2), P4 (at 3), P0 (at 4), P1 (at 5), P2 (at 6), and P5 (at 10). The completion times are marked below the bars: 6.8 for P3, 14 for P4, 26 for P0, 32 for P1, 50 for P2, and 74 for P5.

Waiting Time:

$$\begin{aligned}
 P_0 &= 14 - 4 = 10 \\
 P_1 &= 26 - 5 = 21 \\
 P_2 &= 32 - 6 = 26 \\
 P_3 &= 2 - 2 = 0 \\
 P_4 &= 6.8 - 3 = 3.8 \\
 P_5 &= 50 - 10 = 40
 \end{aligned}$$

Turn Around Time:

$$\begin{aligned}
 P_0 &= 10 + 12 = 22 \\
 P_1 &= 21 + 6 = 27 \\
 P_2 &= 26 + 18 = 44 \\
 P_3 &= 0 + 4.8 = 4.8 \\
 P_4 &= 3.8 + 7.2 = 11 \\
 P_5 &= 40 + 24 = 64
 \end{aligned}$$

AWT = 16.8 ms **ATAT = 28.8 ms**

Non Preemptive SJF: (4 Marks)

CPU idle	P ₃	P ₁	P ₄	P ₀	P ₂	P ₅	
0	2	6.8	12.8	20	32	50	74

Waiting Time:

$$P_0 = 20 - 4 = 16$$

$$P_1 = 6.8 - 5 = 1.8$$

$$P_2 = 32 - 6 = 26$$

$$P_3 = 2 - 2 = 0$$

$$P_4 = 12.8 - 3 = 9.8$$

$$P_5 = 50 - 10 = 40$$

$$AWT = 15.6 \text{ ms}$$

Turn Around Time

$$P_0 = 16 + 12 = 28$$

$$P_1 = 1.8 + 6 = 7.8$$

$$P_2 = 26 + 18 = 44$$

$$P_3 = 0 + 4.8 = 4.8$$

$$P_4 = 9.8 + 7.2 = 17$$

$$P_5 = 40 + 24 = 64$$

$$ATAT = 27.6 \text{ ms}$$