

ABHINAV JHA

2301010211

B.TECH CSE CORE - 4

Page No.: 211

## Capstone Assignment

### Part - A

1. An OS manages hardware resources and provides an interface for user programs. It handles process scheduling, memory allocation, file management & I/O control. Without an OS, every application would need to manage hardware directly, which is inefficient and unsafe.

2. Monolithic Kernel: Fast but less modular as most services run in kernel space.

Layered Kernel: Organises OS components in layers improving maintainability.

Microkernel: Keeps core services like scheduling & IPC in kernel and moves other to user space, increasing reliability & security.

For distributed systems, microkernel is preferable due to fault isolation.

3. Threads share memory space, so switching between threads is faster and cheaper than process switching which requires full context and memory space change. Thread creation & synchronization is lighter, making threading more efficient for parallel work within same application. Processes offer better isolation but higher overhead.

4. First Fit  $P_1 \rightarrow 20 \text{ MB}$   $P_3 \rightarrow 10 \text{ MB}$   
 $P_2$  cannot fit remaining fragments

Best Fit  $P_1 \rightarrow 15 \text{ MB}$   $P_3 \rightarrow 10 \text{ MB}$   
 $P_2 \rightarrow 20 \text{ MB}$

Best fit reduces fragmentation here because it matches block sizes closely to request sizes.

Page No: \_\_\_\_\_

5	Process	Burst Time (ms)	Arrival Time (ms)
	P <sub>1</sub>	5	0
	P <sub>2</sub>	3	1
	P <sub>3</sub>	8	2
	P <sub>4</sub>	6	3

a) Gantt chart : FCFS 

P <sub>1</sub>	P <sub>2</sub>	P <sub>3</sub>	P <sub>4</sub>
----------------	----------------	----------------	----------------

SJF 

P <sub>1</sub>	P <sub>2</sub>	P <sub>4</sub>	P <sub>3</sub>
----------------	----------------	----------------	----------------

Round Robin 

P <sub>1</sub>	P <sub>2</sub>	P <sub>1</sub>	P <sub>3</sub>	P <sub>4</sub>	P <sub>2</sub>	P <sub>4</sub>
----------------	----------------	----------------	----------------	----------------	----------------	----------------

 quantum = 4

b) FCFS  
Completion Time:

$$P_1 = 5, P_2 = 8, P_3 = 16, P_4 = 22$$

$$\text{Turnaround} = \text{Completion} - \text{Arrival}$$

$$P_1 = 5 - 0 = 5$$

$$P_2 = 8 - 1 = 7$$

$$P_3 = 16 - 2 = 14$$

$$P_4 = 22 - 3 = 19$$

$$\text{Waiting} = \text{Turnaround} - \text{Burst}$$

$$P_1 = 5 - 5 = 0$$

$$P_3 = 14 - 8 = 6$$

$$P_2 = 7 - 3 = 4$$

$$P_4 = 19 - 6 = 13$$

$$\text{Avg. waiting} = \frac{0 + 4 + 6 + 13}{4} = \frac{23}{4} = 5.75 \text{ ms}$$

$$\text{Avg. turnaround} = \frac{5 + 7 + 14 + 19}{4} = \frac{45}{4} = 11.25 \text{ ms}$$

SJF

$$\text{Completion Time } P_1 = 5, P_2 = 8, P_4 = 14, P_3 = 22$$

$$\text{Turnaround : } P_1 = 5 - 0 = 5 \\ P_2 = 8 - 1 = 7$$

$$P_3 = 22 - 2 = 20 \\ P_4 = 14 - 3 = 11$$

$$\text{Waiting } P_1 = 5 - 5 = 0 \\ P_2 = 7 - 3 = 4$$

$$P_3 = 20 - 8 = 12 \\ P_4 = 11 - 6 = 5$$

$$\text{Avg. waiting time} = \frac{0 + 4 + 12 + 5}{4} = \frac{21}{4} = 5.25 \text{ ms}$$

$$\text{Avg. TAT} = \frac{5 + 7 + 20 + 11}{4} = \frac{43}{4} = 10.75 \text{ ms}$$

Round Robin

quantum = 4ms

Completion Time

$$P_1 = 16$$

$$P_3 = 20$$

$$P_2 = 7$$

$$P_4 = 22$$

$$\text{Turnaround Time} = P_1 = 16 - 0 = 16$$

$$P_3 = 20 - 2 = 18$$

$$P_2 = 7 - 1 = 6$$

$$P_4 = 22 - 3 = 19$$

$$\text{Waiting Time} = P_1 = 16 - 5 = 11 \\ P_2 = 6 - 3 = 3$$

$$P_3 = 18 - 8 = 10 \\ P_4 = 19 - 6 = 13$$

$$\text{Average waiting time} = \frac{11 + 3 + 10 + 13}{4} = \frac{37}{4} = 9.25 \text{ ms}$$

$$\text{Avg. TAT} = \frac{16 + 6 + 18 + 19}{4} = \frac{59}{4} = 14.75 \text{ ms}$$

g) Algorithm	Avg. waiting (ms)	Avg. TAT (ms)	Completion Time (ms)
FCFS	5.75	1.25	22
SJF	5.25	6.75	22
Round Robin (q=4)	9.25	4.75	22

If goal is minimize avg TAT/waiting then SJF is best.

- If system is time-sharing where responsiveness & fairness matter then Round-Robin is preferable.
6. Banker's algorithm avoids deadlock by checking if resource allocation keeps system safe. Detection & Recovery approach allows deadlock to occur but periodically checks for cycle in wait-for graph & aborts / rolls back a selected process to recover.
7. Use semaphores: 'empty' for free slots, 'full' for filled slots, and 'mutex' for exclusive access. Producer waits on empty, inserts item, signals full. Consumer waits on full, removes item, signals empty. Ensures synchronization & prevents race conditions.
8. Sequence: 2, 1, 4, 2, 3, 4, 3 ; Frames = 3 )  
FIFO replaces the oldest loaded page; LRU replaces least recently used page. Both result in about 5 page faults here. LRU generally performs better in real usage due to locality of reference.
9. a) Two critical issues are consistency across multiple nodes and fault tolerance.  
b) Using replication, distributed metadata servers, and cache coherence mechanisms (leases/callbacks) improves reliability and performance.
10. Synchronous checkpointing captures consistent system state but pauses processes, while asynchronous allow independent checkpoints but requires message logging to avoid inconsistent recovery.

Pseudo sequence: Coordinator  $\rightarrow$  request checkpoint  $\rightarrow$  each process flush and write local state  $\rightarrow$  acknowledge  $\rightarrow$  coordinator commits checkpoint

- 11a) Use priority preemptive scheduling (real time): assign highest priority to security interrupts.
- b) IPC: message queues or MQTT pub-sub for networked devices, shared memory locally if low-latency.
12. System calls shift ~~ex~~ execution from user mode to kernel mode for safe hardware access. Functions like `open()`, `write()` invoke kernel routines, allowing OS to manage file securely.

Get into  
22/11/25