Name → AJAY NATH JHA Roll. No → 23010170
Course → B. Tech CSE Core
Section → C Semester → V

Page No. 01
Date 03 10 25

Subject :→ Operating System

Assignment - 1 :→

Short Answer Type (Part-A) :→

(1.) Modern systems continue to rely on Operating Systems (OS) because hardware alone cannot manage resources or support user-level applications. The OS abstracts complex hardware operations - like memory access, CPU scheduling, I/O handling into simple, usable interfaces. It ensures security, multitasking, resource sharing, and stability. Without an OS, every application would need to handle low-level operations independently, which is impratical and error-prone.

(2.) A Real-Time Operating System (RTOS) is most suitable because wearable health devices require predictable, time-bound responses. Heart-rate monitoring demands minimal latency, reliable sensor reading, and quick decision-making. RTOS ensures deterministic scheduling and efficient management of limited hardware resources, which is essential for medical wearables.

(3.) I would avoid the Layered Architecture because each layer must communicate only with adjacent layers, with increases overhead and latency. In performance-critical systems, this structured communication can slow down execution. Monolithic and microkernel designs, while having their own trade-offs, generally offer faster system calls (monolithic) or higher modularity (microkernel).

(4.) I refuse the claim. OS structure directly affects performance, maintainability, reliability, and security. For example, monolithic kernels are faster but slower due to message passing. The structure impacts how efficiently the OS handles system calls, process scheduling, and device management. Thus, Structure matters significantly beyond simply 'running processes'.

(5.) Process Switching error debugging :→

(i) The Process Control Block stores register values, program counter, CPU state, memory pointers, etc. By analyzing it, we can detect misinitialized registers, incorrect process states, or corruption in saved CPU context - common causes of switching errors.

(ii) Context switching saves the current process's state (registers, PC, flags) into its PCB and loads the next process's state from its PCB. It also updates scheduling queues and may trigger I/O or event-based transitions.

(iii) A blocking synchronous system call is ideal because the process must wait until I/O resources are allocated safely. Synchronous operations ensure system maintains consistent state before resuming execution.

(Part-B) :→ Application / Numerical Based.

(6.) Context Switching Time Calculation.

Given: Save State = 2ms, Load State = 3ms, Scheduler Overhead = 1ms

(a) Total Context Switching Time = (Save + Load + Scheduler Overhead)
= (2+3+1) = 6 ms.

(b) Impact on Multitasking →
Higher switching time reduces CPU efficiency since the processor spends more time switching and less time executing. It increases overall turnaround time and may degrade real-time performance.

(7.) Thread Efficiency Check:

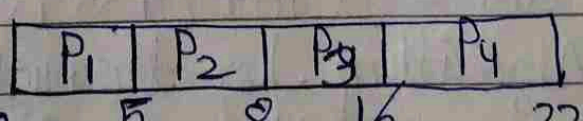
(a) Given, Single-threaded time = 40 seconds,
Threads per process = 4 (ideal conditions)
Estimated Executed Time:
If perfectly parallel: 40 sec ÷ 4 = 10 seconds

(b) Multithreading allows tasks to run concurrently by sharing process resources while reducing context-switch overhead. It improves CPU utilization and shortens execution time for parallelizable tasks.

(8.) CPU Scheduling (FCFS, SJF, RR)
Process and Burst Times:
$P_1 = 5ms$, $P_2 = 3ms$, $P_3 = 8ms$, $P_4 = 6ms$

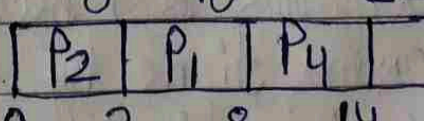
(a) Gantt Chart →

(i) FCFS :



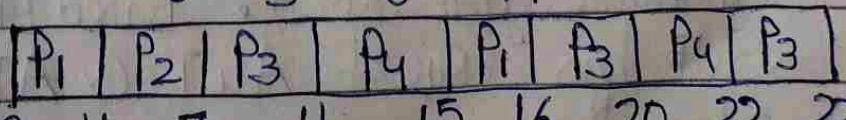
($P_1 \rightarrow P_2 \rightarrow P_3 \rightarrow P_4$)

(ii) Non-preemptive SJF:



($P_2 \rightarrow P_1 \rightarrow P_4 \rightarrow P_3$)

(iii) Round Robin
Quantum = 4ms



(b) Average Waiting & Turn Around Time

(i) FCFS

Process-id	B.T	W.T	T.A.T
P_1	5	0	5
P_2	3	5	8
P_3	8	8	16
P_4	6	16	22

avg W.T = (0+5+8+16)/4 = 7.25ms

T.A.T avg = (5+8+16+22)/4 = 12.75ms

(ii) SJF Non-Preemptive

Process	B.T	W.T	T.A.T
P_2	3	0	3
P_1	5	3	8
P_4	6	8	14
P_3	8	14	22

W.T avg = (0+3+8+14)/4 = 6.25ms

T.A.T avg = (3+8+14+22)/4 = 11.75ms

(iii) Round Robin (quantum = 4)

Waiting times:

$P_1 = 0+(15-5) = 10$; $P_2 = 4$;

$P_3 = (7-0)+(15-8) = 14$; $P_4 = (11-0) = 11$

W.T avg = (10+4+14+11)/4 = 9.75ms

Turn Around times:

$P_1 = 16$, $P_2 = 7$, $P_3 = 22$, $P_4 = 22$

T.A.T avg = (16+7+22+22)/4 = 16.75ms

(C) Best Algorithm for throughput & turnaround → SJF performs best because it minimizes waiting & turnaround time by always choosing shortest available job first. Based on our calculations, SJF has lowest AWT & ATAT among three algorithms.

(9.) Virtualization & IoT Scheduling →

(i) Cloud Migration:

(a) Best OS Architecture →

A Microkernel architecture is ideal because it offers: High modularity, Strong isolation between services, Better fault tolerance & enhanced scalability for distributed cloud systems.

Microkernels reduce attack surface, improving security - crucial in large cloud deployments.

(b) How Virtual Machine Help →

It provide strong isolation, preventing one service from affecting another, Allow easy resource allocation & optimization. It enable migration, snapshots, and rapid deployment. It improve system reliability through sandboxing.

(ii) Smart Home IoT System:

(a) How OS uses scheduling & IPC →

The OS assigns higher priority to tasks like intrusion detection so they receive CPU immediately. Lower-priority tasks (e.g., adjusting lighting) are scheduled only when critical processes finish. IPC mechanisms like message queues and signals ensure devices communicate efficiently without delays or dta loss.

(b) Suitable Scheduling Algorithms →

Priority Scheduling → It gives top priority to security tasks. Rate Monotonic Scheduling (RMS): Good for periodic sensor tasks. Earliest Deadline First (EDF): Ensures time-critical tasks meet deadlines. These algorithms guarantee timely responses, which are essential for a smart home environment.