OS Assignment - 1

Ayush Gupta
2301010206
B-Tech CSE - D

Part-A

1. a) OS acts as an intermediary between hardware and software.

b) - It manages resources (CPU, memory, I/O) efficiently.
- Provides essential services like process scheduling, file systems, and security.
- Provides essential services like process scheduling, file systems, and security

2. - Real-Time Operating System (RTOS) is best.
- RTOS ensures deterministic and timely response, which is crucial for health monitoring.
- Light weight, power-efficient, and can handle sensor inputs reliably.

3. - Avoids Monolithic kernel.
- Because it has all services in one large module, leading to:
  - large codebase and poor modularity.
  - Higher chance of bugs affecting the whole system.
  - Difficult debugging and slower context switching.

4. - Refute
- OS structure affects stability, performance, maintainability, and security.
- Poor structure can cause bottlenecks, crashes, or vulnerability exploits even if processes run.
- Structured design ensures modularity and scalability.

5. 1) PCB stores process state (registers, program counter, flags).

- Analyzing PCB shows if registers or states are misinitialized during switch.

ii) Context switching involves:
- Saving the current process state in its PCB.
- Loading the next process's state from its PCB.
- Updating CPU registers, program counter, and memory mappings.

iii) Use a blocking synchronous system call.
- Blocking ensures the process waits for I/O allocation to complete.
- Synchronous ensures tasks proceed only after I/O allocation succeeds (avoids race conditions).

Part-B

6. - Save state time = 2ms
- Load state time = 3ms
- Scheduler overhead = 1ms

a) Total context switching time (T_cs):

$$T\_cs = (\text{Save state}) + (\text{Load state}) + (\text{Scheduler overhead})$$
$$= 2ms + 3ms + 1ms$$
$$T\_cs = 6ms$$ Ans

b) - This overhead adds to every switch, reducing effective CPU time for actual processes.

- Frequent context switches → lower multitasking efficiency

1. Given - Total execution time = 40 sec

If the task is perfectly parallelizable and there are N threads ^ total scale

Q1. T-Parallel = T-single / N

- For N = 2 threads → T = 40/2 = 20 sec
- For N = 4 threads → T = 40/4 = 10 sec
- For N = 8 threads → T = 40/8 = 5 secs.

2. Realistic : Amdahl's law

If a function p of the program is parallelizable and (1-p) is serial, max speedup S(N) using N processors/threads is:

$$S(N) = \frac{1}{(1-p) + P/N}$$

$$T_{parallel} = \frac{T_{single}}{S(N)}$$

80% parallelizable → p = 0.8, serial = 0.2

Compute speedup for N = 4

$$S(4) = \frac{1}{\left(0.2 + \frac{0.8}{4}\right)} = \frac{1}{0.2 + 0.2} = \frac{1}{0.4} = 0.25$$

T-parallel = 40/2.5 = 16 sec.

Compare

- Ideal 4-thread time = 10 sec
- Realistic (p = 0.8) 4-thread time = 16 sec

Explanation for benefits

- Threads reduce total time ~~on~~ when substantial parts are parallelizable.
- Overheads (synchronization, context switching, ~~co~~ cache coherence) reduce ideal gains - use concurrency when it truly benefits.

8. Process Burst time

Process	Burst time
P1	5
P2	3
P3	8
P4	6

All arrive at time 0.

a) 1. FCFS (First come, First served)

Order: P1 → P2 → P3 → P4

- P1: start 0, burst 5 → completes 0+5 = 5
- P2: start 5, burst 3 → completes at 5+3 = 8
- P3: Start 8, burst 8 → completes at 8+8 = 16
- P4: start 16, burst 6 → completes at 16+6 = 22.

Gantt (timeline showing [start - end] for each:

|P1 [0-5] | P2 [5-8] | P3 [8-16] | P4 [16-22]|

2. Non - preemptive SJF (Shortest Job first)

Order by burst ascending: P2(3), P1(5), P4(6), P3(8)

- P2: start 0 → completes 0+3 = 3.
- P1: start 3 → completes at 3+5 = 8

- P4: start 8 → completes at 8+6 = 14
- P3: start 14 → completes 14+8 = 22.

Gantt:
| P2 [0-3] | P1 [3-8] | P4 [8-14] | P3 [14-22] |

2. Round Robin (Quantum Q = 4ms)

~~Steps~~ Simulate cycles (all arrive at 0). Keep remaining times:

Initial remaining = (P1: 5, P2: 3, P3: 8, P4: 6)

- Cycle 1.
  - Time 0-4: P1 runs for 4 (remaining P1 = 5-4 = 1). Time now = 4.
  - Time 4-7: P2 needs 3 (≤4), runs 3, completes at 7. (P2 remaining = 0).
  - Time 7-11: P3 runs 4 (remaining = 8-4 = 4), Time =
  - Time 11-15: P4 runs 4 (remaining = 6-4 = 2). Time = 15

Remaining after cycle 1: P1 : 1, P2 : 0 (done), P3: 4, P4: 2.

- Cycle 2
  - Time 15-16: P1 runs 1, completes at 16.
  - Time 16-20: P3 runs 4, completes at 20.
  - Time 20-22: P4 runs 2, completes at 22.

Gantt (with time slices)

| P1 [0-4] | P2 [4-7] | P3 [7-11] | P4 [11-15] | P1 [15-16] | P3 [16-20] | P4 [20-22] |

b) • ~~Tur~~ Turn around time (TAT) = Completion time − Arrival time. (Arrival = 0 for all)

• Waiting time (WT) = Turnaround Time − Burst time.

FCFS

• WT : (0+5 +8+16) = 29 → 29/4 = 7.25

• TAT: (5+8+16+22) = 51 → 51/4 = 12.75

SJF

• WT: (3+0+14+8) = 25 → 25/4 = 6.25

• TAT: (8+3+22+14) = 47 → 47/4 = 11.75

RR

• Completion : P1 = 16, P2 = 7, P3 = 20, P4 = 22

• WT = TAT − Burst

• TAT: (16+7+20+22) = 65 ⇒ 65/4 = 16.25

• WT: (11+4+12+16) = 43 → 43/4 = 10.75.

c) − SJF gives lowest average waiting time and turn around time.

− Hence it balances throughput and turnaround best among the three.

9) Cloud migration OS architecture

Architecture:

• Microkernel − because:

  • Minimal core increases security.

  • Services run in user mode, improving fault isolation

• Easy scalability by adding services as modules.

Virtual Machines Role:
• Provide isolated environments for each service.
• Enable resource pooling and dynamic allocation.
• Simplify system management and migration without affecting others.

(iii) Smart home system scheduling

Process scheduling + IPC:
- OS assigns higher priority to critical tasks (Intrusion alerts).
- Lower priority for less critical tasks (light, thermostat).
- IPC (signals, message queues) ensures quick communication between processes.

Suitable algo:
• Priority Scheduling (Preemptive) - ensures critical tasks run immediately.
• Earliest Deadline First (EDF) - good for real-time constraints.
• Round Robin (for non-critical tasks) to ensure fairness.