Topic ..... Date ...../...../.....

PAGE NO.

OPERATING SYSTEMS ASSIGNMENT - 1

RUCCHIKA KAPOOR (2301010240)

PART-A

Q1 Despite the evolution of hardware, why do modern systems still rely heavily on Operating Systems?

Modern systems still rely heavily on OS because:

→ RESOURCE MANAGEMENT:
The OS efficiently manages hardware resources (CPU, memory, storage, I/O devices) and allocates them to different programs.

→ USER AND APPLICATION INTERFACE:
The OS provides a convenient interface b/w hardware and users/applications, enabling portability, multitasking and security.

Q2 You are asked to design an OS for a wearable health device that monitors heart rate. Which type of OS would be most suitable and why?

REAL TIME OPERATING SYSTEM (RTOS)
RTOS ensures timely, predictable, and reliable response to inputs like ~~heat~~ heart rate signals processes data with low latency, provides efficient resource management on small, low-power hardware critical for health monitoring devices.

Teacher's Signature.....

Topic ..... Date ...../...../.....

PAGE NO.

Qs Given the choice to build a new OS kernel for a performance-critical environment, which structure would you avoid (Monolithic, Layered, Microkernal), and why?

Avoid a Monolithic kernel, while it give fast system calls, they lack modularity and are harder to maintain/debug. A bug in one service can crash the whole system, making them unreliable for critical systems.

Qy A developer claims that OS structure doesn't matter as long as processes run. Support or refute this claim with reasoning

Refute the claim because OS structure directly impacts performance, reliability, scalability and security.
eg: microkernal isolates services for fault tolerance, while a layered structure improves maintainability. Just "running processes" isn't enough if the system is slow, unsecure or unstable.

Qs While debugging, you find a process switching error.
i) Explain how analyzing the Process Control Block (PCB) can point to uninitialized ~~reg~~ registers or incorrect states

The PCB stores CPU registers, program counter, state and memory info. By examining it, we can detect uninitialized registers, wrong state flags, incorrect program counter values that causes faulty switching

Teacher's Signature.....

Topic ..... Date ...../...../.....

PAGE NO.

ii) If a task is moved unexpectedly from running to waiting, what precisely does context switching involve?

When a task unexpectedly moves from running to waiting, context switching saves the current process state (registers, program counter, PCB updates) and loads the state of the next process. It ensures execution resumes correctly later

iii) The OS needs to allocate I/O resources mid-execution. Which type of system calls would you use (eg: blocking, non-blocking, synchronous, asynchronous) and why?

Use an asynchronous, non-blocking system calls because this allows the process to continue execution while the I/O is allocated in the background, preventing the CPU from idling

PART-B

Q6 Context switching time calculation:

given:

- save state: 2 ms
- load state: 3 ms

scheduler overhead = 1 ms

a) Calculate total context switching time.

total time = 2+3+1 = 6 ms

b) Explain its impact on multitasking performance.

→ context switching is pure overhead (no useful work is done during this time)

Teacher's Signature.....

Topic ..... Date ...../...../.....

PAGE NO.

→ Higher switching time reduces CPU efficiency, as more time is spent switching than executing processes.

→ In multitasking, frequent context switches with high overhead can slow down throughput and increase response time.

Q7 Thread efficiency check:
- total time in single-threaded = 40 sec.
- threads per process (ideal conditions)

Multithreading is used with n threads per process

- estimate execution time.
- explain how multithreading improves performance.

Execution time estimate:
In ideal conditions (perfect parallelism, no overhead)

$$T_{multi} = \frac{T_{single}}{n} = \frac{40}{n} \text{ seconds.}$$

eg: if $n = 2 \rightarrow 20$ sec
if $n = 4 \rightarrow 10$ sec
if $n = 8 \rightarrow 5$ sec

How multi-threading improves performance

→ It improves performance by running tasks in parallel, reducing execution time.

→ It keeps the CPU busy even during I/O waits, avoiding idle time.

Teacher's Signature.....

Topic ..... Date ...../...../.....

PAGE NO.

→ Threads share resources, making execution faster and more efficient

Q8 Given these processes and burst times:

Process	P1	P2	P3	P4
Time (ms)	5	3	8	6

a) Draw Gantt charts for:

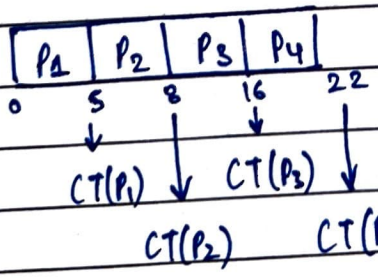
FCFS:

Process	Arrival time (AT)	Burst Time (BT)	Completion Time (CT)	Waiting Time (WT)	TAT
P_1	0	5	5	5-5 = 0	5-0 = 5
P_2	0	3	8	8-3 = 5	8-0 = 8
P_3	0	8	16	16-8 = 8	16-0 = 16
P_4	0	6	22	22-6 = 16	22-0 = 22

WT = Turnaround − Burst (TAT − BT)

TAT = Completion − Arrival (CT − AT)

Gantt chart,



Avg. waiting time = (0+5+8+16)/4 = 7.25 ms

Avg. turnaround time = (5+8+16+22)/4 = 12.75 ms.

Teacher's Signature.....

Topic .....

Date ...../...../.....

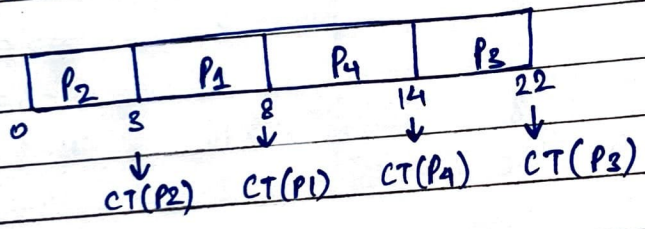
b) Non - preemptive SJF :

Process	AT	BT	CT	WT	TAT
P_1	0	5	8	8-5 = 3	8 = 8-0
P_2	0	3	3	3-3 = 0	3 = 3-0
P_3	0	8	22	22-8 = 14	22 = 22-0
P_4	0	6	14	14-6 = 8	14 = 14-0

TAT = CT - AT

WT = TAT - BT

Gantt chart,



Avg. waiting time = (3+0+14+8)/4 = 6.25 ms

Avg. turnaround time = (8+3+22+14)/4 = 11.75 ms

c) Round Robin (quantum = 4ms)

Process	AT	BT	CT	WT	TAT
P_1	0	5	16	16-5 = 11	16-0 = 16
P_2	0	3	7	7-3 = 4	7-0 = 7
P_3	0	8	20	20-8 = 12	20-0 = 20
P_4	0	6	22	22-6 = 16	22-0 = 22

WT = TAT - BT

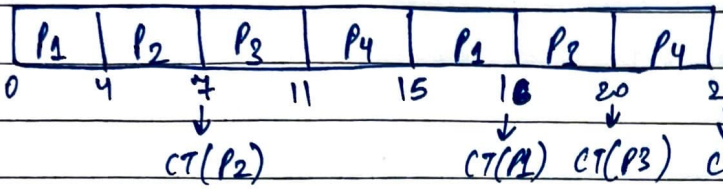
TAT = CT - AT

Teacher's Signature.....

Topic ..... Date ...../...../.....

PAGE NO.

Gyantt chart,



Avg. time (waiting) = (11+4+12+16)/4 = 10.75 ms
Avg. turnaround = (16+7+20+22)/4 = 16.25 ms

* Non-preemptive SJF is best because it gives the lowest avg. waiting time (6.25 ms) and turnaround time (11.75 ms), while throughput remains the same (4/22) in all methods. It balances turnaround and throughput better than FCFS and RR.

Q9 i) Virtualized cloud migration

a) MICROKERNAL ARCHITECTURE would be the best choice because,
high scalability → services (like drivers, file system, networking) run in user space and can be extended / updated easily.
High security and fault isolation → failure in one service doesn't crash the entire system.

b) Role of Virtual Machines in migration

- Isolation:
each VM runs its own OS, preventing one service failure or attack from affecting others.

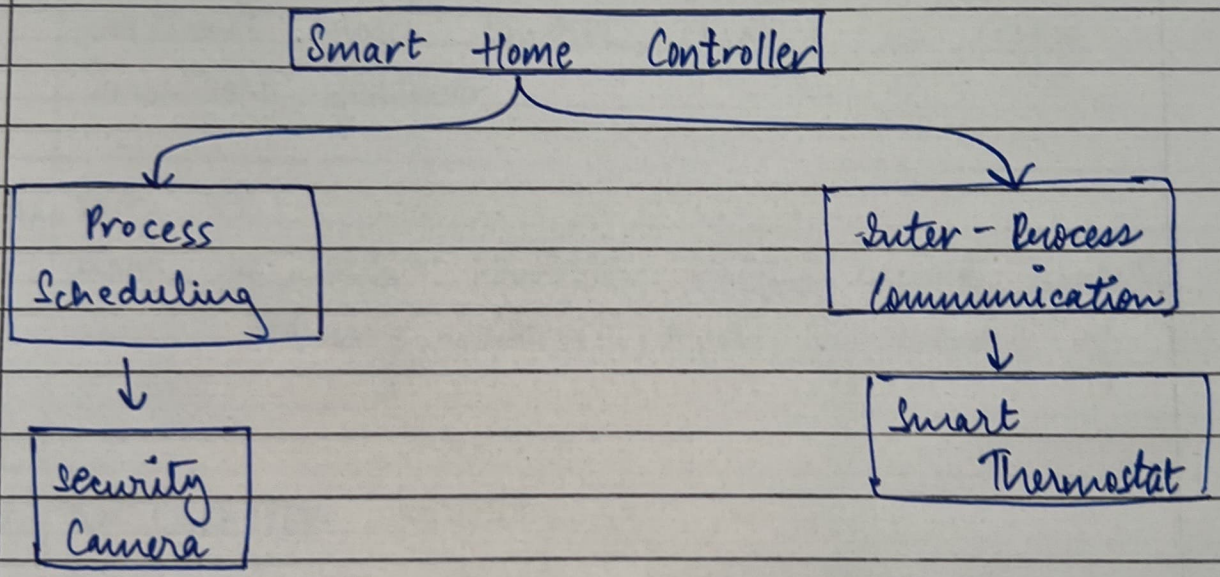
Teacher's Signature.....

Topic ..... Date ...../...../.....

b) Suitable Scheduling Algorithms:

- priority scheduling → ensures urgent processes like security alerts preempt routine tasks.
- Earliest Deadline First (EDF) → Useful when tasks must complete before specific deadlines.
- Rate-Monotonic Scheduling (RMS): suitable for periodic ~~table~~ tasks, like sensor checks.

Among these, PS or EDF are most effective, as they gaurantee timely execution of safety-crucial events while maintaining system responsiveness.



Teacher's Signature.....