

Name → Purna Kandpal  
 course → BTech CSE - 'D'  
 Roll No. → 2301010251

## Operating System Assignment 01

### Part A

ques1

Despite \_\_\_\_\_ systems?

Ans 1. Modern systems still rely heavily on operating systems because :-

- Resource management → The OS efficiently manages hardware resources (CPU, memory, storage, I/O devices) and allocates them to different programs.
- User & Application Interface → The OS provides a convenient interface b/w hardware and users/applications, enabling portability, multitasking and security.

ques2. You \_\_\_\_\_ why?

Ans 2. Real-Time Operating System (RTOS)

RTOS ensures timely, predictable, and reliable response to inputs like heart rate signals process data with low latency, provides efficient resource management on small, low-power hardware - critical for health monitoring devices.

ques3. Given \_\_\_\_\_ Why?

Ans 3. 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.

TOPIC ..... DATE .....

usy

nsy

A developer reasoning.

Refute the claim, because OS structure directly impacts performance, reliability, scalability and security.

For ex:- microkernel isolates services for fault tolerance, while a layered structure improves maintainability. Just "running processes" isn't enough if the system is low, insecure or unstable.

Ques

i) Explain states.

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

ii) If a involve?

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

iii) The and why?

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

B

b) 6

content - performance

(a) Total content switching time,

save state = 2ms

load state = 3ms

Scheduler overhead = 1ms

$$\text{Total time} = 2 + 3 + 1 = 6 \text{ ms}$$

b) Explain - performance.

- content switching is pure overhead (no useful work is done during this time)
- Higher switching time reduces CPU efficiency, as more time is spent switching than executing processes.
- In multitasking, frequent content switches with high overhead can slow down throughput and increase response time.

Ques 7.

Given :- execution time (single-threaded) = 40 sec

Multithreading is used with n threads per process

Execution time estimate :

In ideal conditions (perfect parallelism, no overhead) :

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

Example :

$$\text{if } n = 2 \rightarrow 20 \text{ sec}$$

$$\text{if } n = 4 \rightarrow 10 \text{ sec}$$

$$\text{if } n = 8 \rightarrow 5 \text{ sec}$$

## How multithreading 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.
- Threads share resources, making execution faster and more efficient.

Ques 8

Given times

Process : P<sub>1</sub>    P<sub>2</sub>    P<sub>3</sub>    P<sub>4</sub>

Burst time : 5    3    8    6

(a)

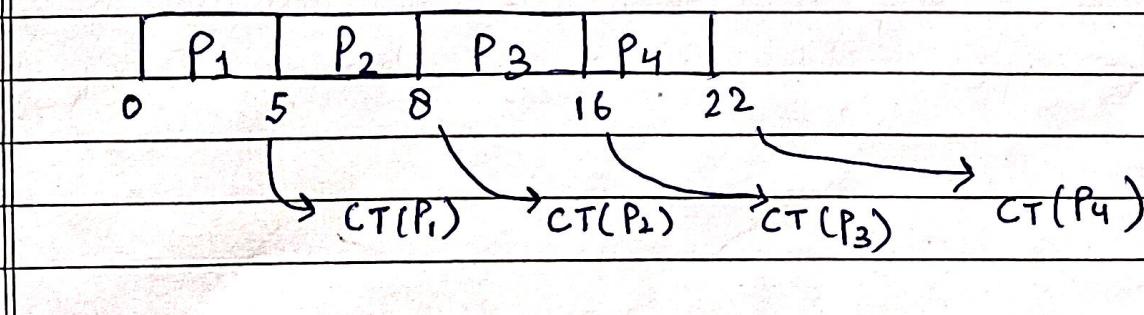
FCFS

Process	Arrival time (AT)	Burst Time (BT)	Completion Time (CT)	Waiting Time (WT)	TAT
P <sub>1</sub>	0	5	5	$5-5=0$	$5-0=5$
P <sub>2</sub>	0	3	8	$8-3=5$	$8-0=8$
P <sub>3</sub>	0	8	16	$16-8=8$	$16-0=16$
P <sub>4</sub>	0	6	22	$22-6=16$	$22-0=22$

$$WT = Turnaround - Burst \quad (TAT - BT)$$

$$TAT = completion - arrival \quad (CT - AT)$$

Gantt chart,



TOPIC ..... DATE .....

$$\text{Avg waiting time} = (0+5+8+16)/4 = 7.25 \text{ ms}$$

$$\text{Avg turnaround time} = (5+8+16+22)/4 = 12.75 \text{ ms}$$

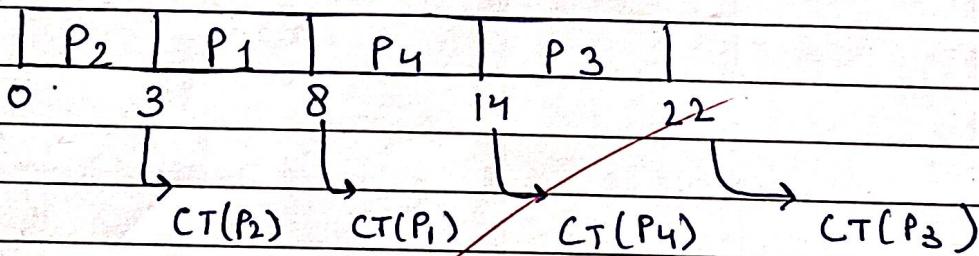
(b) Non-preemptive SJF

Process	AT	BT	CT	WT	TAT
P <sub>1</sub>	0	5	8	8-5 = 3	8 = 8-0
P <sub>2</sub>	0	3	3	3-3 = 0	3 = 3-0
P <sub>3</sub>	0	8	22	22-8 = 14	22 = 22-0
P <sub>4</sub>	0	6	14	14-6 = 8	14 = 14-0

$$\text{TAT} = CT - AT$$

$$WT = TAT - BT$$

Gantt chart,



$$\text{Avg waiting time} = (3+0+14+8)/4 = 6.25 \text{ ms}$$

$$\text{Avg turnaround time} = (8+3+22+14)/4 = 11.75 \text{ ms}$$

TOPIC ..... DATE .....

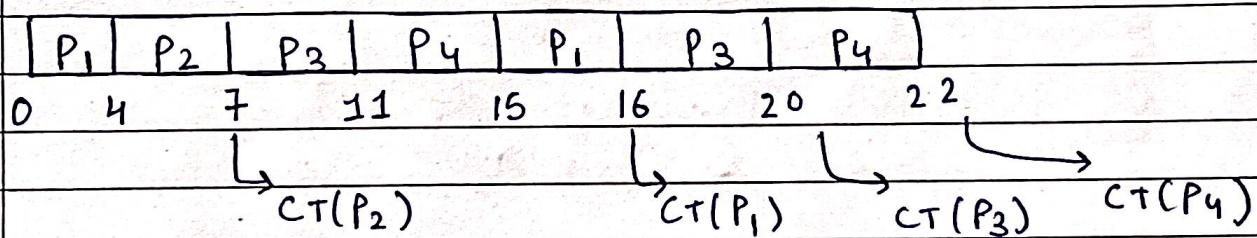
(c) Round Robin (quantum = 4ms)

Process	AT	BT	CT	WT	TAT
P <sub>1</sub>	0	5	16	16 - 5 = 11	16 - 0 = 16
P <sub>2</sub>	0	3	7	7 - 3 = 4	7 - 0 = 7
P <sub>3</sub>	0	8	20	20 - 8 = 12	20 - 0 = 20
P <sub>4</sub>	0	6	22	22 - 6 = 16	22 - 0 = 22

$$WT = TAT - BT$$

$$TAT = CT - AT$$

Gantt chart,



$$\text{Avg waiting time} = (11 + 4 + 12 + 16) / 4 = 10.75 \text{ ms}$$

$$\text{Avg turnaround} = (16 + 7 + 20 + 22) / 4 = 16.25 \text{ ms}$$

\* Non-preemptive SJF is best because it gives the lowest average 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 or RR.

## (a) Smart Home System (TOT devices)

- OS role with scheduling + IPC
- Process scheduling → The OS assigns higher priority to critical tasks (eg. intrusion detection) so they preempt less urgent ones (like lighting)
- Inter-Process communication (IPC) → Enables devices and controller processes to exchange data quickly (eg: camera sends motion alert → controller process reacts immediately)

(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 tasks, like sensor checks.

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

### Smart Home Controller

Process  
Scheduling

Inter-Process  
communication

Security  
Camera

Smart  
Thermostat

Got it!  
G1/G2