Activity Slow Learner

Scenario: 1

Imagine a computer system running a multi-programmed operating system. Three processes – **P1**, **P2**, and **P3** – are loaded into memory.

P1 has just been created and is ready to run.

P2 is currently using the CPU.

P3 has requested input from the keyboard and is waiting for it.

After a few seconds:

- **P2** finishes its time slice and is moved back to the ready queue.
- The OS selects **P1** to run on the CPU.
- Meanwhile, the keyboard input requested by **P3** is received, and it moves to the ready queue.

Case Study Questions:

Q1: What is the current state of P1, P2, and P3 at the beginning of the scenario?

- **P1**: Ready state (has just been created and is ready to run)
- **P2**: Running state (currently using the CPU)
- **P3**: Waiting state (waiting for keyboard input)

Q2: What causes P2 to leave the CPU and move to the Ready state?

- **Answer**: P2 finishes its time slice (time quantum), and due to time-sharing, it is **preempted** by the OS and moved back to the **Ready** queue.

Q3: Why is P3 in the Waiting state initially?

- **Answer**: P3 is in the **Waiting** state because it has **requested input from the keyboard**, which is an I/O operation. Until the input is received, the process cannot continue execution.

Q4: After the keyboard input is received, what is the new state of P3?

- **Answer**: Once the keyboard input is received, P3 transitions from the **Waiting** state to the **Ready** state, meaning it's now prepared to be scheduled for the CPU.

Q5: Describe the state transitions for P1 in this scenario.

- **Initial State**: P1 is created → enters the **Ready** state

- **Transition**: After P2's time slice ends, the OS selects P1 → moves to the **Running** state
- (Not mentioned in the scenario but implied): After its execution or time slice, P1 would either finish (terminate) or go back to **Ready** if preempted or to **Waiting** if it requests I/O.

Q1. Which process state indicates the process is waiting for an event such as I/O completion?
A. Running
B. Ready
C. **Waiting**
D. Terminated

Q2. What does the OS do when a process finishes its time slice?
A. Terminates the process
B. Moves it to the waiting queue
C**. Moves it to the ready queue**
D. Restarts the process

Q3. What triggers a process to move from the Waiting state to the Ready state?
A**. Completion of I/O or the awaited event**
B. Expiration of time slice
C. Arrival of new process
D. Process crash

Q4. In a preemptive multitasking system, how is the next process selected to run?
A. Random selection
B. User request
C**. Scheduler algorithm**
D. Manual execution

Q5. What is the final state of a process after it completes its execution?
A. Waiting
B. Ready
C**. Terminated**
D. Running

Answer Key:

Q1: **C**; Q2: **C** Q3: **A** Q4: **C** Q5: **C**

Story-Based Scenario

A toy factory has **one machine** that makes toys. Four toy orders (like processes) arrive at different times and take different amounts of time to finish.

Order (Process)	Arrival Time (in mins)	Time to Finish (Burst Time)
O1	0	5
O2	1	3
O3	2	8
O4	3	6

The manager must decide **which order to process first**. They can use different **scheduling rules**.

1. Using FCFS, what is the order of processing?

Answer: O1 → O2 → O3 → O4

- Based on arrival times: O1 (0) → O2 (1) → O3 (2) → O4 (3)

2. Which order will finish first in SJF?

Answer: O2

- SJF = Shortest Job First
- O2 has the shortest burst time (3 mins), so it finishes first once it arrives.

3. In Round Robin (Time Slice = 2 mins), which order gets the CPU again after the first round?

Answer: O1

- Round 1: O1 (2 mins), O2 (2 mins), O3 (2 mins), O4 (2 mins)
- In Round Robin, we go in a circular manner, so **O1 gets the CPU again** at the beginning of the **second round**.

4. What happens if one order always has a higher priority than the others?

Answer: The others may **never get processed** (this is called **starvation**).

- If one order (e.g., O1) has the **highest priority** always, other processes like O2, O3, and O4 may wait forever.

5. Which algorithm is fairest to all orders?

Answer: **Round Robin**

- Because it gives **equal time slices** to each order in a cycle.

- Ensures all processes get CPU time fairly.

MCQs

What does FCFS stand for?
A. Fastest CPU First
B. First Come First Serve
C. Finish CPU Fast
D. First CPU Service

Q2. In SJF, which order is chosen?
A. The biggest
B**. The one with the shortest time**
C. The one with highest number
D. The last one

Q3. Round Robin gives each process:
A. No time
B. One chance only
C. **Equal time slices**
D. Random time

Q4. In a factory with only one machine (CPU), how many orders can be "running" at one time?
A. 2
B. 4
C. 1
D. All

Q5. What problem happens if we always choose highest priority only?
A. Everyone is happy
B. Some orders may starve
C. It runs faster
D. The machine breaks

Activity Moderate Learner (Case study)

A modern multitasking operating system is managing four processes: **P1**, **P2**, **P3**, and **P4**. The system uses a **preemptive priority-based CPU scheduling algorithm**, where a **lower number indicates higher priority**.

- **P1 (Priority: 2)** is in the **Ready** queue. **P2 (Priority: 1)** is **Running** on the CPU. **P3 (Priority: 3)** is in the **Waiting** state for disk I/O. **P4 (Priority: 1)** is **New** and just admitted into memory.

The following events occur:

- **P2** initiates a file read operation and enters the **Waiting** state.

- The scheduler now chooses the **highest priority** process from the Ready queue.
- **P4** finishes loading and enters the **Ready** queue.
- The disk I/O for **P3** completes, and it is moved to the Ready queue.
- After some time, **P4** completes execution and terminates.

1 After P2 initiates the file read, which process is selected next by the scheduler? Why?

Answer: **P1** is selected next.

- When P2 performs a file read, it **enters the Waiting state** (for I/O).
- Now the scheduler checks the **Ready queue**.
- Among the Ready processes, we have only **P1 (Priority 2)** available at this moment.
- Since **P4 is still in the New state**, P1 is the **highest priority (lowest number)** among Ready processes.
  So, **P1 gets selected to run**.

2 What happens when P4 moves from New to Ready? How does it affect scheduling?

Answer: P4 has **Priority 1**, which is **higher than P1 (Priority 2)**.

- As soon as P4 enters the **Ready queue**, it is **compared** with the currently running process.
- If **P1 is running** at this time, it is **preempted** by P4.
  So, **P4 will replace P1** on the CPU immediately due to **higher priority**.

3 When P3 completes I/O and returns to the Ready state, is it selected to run immediately? Justify your answer using priority levels.

Answer: **No, P3 is not selected immediately.**

- P3 has **Priority 3**, which is **lower** than both:
  - P1 (Priority 2)
  - P4 (Priority 1, possibly still running)
- Since **priority 1 and 2** are both higher, P3 must **wait in the Ready queue** until all **higher-priority processes complete**.

  P3 runs **only when no higher-priority processes** are in the Ready state or Running.

4 What is the complete life cycle path (states) that P2 follows in this scenario?

Answer: P2 goes through the following states:

- **Running** (Initially using CPU)
- **Waiting** (After initiating a file read operation)
- **Ready** (After I/O completes — not mentioned but implied)
- **Running** (if rescheduled after higher-priority processes terminate)
- **Terminated** (eventually, though not described in this scenario)

Full cycle: **Running → Waiting → Ready → (Running) → Terminated**

5 Why does P4 terminate without entering the Waiting state? What does this imply about its execution?

Answer: P4 never entered the Waiting state, which implies:

- It **did not perform any I/O operations** (disk, keyboard, file, etc.).
- It was likely a **CPU-bound process** — it only used the processor and completed without blocking.

This means P4's execution was **straightforward**, using **only the CPU** until it completed.

MCQs Asked in Placement Drives

Q1. In a preemptive priority-based scheduling system, which factor *must* the OS consider when selecting the next process?
A. I/O burst time
B. Process priority
C. Process age
D. Parent-child relationship

Q2. What causes a context switch between two processes in the scenario above?
A. Process termination
B. A higher-priority process entering the Ready queue
C. Manual user interruption
D. Process entering the Waiting state

Q3. Which of the following is true about a process in the New state?
A. It can request I/O
B. It can be scheduled for CPU
C. It has not yet been admitted to the Ready queue
D. It is waiting for memory deallocation

Q4. Why does a process enter the Terminated state?
A. After being idle too long
B. Due to preemption
C. After successful or abnormal completion
D. If it is moved to the Ready queue

Q5. Which of the following best describes the difference between Ready and Waiting states?
A. Ready waits for I/O; Waiting waits for CPU
B. Ready is scheduled for CPU; Waiting waits for event completion
C. Waiting is selected by scheduler; Ready is not
D. There is no difference

Story-Based Scenario

You are a junior system administrator at a mid-sized company. The system handles four user requests (processes), and your task is to analyze how different CPU scheduling algorithms affect performance.

Process	Arrival Time (ms)	Burst Time (ms)	Priority (1 = High)
P1	0	5	3
P2	1	3	1
P3	2	8	2
P4	3	6	4

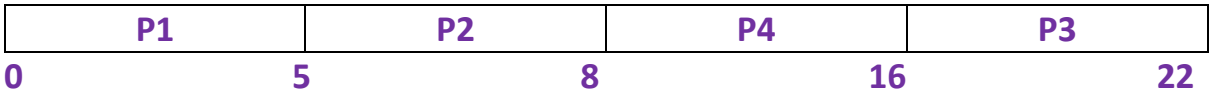
Use a Gantt chart format to visualize process execution. FCFS (First Come First Serve), SJF (Shortest Job First – Non-Preemptive), Priority Scheduling (Non-Preemptive)

FCFS

P1	P2	P3	P4
0	5	8	16	22

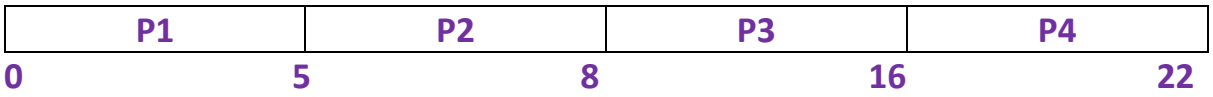
Process	Completion Time	Turnaround Time	Waiting Time
P1	5	5	0
P2	8	7	4
P3	16	14	6
P4	22	19	13

SJF



Process	Completion Time	Turnaround Time	Waiting Time
P1	5	5	0
P2	8	7	4
P3	22	20	12
P4	16	13	7

Priority Scheduling (Non-Preemptive)



Process	Completion Time	Turnaround Time	Waiting Time
P1	5	5	0
P2	8	7	4
P3	16	14	6
P4	22	19	13

Q1. In FCFS, which process is scheduled first?

A. The one with highest priority

B. The one with shortest burst time

C. The one that arrives first

D. Chosen randomly

Q2. What is the main advantage of SJF?

A. Reduces turnaround time

B. Always fair

C. Ignores burst time

D. Supports parallel processing

Q3. In Priority Scheduling, a process with priority = 1:
A. Will be scheduled last
B. Has the highest priority
C. Has lowest priority
D. Skipped by CPU

Q4. Which scheduling algorithm may cause starvation?
A. FCFS
B. Round Robin
C. SJF
D. Priority Scheduling

Q5. Average waiting time is:
A. Always zero
B. The same in all algorithms
C. Total waiting time ÷ number of processes
D. Completion time - Burst time

Fast Learner Activity

A university's operating system lab team has developed a custom OS kernel. During testing, they observe an unusual behavior: two processes, **P1** and **P2**, both marked as "Running" simultaneously on a **single-core system**.

Upon investigation, the team inspects the **Process Control Blocks (PCBs)** of all active processes. Below are the **PCB fields** for P1, P2, and P3:

Field	**P1**	**P2**	**P3**
Process ID	101	102	103
Process State	Running	Running	Waiting (I/O)
Program Counter	0x004A	0x1F3C	0x03B2
CPU Registers	Saved	Saved	Saved
Scheduling Info	Priority 2	Priority 1	Priority 3
Memory Limits	0x0000–0x1FFF	0x2000–0x3FFF	0x4000–0x5FFF
I/O Status	No pending I/O	No pending I/O	Disk read pending
Accounting Info	8 ms CPU time	5 ms CPU time	3 ms CPU time

1. Identify the **error in PCB management**. What state should P1's PCB have been updated to when P2 was scheduled?

Answer:

- Error: Both P1 and P2 are marked as "Running" on a single-core system, which is logically impossible — only one process can run at a time.
- Correction:
  When P2 was scheduled, P1's state in its PCB should have been updated to "Ready" or "Waiting" (depending on its activity), not "Running".

2. Explain the **role of the PCB's Program Counter and CPU Registers** during context switching.

Answer:

- Program Counter (PC):
  Holds the address of the next instruction to execute. It allows the OS to resume execution of the process later.
- CPU Registers:
  Store the current state of the process, such as variable values, instruction flags, etc.
- During context switching:
  - The current process's PC and registers are saved to its PCB.
  - The new process's PC and registers are loaded from its PCB to the CPU.

3. How could such an error lead to system instability or incorrect resource management?

Answer:

- Multiple processes thinking they "own" the CPU may lead to:
  - Race conditions
  - Data corruption
  - Inconsistent memory access
- The scheduler may duplicate CPU time, miss context switches, or incorrectly assign I/O or memory to multiple processes simultaneously.
- Can also cause:
  - System crashes
  - Unpredictable behavior

- Security vulnerabilities

4. Propose **two mechanisms** that could prevent or detect PCB synchronization issues.

Answer:

(a) Mutual Exclusion for Scheduler Access:

- Use locks or semaphores to ensure only one process can be updated/scheduled at a time by the scheduler.

(b) State Validation Checks:

- Before scheduling, the OS should validate process states to ensure only one process is marked "Running".
- Implement assertions or audits during development/testing.

5. Suppose P3 completes its I/O. Describe the full sequence of changes in its PCB if it becomes the next scheduled process.

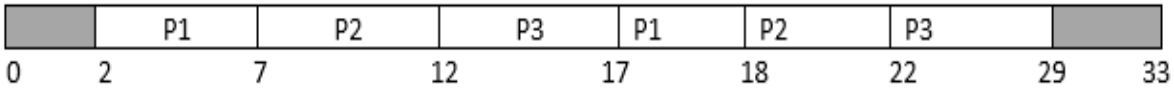
Answer:

Sequence of PCB changes for P3:

1. I/O Completion:
   - PCB state changes from "Waiting" → "Ready".
   - I/O Status is updated to "No pending I/O".
2. Scheduled by OS:
   - OS selects P3 (e.g., based on priority or time slice).
   - PCB state changes from "Ready" → "Running".
3. Context Switch Occurs:
   - Program Counter and CPU Registers for the previous process (e.g., P2) are saved.
   - P3's PC and registers are loaded into the CPU.
4. Accounting Info Updated:
   - Start tracking CPU time usage for P3 (e.g., increases from 3 ms onward).

Three process p1, P2 and P3 arrive at time zero. Their total execution time is 10ms, 15ms, and 20ms respectively. They spent first 20% of their execution time in doing I/O, next 60% in CPU processing and the last 20% again doing I/O. For what percentage of time was the CPU free? Use Round robin algorithm with time quantum 5ms.

Process	Execution time	I/O burst	CPU burst	I/O Burst
P1	10	2	6	2
P2	15	3	9	3
P3	20	4	12	4



✅ **CPU Free Time %**

$$\text{CPU Free Time } \% = \left(\frac{6}{30}\right) \times 100 = 20\%$$

✅ **Final Answer:**

🟩 The CPU was free for approximately 20% of the total execution time.