

**Fall**

**2025**

Department of Computer Science & Engineering

**Course Title: Operating Systems Lab**

**Course Code:** **CSE 406**

**Report No:** **03**

**Date of Submission: 28th February, 2025**

**Submitted to:**

**Atia Rahman Orthi**

**Lecturer**

Department of Computer Science & Engineering

Submitted by:

**Jannatul Fardaous Jim**

Reg No: 21201071

Roll: 071

Sec: B1

E-mail: 21201071@uap-bd.edu

**Round Robin Scheduling Algorithm: Implementation and Analysis:**

**Theory:Round Robin (RR)** is one of the simplest and most widely used CPU scheduling algorithms in time-sharing systems. It assigns a fixed time quantum to each process in a cyclic order, ensuring that no single process monopolizes the CPU. If a process does not complete within its allocated quantum, it is placed at the end of the queue, and the CPU is assigned to the next process in line. This approach ensures fairness and reduces starvation issues.

**Round Robin scheduling** executes processes in a cyclic order, similar to a queue at a ticket counter. During the lab, we:

* **Understood how Round Robin scheduling works.**
* **Implemented the algorithm step by step.**
* **Calculated key metrics for the given processes:**

1. **Completion Time (CT):** The time when a process finishes execution.
2. **Turnaround Time (TAT):** The total time a process spent in the system.
3. **Waiting Time (WT):** The time a process spent waiting before execution.
4. **Response Time (RT):** The time from the process’s arrival to its first execution.

## **Problem Statement:**

A system has four processes that arrive at different times and require a certain amount of CPU execution time. The processes are scheduled using the **Round Robin scheduling algorithm** with a fixed time quantum. Given the following information:

|  |  |  |
| --- | --- | --- |
| **Process** | **Arrival Time (AT)** | **Burst Time (BT)** |
| **P1** | **0** | **5** |
| **P2** | **1** | **4** |
| **P3** | **2** | **2** |
| **P4** | **4** | **1** |

## **Determine the Completion Time (CT) for each process.**

1. Calculate the Turnaround Time (TAT) for each process using the formula: ****TAT = CT - AT****

## **Calculate the Waiting Time (WT) for each process using the formula:**

## ****WT = TAT - BT****

1. Calculate the Response Time (RT) for each process.

## **Calculations:**

Step-by-step execution of the **Round Robin** algorithm: time Quantum = 2 ms.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Process** | **Arrival Time** | **Burst Time** | **Completion Time**  **(CT)=AT + BT** | **Turnaround Time**  **(TAT) =CT-AT** | **Waiting Time (WT)=TAT-BT** | **Response Time(RT)** |
| **P1** | **0** | **5** | **12** | **7** | **12** | **0** |
| **P2** | **1** | **4** | **11** | **6** | **10** | **1** |
| **P3** | **2** | **2** | **6** | **2** | **4** | **2** |
| **P4** | **4** | **1** | **9** | **4** | **5** | **4** |

**Logic Behind Solving the Round Robin Scheduling Problem:**

1. The processes are sorted based on their arrival times to ensure correct scheduling.
2. A queue is used to maintain the execution order, and processes are scheduled cyclically based on the time quantum.
3. Each process executes for either the full quantum or its remaining burst time (whichever is smaller).
4. If a process does not finish, it is re-added to the queue to be executed again after other processes.
5. Completion Time is recorded when a process finishes execution, from which Turnaround Time and Waiting Time are derived.
6. Response Time is determined when the process first gets CPU execution.

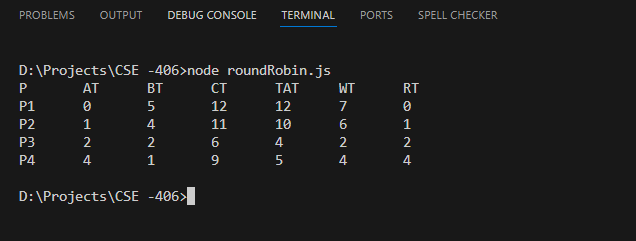
**Round Robin Scheduling Algorithm - JavaScript Code Implementation Git-hub Link**

**Git-hub Link: <https://github.com/JannatJimu/CSE-406/blob/main/roundRobin.js>**





**Round Robin Scheduling Algorithm - Program Execution Output:**



## **Conclusion:**

1. Ensures fair CPU allocation, ideal for time-sharing systems.
2. Executes processes cyclically and computes key metrics.
3. Time quantum choice is crucial:

* Too large → Behaves like FCFS.
* Too small → Increases context switching.

1. Optimal quantum balances efficiency and fairness.