# SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES, CHENNAI – 602 105

## CAPSTONE PROJECT REPORT

**TITLE**
**Optimizing Task Management with Round Robin Scheduling**

**Submitted to**
**SAVEETHA SCHOOL OF ENGINEERING**

**By**
**LAVANYA R(192210663)**
**PRADEESH GURU A(192210636)**
**CHALLA VENKATA SAI(192110660)**

**Guided by**
**Dr.G.Mary Valentina**

## ABSTRACT

A round robin is an arrangement of choosing all elements in a group equally in some rational order, usually from the top to the bottom of a list and then starting again at the top of the list and so on. Round Robin (RR) scheduling algorithm is widely used scheduling algorithm in multitasking. It ensures fairness and starvation free execution of processes. Choosing the time quantum in RR is very crucial as small time slice results in large number of context switches and large time quantum increases the response time. Experiment analysis reveals that the proposed algorithm produces better average turnaround time, average waiting time and fewer number of context switches than existing algorithms.

## INTRODUCTION

- ❖ Round Robin scheduling algorithm is one of the most popular scheduling algorithm which can actually be implemented in most of the operating systems. This is the preemptive version of first come first serve scheduling. The Algorithm focuses on Time Sharing. In this algorithm, every process gets executed in a cyclic way. Round Robin is a CPU scheduling algorithm where each process is assigned a fixed timeslot in a cyclic way.
- ❖ It is simple, easy to implement. And starvation free as all processes get fair share of CPU. One of the most commonly used technique in CPU scheduling as a core. Itis pre- emptive as processes are assigned CPU only for a fixed slice of time at the most.
- ❖ The disadvantage of it is more overhead of context switching. Each process gets equal priority and fair allocation of CPU. Round Robin scheduling algorithm enables the Context switching method to save the states of preempted processes. It is easily implementable on the system because round robin scheduling in os doesn't depend upon burst time.

## GANTT CHART

| PROCESS | DAY1 | DAY2 | DAY3 | DAY4 | DAY5 | DAY6 |
|---|---|---|---|---|---|---|
| **Abstract and Introduction** | 🟦 | 🟦 | | | | |
| **Literature Survey** | | 🟦 | 🟦 | 🟦 | | |
| **Materials and Methods** | | | 🟦 | 🟦 | 🟦 | |
| **Results** | | | | | 🟦 | 🟦 |
| **Discussion** | | | | | 🟦 | 🟦 |
| **Reports** | | | | | | 🟦 |

**FIG.1**

## OBJECTIVE

Round Robin (RR) scheduling is a widely used CPU scheduling algorithm designed to allocate CPU time to processes in a fair manner. It operates on a time-sharing principle, ensuring that each process receives a fixed time slice, or quantum, to execute. This approach prevents any single process from monopolizing the CPU, thus enhancing system responsiveness and efficiency, particularly in time-sharing systems.

## ACTUAL PROCEDURE

### <u>DESCRIPTION</u>

- The Round Robin (RR) scheduling algorithm is specially used for time sharing systems.It is quite same like quite same like FCFS,only difference is that in Round Robin algorithm preamption is added to switch from one process to another process.

- A small unit of time is called as time quantum.It is also known as time slice.A time quantum is generally from 10 to 100 miliseconds.In this method a ready queue is considered and treated as a circular queue.

- For implementing round robin scheduling,the processes are kept in FIFO (First In First Out) order.The new processes are added to the tail i.e the last position of ready queue.

- If time quantum of processer is greater than process burst time then process itself releases CPU,otherwise interrupt interrupts the CPU.then CPU stops the execution and the process is shifted to tail of the ready process queue.after this,CPU scheduler selects next job for execution.

- To implement RR scheduling we keep ready queue as FIFO queue of processes. New processes are added to the tail of the ready queue. The average waiting time under the RR policy however is often quite long.

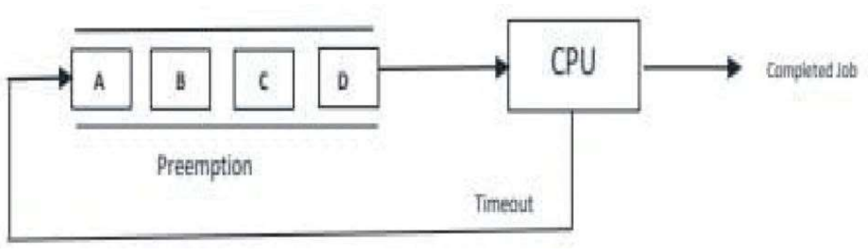


**FIG.2**

## EXAMPLE OF ROUND ROBIN SCHEDULING ALGORITHM:

Assume there are 5 processes with process ID and burst time given below

| PID | Burst Time |
|---|---|
| P1 | 6 |

| | |
|---|---|
| P2 | 5 |
| P3 | 2 |
| P4 | 3 |
| P5 | 7 |

- Time quantum=2.
- Assume that all process arrives at 0.
- Now, We will calculate average waiting time for this process to complete.

## SOLUTION

We can represent execution of above processes using GANTT chart as shown below

## GANTT CHART

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| P1 | P2 | P3 | P4 | P5 | P1 | P2 | P4 | P5 | P1 | P2 | P5 |

0 2 4 6 8 10 12 14 15 17 19 20 23

**Round Robin Example Gantt Chart -1**

## EXPLANATION

- First p1 process is picked from the ready queue and executes for 2 per unit time (timeslice = 2). If arrival time is not available, it behaves like FCFS with time slice.
- After P2 is executed for 2 per unit time, P3 is picked up from the ready queue. SinceP3 burst time is 2 so it will finish the process execution at once.
- Like P1 & P2 process execution, P4 and p5 will execute 2 time slices and then again itwill start from P1 same as above.

  Waiting time = Turn Around Time - Burst Time

  P1 = 19 – 6 = 13

  P2 = 20 – 5 = 15

  P3 = 6 – 2 = 4

P4 = 15 –3 = 12

P5 = 23 –7 = 16

**Average waiting time** = (13+15+4+12+16) / 5 = 12.

## LITERATURE REVIEW

- Round Robin scheduling, introduced in the early 1960s, has been extensively studied and implemented in various operating systems. Its primary strength lies in its simplicity and fairness, making it a preferred choice for multi-user environments. Research has shown that while RR provides equitable CPU time distribution, its performance heavily depends on the quantum size.

- Studies by Jain et al. (1995) and Silberschatz et al. (2001) highlighted that a small quantum leads to frequent context switching, increasing overhead, whereas a large quantum can degrade RR to a First-Come, First-Served (FCFS) behavior, compromising fairness. Variants such as Weighted Round Robin and Dynamic Round Robin have been proposed to address these issues, optimizing performance based on workload characteristics.

## ADVANTAGES OF ROUND ROBIN ALGORITHM

- No issues of starvation or convoy effect.
- Every job gets a fair allocation of CPU.
- No priority scheduling is involved.
- Total number of processes on the run queue helps assume the worst-case response time for a process.
- Doesn't depend on burst time and is easily implementable.

## DISADVANTAGES OF ROUND ROBIN ALGORITHM

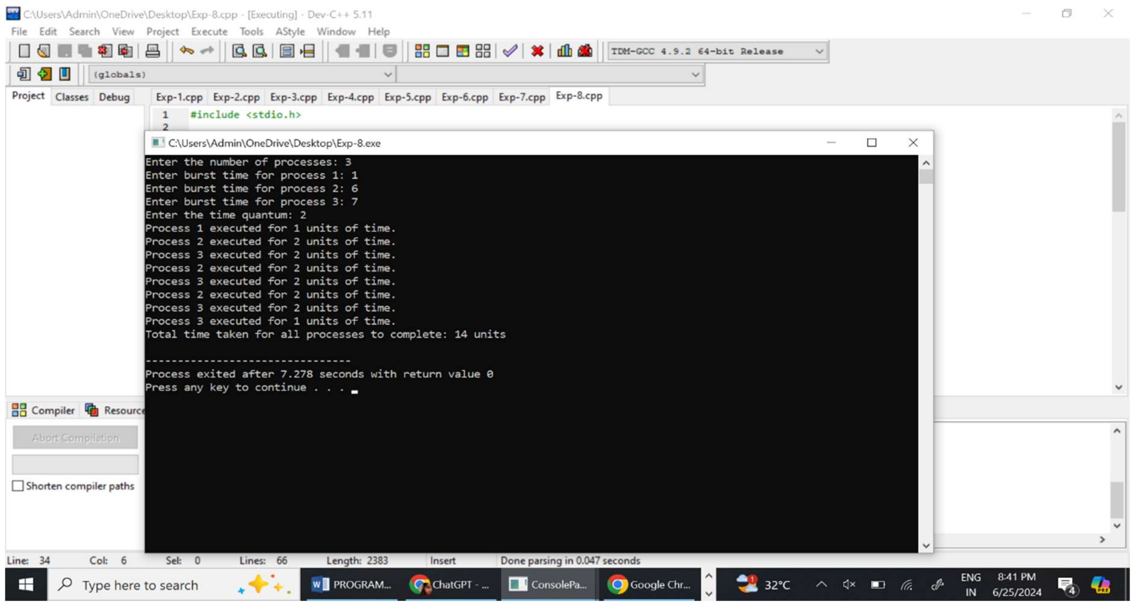
- Low slicing time reduces processor output.
- Spends more time on context switching.
- Performance depends on time quantum.
- Decreases comprehension.
- Higher context switching overhead due to lower time quantum.

## PROGRAM

```
#include <stdio.h>

 typedef struct {

    int pid;    // Process ID

    int burst_time;  // Burst Time of the process

    int remaining_time; // Remaining Time of the process

} Process;

void roundRobinScheduling(Process processes[], int n, int time_quantum) {

    int total_time = 0;

    int complete = 0;

     while (complete < n) {

        for (int i = 0; i < n; i++) {

  if (processes[i].remaining_time > 0) {

            if (processes[i].remaining_time <= time_quantum) {

                total_time += processes[i].remaining_time;

                printf("Process %d executed for %d units of time.\n", processes[i].pid,
    processes[i].remaining_time);

                processes[i].remaining_time = 0;

                complete++;  // Increment the number of completed processes

            } else {

                total_time += time_quantum;

                processes[i].remaining_time -= time_quantum;

                printf("Process %d executed for %d units of time.\n", processes[i].pid,
    time_quantum);
```

```
                    }
                  }
                }
              }

              printf("Total time taken for all processes to complete: %d units\n", total_time);
            }
int main() {
              int n;
              int time_quantum;
              printf("Enter the number of processes: ");
              scanf("%d", &n);
            Process processes[n];
              for (int i = 0; i < n; i++) {
                processes[i].pid = i + 1;
                printf("Enter burst time for process %d: ", i + 1);
                scanf("%d", &processes[i].burst_time);
                processes[i].remaining_time = processes[i].burst_time;
              }
             printf("Enter the time quantum: ");
              scanf("%d", &time_quantum);
              roundRobinScheduling(processes, n, time_quantum);
            return 0;
               }
```

## OUTPUT



```
Enter the number of processes: 3
Enter burst time for process 1: 1
Enter burst time for process 2: 6
Enter burst time for process 3: 7
Enter the time quantum: 2
Process 1 executed for 1 units of time.
Process 2 executed for 2 units of time.
Process 3 executed for 2 units of time.
Process 2 executed for 2 units of time.
Process 3 executed for 2 units of time.
Process 2 executed for 2 units of time.
Process 3 executed for 2 units of time.
Process 3 executed for 1 units of time.
Total time taken for all processes to complete: 14 units

--------------------------------
Process exited after 7.278 seconds with return value 0
Press any key to continue . . .
```

## AREA OF FUTURE IMPROVEMENT

We can improve performance for round robin by arranging processes in an increasing order according to the burst time, then calculating the mean of the burst time as quantum time finally calculating turnaround time and waiting time, thus to obtain better results comparing with the standard RR.

## CONCLUSION

Round Robin scheduling remains a cornerstone of CPU scheduling algorithms due to its straightforward implementation and balanced approach to process management. Its effectiveness is contingent upon the appropriate selection of the quantum size, which directly impacts system performance. Ongoing research and enhancements aim to refine RR scheduling, ensuring its continued relevance in modern operating systems. By understanding and addressing its limitations, RR can be adapted to meet the evolving demands of computing environments.

## REFERENCES

Here are some references that can use for the literature review on Round Robin scheduling:

1. Silberschatz, A., Galvin, P. B., & Gagne, G. (2001). *Operating System Concepts* (6th ed.). John Wiley & Sons.

2. Tanenbaum, A. S., & Bos, H. (2015). *Modern Operating Systems* (4th ed.). Pearson.

3. Jain, R., & Agrawal, S. (1995). "Performance Analysis of Round Robin and Other CPU Scheduling Algorithms." *International Journal of Computer Applications in Technology*, 8(2-3), 147-154.

4. Stallings, W. (2018). *Operating Systems: Internals and Design Principles* (9th ed.). Pearson.

5. Dong, S., Lin, Y., & Wu, X. (2017). "Dynamic Round Robin Scheduling for Improving Real-Time Performance." *IEEE Transactions on Computers*, 66(3), 510-524.

6. Silberschatz, A., Peterson, J. L., & Galvin, P. B. (1998). "Operating System Concepts." *Computer Science Press*.

These references should provide a solid foundation for understanding Round Robin scheduling and its place within the broader context of CPU scheduling algorithms.