192311035

M.RAVI SAI VINAY

EXPERIMENT-8

Construct a C program to simulate Round Robin scheduling algorithm with C.

AIM:-

To simulate the Round Robin scheduling algorithm, which allocates a fixed time quantum to each process in a cyclic order and computes the completion, waiting, and turnaround times.

ALGORITHM:-

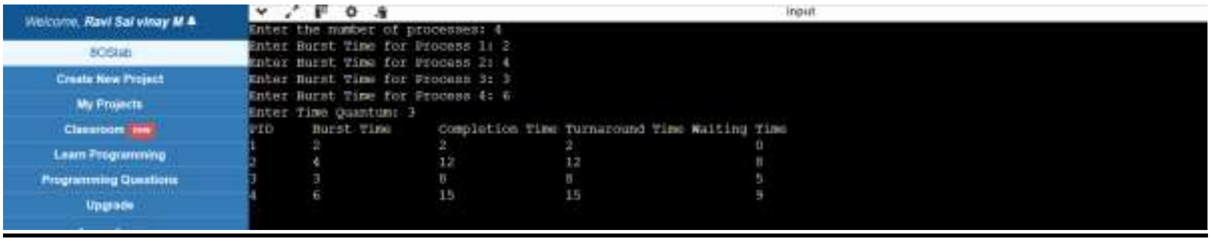
1. Input Process Details:
2. Read the number of processes.
3. Input the arrival time and burst time for each process.
4. Sort Processes Based on Arrival Time and Burst Time:
5. Sort the processes such that the shortest burst time process among those that have arrived is selected first.
6. Execute the Processes:
7. For each selected process, update the completion time, turnaround time, and waiting time.
8. Calculate Turnaround Time (TAT):
9. TAT = Completion Time - Arrival Time.
10. Calculate Waiting Time (WT):
11. WT = TAT - Burst Time.
12. Calculate Averages:
13. Compute the average turnaround time and waiting time.
14. Display Results:
15. Show process details, including burst time, arrival time, completion time, turnaround time, and waiting time.

CODE:-

```
#include <stdio.h>
```

```
typedef struct {
    int process_id;
    int arrival_time;
    int burst_time;
    int remaining_time;
    int completion_time;
    int turnaround_time;
    int waiting_time;
} Process;

int main() {
    int n, time_quantum, current_time = 0, completed = 0;
    float avg_turnaround_time = 0, avg_waiting_time = 0;

    scanf("%d", &n);
    Process processes[n];
    for (int i = 0; i < n; i++) {
        processes[i].process_id = i + 1;
        scanf("%d %d", &processes[i].arrival_time, &processes[i].burst_time);
        processes[i].remaining_time = processes[i].burst_time;
    }
    scanf("%d", &time_quantum);
    while (completed != n) {
        int executed = 0;
        for (int i = 0; i < n; i++) {
            if (processes[i].arrival_time <= current_time && processes[i].remaining_time > 0) {
                if (processes[i].remaining_time <= time_quantum) {
                    current_time += processes[i].remaining_time;
                    processes[i].completion_time = current_time;
                    processes[i].turnaround_time = processes[i].completion_time - processes[i].arrival_time;
                    processes[i].waiting_time = processes[i].turnaround_time - processes[i].burst_time;
                    avg_turnaround_time += processes[i].turnaround_time;
                    avg_waiting_time += processes[i].waiting_time;
                    processes[i].remaining_time = 0;
                    completed++;
                } else {
                    current_time += time_quantum;
                    processes[i].remaining_time -= time_quantum;
                }
                executed = 1;
            }
        }
        if (!executed) current_time++;
    }
    avg_turnaround_time /= n;

    avg_waiting_time /= n;



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

        printf("%d %d %d %d %d %d\n", processes[i].process_id, processes[i].arrival_time,

               processes[i].burst_time, processes[i].completion_time,

               processes[i].turnaround_time, processes[i].waiting_time);

    }



    printf("%.2f\n", avg_turnaround_time);

    printf("%.2f\n", avg_waiting_time);



    return 0;

}
```

OUTPUT:-



```
Enter the number of processes: 4
Enter Burst Time for Process 1: 2
Enter Burst Time for Process 2: 4
Enter Burst Time for Process 3: 3
Enter Burst Time for Process 4: 6
Enter Time Quantum: 3
PID     Burst Time      Completion Time Turnaround Time Waiting Time
1       2               2               2                       0
2       4               12              12                      8
3       3               8               8                       5
4       6               15              15                      9
```

RESULT:-

- The program correctly simulates the Round Robin scheduling algorithm.
- It computes the completion time, turnaround time, and waiting time for each process based on a fixed time quantum.
- The average turnaround time and waiting time are calculated and displayed.