OPERATING SYSTEM
CSE-316
PROJECT REPORT

Topic- Round-Robin Scheduling Algorithm Simulation

Submitted By:	**Submitted To:**
Muhammad Yousuf Khan	Cherry Khosla
K22UP	UID: 13436
R.No:45	

DECLARATION

As the author of this project, I, Muhammad Yousuf Khan, solemnly declare and affirm that all the work submitted is original and entirely my own. I have not engaged in any form of cheating or plagiarism whatsoever. All sources used in this project have been rightfully cited and acknowledged without any direct copying of information from any source. I am fully aware that academic integrity is a vital aspect of scholarly work, and as such, any violation will be met with severe consequences. Therefore, I pledge to uphold academic honesty at all times throughout my academic journey.

Name of Student : Muhammad Yousuf Khan
Registration Number: 12204116

Certificate of Completion

This is to certify that the declaration statement made by Muhammad Yousuf Khan is correct to the best of my knowledge and belief. He has completed this Project under my guidance and supervision. The present work is the result of their original investigation, effort and study. No part of the work has ever been submitted for any other degree at any University. The Project is fit for the submission and partial fulfillment of the conditions for the award of B.Tech degree in Computer Science and Engineering (AI & ML) from Lovely Professional University, Phagwara.

Signature Of Instructor

ACKNOWLEDGEMENT

I would like to extend my sincere thanks to the Lovely School of Computer Science and Engineering for providing me with the opportunity to fulfill my wish and achieve my goal. I am grateful to Cherry Khosla mam for providing me with the opportunity to undertake this project and for providing me with all the necessary facilities. I am highly thankful to sir for her active support, valuable time and advice, whole-hearted guidance, sincere cooperation, and pain-taking involvement during the study and in completing the assignment of preparing the said project within the time stipulated. Lastly, I am thankful to all those, particularly the various friends, who have been instrumental in creating a proper, healthy, and conducive environment and including new and fresh innovative ideas for me during the project.

____

Signature Of Instructor

Question

To create a simulation program for testing the Round Robin scheduling algorithm, we will generate a set of processes with random arrival times and CPU burst times, and run the algorithm for a specific time duration. The program will record the average waiting time and turnaround time for each process. Our aim is to compare the results of this simulation with the ideal scenario of a perfect scheduler. We will output the results in HTML format for easy readability and understanding.
To further improve the accuracy of our simulation program, we will incorporate a feature that allows for user-defined quantum time. This will enable us to test the algorithm under different scenarios and evaluate its performance in varying conditions. Additionally, we will implement a visualization tool that displays the progress of each process in real-time, providing a more intuitive understanding of the scheduling process.

Introduction

Modern operating systems require efficient process scheduling algorithms for multitasking environments

- Round Robin is a widely used and versatile scheduling algorithm
- The algorithm ensures fairness among competing processes and optimizes resource utilization
- A simulation program is presented to assess the performance of the Round Robin algorithm
- The program analyzes the algorithm's impact on workload management, CPU usage, and overall system performance
- Round Robin is useful in environments that require equal time-sharing among processes
- The algorithm efficiently allocates time slices to each process for fair CPU time sharing
- The simulation program evaluates the algorithm's performance in various metrics and provides insight for potential improvement
- Round Robin is critical for optimal resource utilization and fairness among competing processes in modern operating systems
- The simulation program helps understand the algorithm's performance and improve it to meet evolving demands of computer systems.

Round Robin Scheduling

The Round Robin scheduling algorithm is known for its simplicity and fairness. It allocates a fixed time quantum to each process, allowing it to execute for a limited time before moving to the back of the queue. This cyclic nature ensures that no process monopolizes the CPU for an extended period, promoting fairness and responsiveness in the system.

Objectives of the Simulation

Our project is designed to meet several key objectives:

- Process Generation: We generate a set of random processes, each with its own arrival time and CPU burst time. These processes represent real-world tasks that need to be scheduled.
- Round Robin Implementation: We implement the Round Robin scheduling algorithm to manage these processes efficiently. The algorithm runs them in a cyclic fashion, allocating each process a fixed time slice.
- Simulation Execution: The simulation program is executed for a defined period, which allows us to observe how processes are scheduled and how they interact with one another.
- Data Collection: We collect essential data during the simulation, primarily focusing on the average waiting time and turnaround time for each process. These metrics provide insights into process efficiency and system performance.

Requirements of the solution

Process Generation

The simulation program must be capable of generating a variable number of random processes. Each process should possess two critical attributes:

- **Arrival Time:** The time at which a process arrives in the system.
- **CPU Burst Time:** The time a process requires to complete its execution.

These attributes are randomly generated to simulate real-world scenarios where tasks enter the system at different intervals and have varying execution times. A random number generator is employed to ensure the stochastic nature of the processes.

Round Robin Scheduling Algorithm Implementation

- The Round Robin scheduling algorithm segments time into fixed intervals or time slices called time quantum.
- A queue structure, commonly known as the ready queue, manages the processes that are ready to execute.
- The algorithm processes this queue in a cyclic manner.
- Processes are executed for a specific time quantum, allowing for fair distribution of CPU time among competing tasks.
- When a process exhausts its time quantum, it's preempted and placed back in the ready queue.
- The next process in the queue gets a turn, maintaining the fairness of CPU allocation.

Simulation Execution

The simulation program runs for a predetermined duration, simulating the passage of time within the system. The set duration, such as 100 time units, ensures that the program can effectively observe and analyze the behavior of processes as they interact with the scheduler and other processes.

Data Collection

During the simulation execution, the program collects and calculates key performance metrics for each process:

- **Waiting Time:** The total time a process spends waiting in the ready queue before execution.
- **Turnaround Time:** The total time taken for a process to complete from arrival to execution completion.

Step wise Pseudo Code

Github link: **https://github.com/yousufkhn/roundRobin-scheduling-simulation**

```
# Round Robin Scheduling Simulation Pseudo Code
# Step 1: Input Parameters Input: numberOfProcesses, timeQuantum

 # Step 2: Generate Random Processes processes =
GenerateRandomProcesses(numberOfProcesses)

# Step 3: Initialize Variables currentTime = 0 readyQueue = CreateEmptyQueue()
ganttChartData = ""

# Step 4: Sort Processes by Arrival Time SortProcessesByArrivalTime(processes)

# Step 5: Main Simulation Loop while not AllProcessesCompleted(processes):
# 5.1: Check for New Arrivals and Add to Ready Queue
EnqueueNewArrivals(processes, readyQueue, currentTime) if not
IsEmpty(readyQueue):
# 5.2: Execute the Current Process currentProcess = Dequeue(readyQueue)
executionTime = Min(timeQuantum, currentProcess.burstTime)
currentProcess.burstTime -= executionTime # 5.3: Update Gantt Chart
ganttChartData += GenerateGanttChartEntry(currentProcess.id, executionTime) if
currentProcess.burstTime > 0: Enqueue(readyQueue, currentProcess) else:
currentProcess.turnaroundTime = currentTime - currentProcess.arrivalTime else:
 # 5.4: CPU Idle currentTime++

# Step 6: Calculate Averages avgTurnaroundTime =
CalculateAverageTurnaroundTime(processes) avgWaitingTime =
CalculateAverageWaitingTime(processes)

 # Step 7: Output Results OutputResults(processes, avgTurnaroundTime,
avgWaitingTime)
 # Step 8: Save Gantt Chart SaveGanttChartToFile(ganttChartData)

 # Helper Function 1: Generate Random Processes Function
GenerateRandomProcesses(numberOfProcesses): processes = CreateEmptyList() for i
from 1 to numberOfProcesses: arrivalTime = GenerateRandomArrivalTime()
burstTime = GenerateRandomBurstTime() process = CreateProcess(i, arrivalTime,
burstTime, 0, 0) AppendToList(processes, process) return processes
```

# Helper Function 2: Sort Processes by Arrival Time Function SortProcessesByArrivalTime(processes): SortListByArrivalTime(processes)

# Helper Function 23 Generate Gantt Chart Entry Function GenerateGanttChartEntry(processId, executionTime): entry = "Task " + processId entry += RepeatCharacter("-", executionTime) entry += ">\n" return entry

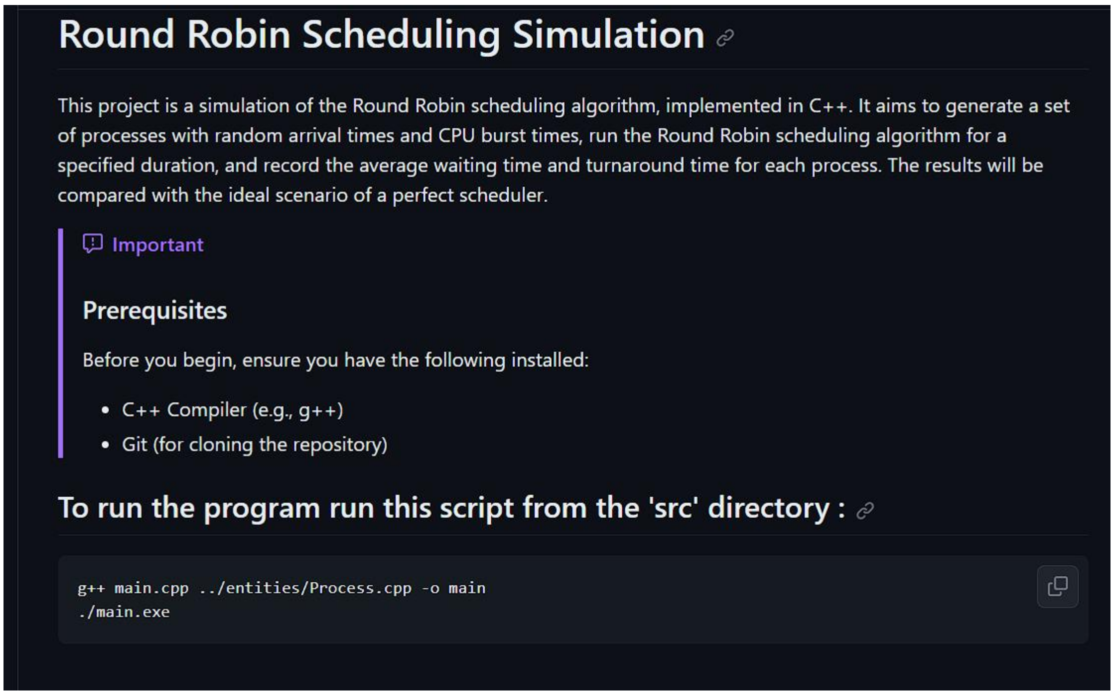
# Helper Function 4: Calculate Average Turnaround Time Function CalculateAverageTurnaroundTime(processes): totalTurnaroundTime = 0 For each process in processes: totalTurnaroundTime += process.turnaroundTime return totalTurnaroundTime / Length(processes)

# Helper Function 5: Calculate Average Waiting Time Function CalculateAverageWaitingTime(processes): totalWaitingTime = 0 For each process in processes: totalWaitingTime += process.waitingTime return totalWaitingTime / Length(processes)

# Helper Function 6: Output Results Function OutputResults(processes, avgTurnaroundTime, avgWaitingTime): Output("Process ID\tWaiting Time\tTurnaround Time") For each process in processes: Output(process.id + "\t\t" + process.waitingTime + "\t\t" + process.turnaroundTime) Output("Average Turnaround Time: " + avgTurnaroundTime + " units") Output("Average Waiting Time: " + avgWaitingTime + " units")

# Helper Function 7: Save Gantt Chart to File Function SaveGanttChartToFile(ganttChartData): WriteToFile("ganttChart.txt", ganttChartData)

<u>Project Snapshots:</u>



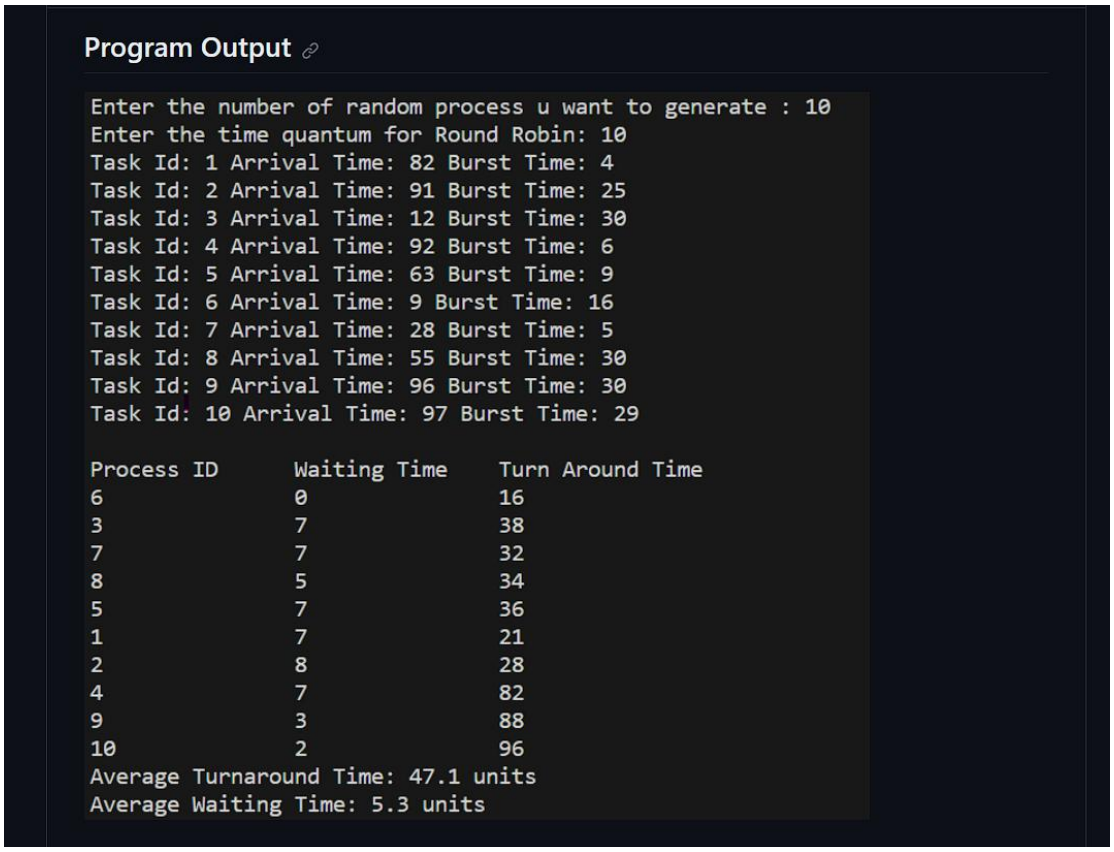
Round Robin Scheduling Simulation

This project is a simulation of the Round Robin scheduling algorithm, implemented in C++. It aims to generate a set of processes with random arrival times and CPU burst times, run the Round Robin scheduling algorithm for a specified duration, and record the average waiting time and turnaround time for each process. The results will be compared with the ideal scenario of a perfect scheduler.

> Important
>
> **Prerequisites**
>
> Before you begin, ensure you have the following installed:
>
> - C++ Compiler (e.g., g++)
> - Git (for cloning the repository)

To run the program run this script from the 'src' directory :

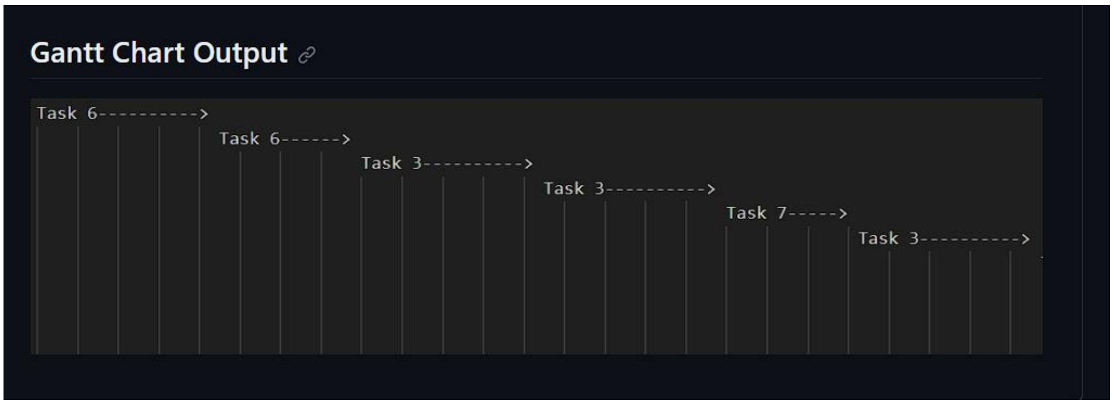
```
g++ main.cpp ../entities/Process.cpp -o main
./main.exe
```



Program Output

```
Enter the number of random process u want to generate : 10
Enter the time quantum for Round Robin: 10
Task Id: 1 Arrival Time: 82 Burst Time: 4
Task Id: 2 Arrival Time: 91 Burst Time: 25
Task Id: 3 Arrival Time: 12 Burst Time: 30
Task Id: 4 Arrival Time: 92 Burst Time: 6
Task Id: 5 Arrival Time: 63 Burst Time: 9
Task Id: 6 Arrival Time: 9 Burst Time: 16
Task Id: 7 Arrival Time: 28 Burst Time: 5
Task Id: 8 Arrival Time: 55 Burst Time: 30
Task Id: 9 Arrival Time: 96 Burst Time: 30
Task Id: 10 Arrival Time: 97 Burst Time: 29

Process ID      Waiting Time    Turn Around Time
6               0               16
3               7               38
7               7               32
8               5               34
5               7               36
1               7               21
2               8               28
4               7               82
9               3               88
10              2               96
Average Turnaround Time: 47.1 units
Average Waiting Time: 5.3 units
```



Gantt Chart Output

```
Task 6---------->
                  Task 6------>
                               Task 3---------->
                                                 Task 3---------->
                                                                  Task 7----->
                                                                               Task 3---------->
```