KARNATAK LAW SOCIETY’S

GOGTE INSTITUTE OF TECHNOLOGY

UDYAMBAG, BELAGAVI-590008

(An Autonomous Institution under Visvesvaraya Technological University, Belagavi)

**(APPROVED BY AICTE, NEW DELHI)**



*Course Activity Report*

*On*

***ROUND ROBIN SCHEDULING ALGORITHM***

*Submitted in the partial fulfillment for the academic requirement**of*

***4th Semester B.E.***

***in***

***Information Science and Engineering***

***Submitted by***

Aryan Kulkarni 2GI19IS009

John Nixon 2GI19IS016

Pratik Desai 2GI19IS037

V Gopinath 2GI19IS056

**GUIDE**

**Mrs. Shubha Sanu**

**2021**

**Title: Round Robin Scheduling Algorithm**

**Team Members Details:**

|  |  |  |
| --- | --- | --- |
| **S. No.** | **USN** | **Student Name** |
| 1 | 2GI19IS009 | Aryan Kulkarni |
| 2 | 2GI19IS016 | John Nixon |
| 3 | 2GI19IS037 | Pratik Desai |
| 4 | 2GI19IS056 | V Gopinath |

**Marks Allocation:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Batch No.: 8** | | | | | |
| 1. | Seminar Title: Round Robin Scheduling Algorithm | Marks Range | **USN** | | | |
| **2GI19IS009** | **2GI19IS016** | **2GI19IS037** | **2GI19IS056** |
| 2. | Abstract (PO2) | 0-2 |  |  |  |  |
| 3. | Application of the topic to the course (PO2) | 0-3 |  |  |  |  |
| 4. | Literature survey and its findings (PO2) | 0-4 |  |  |  |  |
| 5. | Methodology, Results and Conclusion  (PO1, PO3, PO4) | 0-6 |  |  |  |  |
| 6. | Report and Oral presentation skill (PO9, PO10) | 0-5 |  |  |  |  |
|  | Total | 20 |  |  |  |  |

**Signature of Staff**

**What is Round-Robin Scheduling?**

The name of this algorithm comes from the round-robin principle, where each person gets an equal share of something in turns. It is the oldest, simplest scheduling algorithm, which is mostly used for multitasking.

In Round-robin scheduling, each ready task runs turn by turn only in a cyclic queue for a limited time slice. This algorithm also offers starvation free execution of processes.

## Characteristics of Round-Robin Scheduling :

* Round robin is a pre-emptive algorithm
* The CPU is shifted to the next process after fixed interval time, which is called time quantum/time slice.
* The process that is preempted is added to the end of the queue.
* Round robin is a hybrid model which is clock-driven
* Time slice should be minimum, which is assigned for a specific task that needs to be processed. However, it may differ OS to OS.
* It is a real time algorithm which responds to the event within a specific time limit.
* Round robin is one of the oldest, fairest, and easiest algorithm.
* Widely used scheduling method in traditional OS.

## Important terms:

1. **Completion Time** It is the time at which any process completes its execution.
2. **Arrival Time**: **Time** at which the **process** arrives in the ready queue.
3. **Turn Around Time** This mainly indicates the time Difference between completion time and arrival time.
4. The Formula to calculate the turn around time is :**Turn Around Time = Completion Time – Arrival Time**
5. Burst **Time**: **Time** required by a **process** for CPU execution.
6. **Waiting Time(W.T):** It Indicates the time Difference between turn around time and burst time.
7. The formula to calculate waiting time is :**Waiting Time = Turn Around Time – Burst Time**

## Round Robin Scheduling Algorithm Example

## In the following example, there are six processes named as P1, P2, P3, P4, P5 and P6. Their arrival time and burst time are given below in the table. The time quantum of the system is 4 units.

|  |  |  |
| --- | --- | --- |
| Process ID | Arrival Time | Burst Time |
| 1 | 0 | 5 |
| 2 | 1 | 6 |
| 3 | 2 | 3 |
| 4 | 3 | 1 |
| 5 | 4 | 5 |
| 6 | 6 | 4 |

## According to the algorithm, we have to maintain the ready queue and the Gantt chart. The structure of both the data structures will be changed after every scheduling.

## Ready Queue:

## Initially, at time 0, process P1 arrives which will be scheduled for the time slice 4 units. Hence in the ready queue, there will be only one process P1 at starting with CPU burst time 5 units.

|  |
| --- |
| P1 |
| 5 |

## GANTT chart

## The P1 will be executed for 4 units first.

## os RR Scheduling Example GANTT chart

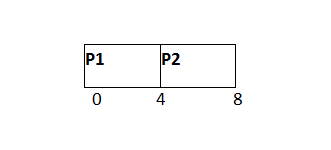
## Ready Queue

## Meanwhile the execution of P1, four more processes P2, P3, P4 and P5 arrives in the ready queue. P1 has not completed yet, it needs another 1 unit of time hence it will also be added back to the ready queue.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| P2 | P3 | P4 | P5 | P1 |
| 6 | 3 | 1 | 5 | 1 |

**GANTT chart**

After P1, P2 will be executed for 4 units of time which is shown in the Gantt chart.



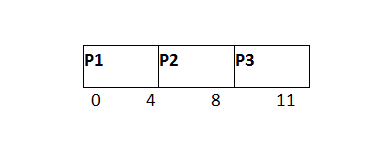
**Ready Queue**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| P3 | P4 | P5 | P1 | P6 | P2 |
| 3 | 1 | 5 | 1 | 4 | 2 |

During the execution of P2, one more process P6 is arrived in the ready queue. Since P2 has not completed yet hence, P2 will also be added back to the ready queue with the remaining burst time 2 units.

**GANTT chart**

After P1 and P2, P3 will get executed for 3 units of time since its CPU burst time is only 3 seconds.



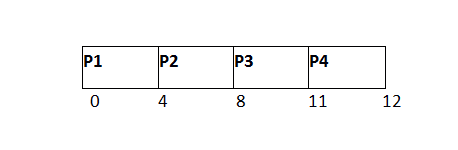
**Ready Queue**

Since P3 has been completed, hence it will be terminated and not be added to the ready queue. The next process will be executed is P4.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| P4 | P5 | P1 | P6 | P2 |
| 1 | 5 | 1 | 4 | 2 |

**GANTT chart**

After, P1, P2 and P3, P4 will get executed. Its burst time is only 1 unit which is lesser then the time quantum hence it will be completed.



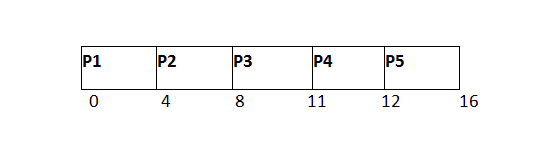
**Ready Queue**

The next process in the ready queue is P5 with 5 units of burst time. Since P4 is completed hence it will not be added back to the queue.

|  |  |  |  |
| --- | --- | --- | --- |
| P5 | P1 | P6 | P2 |
| 5 | 1 | 4 | 2 |

**GANTT chart**

P5 will be executed for the whole time slice because it requires 5 units of burst time which is higher than the time slice.



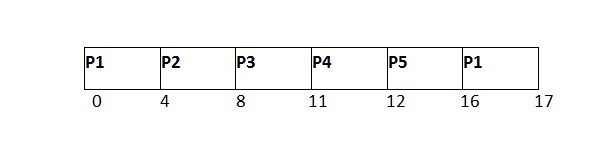
**Ready Queue**

|  |  |  |  |
| --- | --- | --- | --- |
| P1 | P6 | P2 | P5 |
| 1 | 4 | 2 | 1 |

P5 has not been completed yet; it will be added back to the queue with the remaining burst time of 1 unit.

**GANTT Chart**

The process P1 will be given the next turn to complete its execution. Since it only requires 1 unit of burst time hence it will be completed.

****

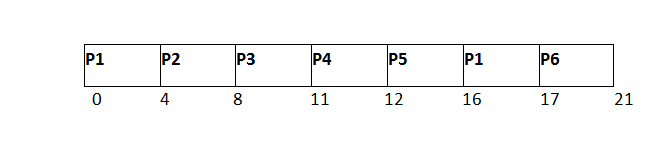
**Ready Queue**

P1 is completed and will not be added back to the ready queue. The next process P6 requires only 4 units of burst time and it will be executed next.

|  |  |  |
| --- | --- | --- |
| **P6** | **P2** | **P5** |
| **4** | **2** | **1** |

**GANTT chart**

P6 will be executed for 4 units of time till completion.

****

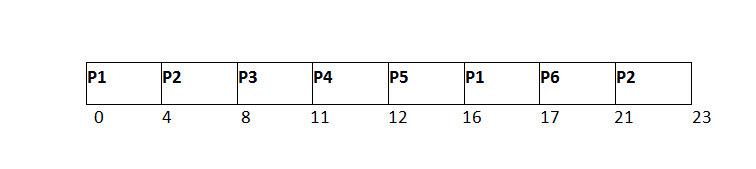
**Ready Queue**

Since P6 is completed, hence it will not be added again to the queue. There are only two processes present in the ready queue. The Next process P2 requires only 2 units of time.

|  |  |
| --- | --- |
| **P2** | **P5** |
| **2** | **1** |

**GANTT Chart**

P2 will get executed again, since it only requires only 2 units of time hence this will be completed.

****

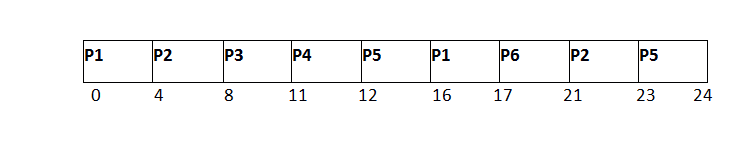
**Ready Queue**

Now, the only available process in the queue is P5 which requires 1 unit of burst time. Since the time slice is of 4 units hence it will be completed in the next burst.

|  |
| --- |
| **P5** |
| **1** |

**GANTT chart**

P5 will get executed till completion.



The completion time, Turnaround time and waiting time will be calculated as shown in the table below.

As, we know,

1. Turn Around Time = Completion Time - Arrival Time
2. Waiting Time = Turn Around Time - Burst Time

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Process ID** | **Arrival Time** | **Burst Time** | **Completion Time** | **Turn Around Time** | **Waiting Time** |
| 1 | 0 | 5 | 17 | 17 | 12 |
| 2 | 1 | 6 | 23 | 22 | 16 |
| 3 | 2 | 3 | 11 | 9 | 6 |
| 4 | 3 | 1 | 12 | 9 | 8 |
| 5 | 4 | 5 | 24 | 20 | 15 |
| 6 | 6 | 4 | 21 | 15 | 11 |

**Avg Waiting Time = 11.33**

**Avg Turn Around Time = 15.33**

**Advantage of Round-robin Scheduling**

* It doesn't face the issues of starvation or convoy effect.
* All the jobs get a fair allocation of CPU.
* It deals with all process without any priority
* If you know the total number of processes on the run queue, then you can also assume the worst-case response time for the same process.
* This scheduling method does not depend upon burst time. That's why it is easily implementable on the system.
* Once a process is executed for a specific set of the period, the process is pre-empted, and another process executes for that given time period.
* Allows OS to use the Context switching method to save states of pre-empted processes.
* It gives the best performance in terms of average response time.

**Disadvantages of Round-robin Scheduling**

* If slicing time of OS is low, the processor output will be reduced.
* This method spends more time on context switching
* Its performance heavily depends on time quantum.
* Priorities cannot be set for the processes.
* Round-robin scheduling doesn't give special priority to more important tasks.
* Decreases comprehension
* Lower time quantum results in higher the context switching overhead in the system.
* Finding a correct time quantum is a quite difficult task in this system.

**REFERENCES:**

* + [**https://en.wikipedia.org/wiki/Round-robin\_scheduling**](https://en.wikipedia.org/wiki/Round-robin_scheduling)
  + [**https://www.geeksforgeeks.org/program-round-robin-scheduling-set-1/**](https://www.geeksforgeeks.org/program-round-robin-scheduling-set-1/)
  + [**https://www.tutorialspoint.com/operating\_system/os\_process\_scheduling\_algorithms.htm**](https://www.tutorialspoint.com/operating_system/os_process_scheduling_algorithms.htm)