Contents

[**USCS3PO1: USCS303 – Operating System (0S) 1**](#_Toc78385942)

[**USCS303-OS:Practical -03:RR Scheduling Algorithm 1**](#_Toc78385943)

[**Round- Robin(RR) Scheduling Algorithm 1**](#_Toc78385944)

[**Practical Date 1**](#_Toc78385945)

[**Practical Aim 1**](#_Toc78385946)

[**Algorithm 2**](#_Toc78385947)

[**Flowchart 3**](#_Toc78385948)

[**Solved Example 3**](#_Toc78385949)

[**Gnatt Chart 7**](#_Toc78385950)

[**Implementation 10**](#_Toc78385951)

[**Input 12**](#_Toc78385952)

[**Output 12**](#_Toc78385953)

[**Sample Output(Screenshots of all the examples) 13**](#_Toc78385954)

# **USCS3PO1: USCS303 – Operating System (0S)**

## **USCS303-OS:Practical -03:RR Scheduling Algorithm**

## **Round- Robin(RR) Scheduling Algorithm**

## 

### **Practical Date:** 28th July ,2021 (Wednesday)

### **Practical Aim :**Implement RR scheduling algorithm

Round-robin (RR) scheduling,but in Round-robin (RR) scheduling ,pre-emption is added which enables the system to switch between processes.

This algorithm is similar to FCFS scheduling , but in Round-robin(RR)scheduling, pre-emption is added, which enables the system to switch between processes.

Round -robin scheduling algorithm is used to schedule process fairly each job a time slot or quantum and the interrupting the job a time slot or quantum and the interrupting the job if it is not completed by then the job come after the other job which is arrived in the quantum time that makes these scheduling fairly.

# **Algorithm :**

**Step 1:** Input the number of process and time quanta or time slice required to be scheduled using RR, burst time for each process.

**Step 2:** Choose the first process in the ready queue, set a timer to interrupt it after time quantum and dispatches it. Check if any other process request has arrived if a process request arrives during the quantum time in which another process is executing ,then add the new process to the ready queue.

**Step 3:** After the quantum time has passed ,check for any process in the Ready queue . If the ready queue is empty then continue the current process . If the queue not empty and the current process is completed, then add the current process to the end of the ready queue.

**Step 4:** Take the first process from the Ready queue and start executing it.Calculate the Turn Around Time and Waiting Time for each processing using RR.

**Step 5:** Reapeat all steps above from Step2 to Step4.

**Step 6:** If the process is complete and the ready queue is empty then the task is complete.

**Step 7:**Calculate the Average Waiting Time and Average Turn Around Time .

**Step 8:**Stop.

### 

### **Flowchart :**

### **RR Flowchart**

### 

### Select a process

**Burst Time < Time Quantum**

Yes

**Execute till completion**

No

No

**Execute for Time Quantum**

**Termination**

Time Quantum Expires

**Process executed completely?**

Yes

### **Solved Example**

### **Consider the following example containing three processes arriving at time t= 0 ms**

### 

|  |  |
| --- | --- |
| **Process ID** | **Burst Time** |
| **P0** | **24** |
| **P1** | **3** |
| **P2** | **3** |

### **Assume Time Quanta :4ms**

### **Step1:**Consider the time quanta /time slice = 4 ms.

### **Step 2:**Following shows the scheduling and execution of processes .

**Step 2.1:**P0 process arrives at 0 with 24 ms as the Burst time which is greater than time quanta = 4 ms. So P0 executes for 4 ms and goes in waiting queue .

System Time : 0

Process Schedule :P0

Remaining Time :24 - 4 =20

Waiting Time : 0 - 0 = 0

Turn Around Time : 0 + 4 =4

**Step 2.2 :** Next P1 process executes for 3 ms which is greater than quanta time. So P1 executes and gets terminated .

System Time : 4

Process Schedule : P0,P1

Remaining Time : 3 - 4 =-1=0

Waiting Time : 4 - 0 = 4

Turn Around Time : 4 + 3 =7

**Step 2.3 :** Next P2 process executes for 3 ms which is greater than quanta time. So P2 executes and gets terminated .

System Time : 7

Process Schedule :P0,P1,P2

Remaining Time :3 - 4 =-1= 0

Waiting Time : 7 - 0 = 7

Turn Around Time : 7 + 3 =10

### 

**Step 2.4 :**Now P0 turns comes again and it’s the only process for execution so for 4 ms of quanta it gets executed .

System Time : 10

Process Schedule :P0,P1,P2,P0

Remaining Time :20- 4 =16

Waiting Time : 0

Turn Around Time : 10 + 4 = 14

**Step 2.5:** Again , P0 continues to execute for next 4 ms . Waiting for P0 will be zero.

System Time : 14

Process Schedule :P0,P1,P2,P0,P0

Remaining Time :16 - 4 =12

Waiting Time : 0

Turn Around Time : 14 + 4 = 18

**Step 2.6 :** P0 continues to execute for next 4 ms.

System Time : 18

Process Schedule :P0,P1,P2,P0,P0,P0

Remaining Time :12 - 4 = 8

Waiting Time : 0

Turn Around Time : 18 + 4 =22

**Step 2.7 :** P0 continues to execute for next 4 ms.

System Time : 22

Process Schedule :P0,P1,P2,P0,P0,P0,P0

Remaining Time : 8 - 4 = 4

Waiting Time : 0

Turn Around Time : 22 + 4 = 26

**Step 2.8 :** P0 continues to execute for next 4 ms.

System Time : 26

Process Schedule :P0,P1,P2,P0,P0,P0,P0,P0

Remaining Time : 4 - 4 = 0

Waiting Time : 0

Turn Around Time : 26 + 4 = 30

**Step 3:** Calculate the Average Waiting Time and Average Turn Around Time.

Average Waiting Time =( 6 + 4 + 7 )/3

=17/3

=5.666667

Average Turn Around Time =(30 + 7 + 10)/3

=47/3

=15.6666667

**Step 4 :** After Scheduling of all provided process:

|  |  |  |  |
| --- | --- | --- | --- |
| **Process ID** | **Burst Time** | **Turn Around Time**  **(Completion Time – Arrival Time)** | **Waiting Time (Turn Around Time – Burst Time)** |
| P0 | 24 | 30 – 0 = 30 | 30 – 24 = 6 |
| P1 | 3 | 4 + 3 = 7 | 7 – 3 = 4 |
| P2 | 3 | 7 + 3 = 10 | 10 – 3 = 7 |
| Average |  | 15.66666667 | 5.6666667 |

### **Gnatt Chart :**



|  |  |  |  |
| --- | --- | --- | --- |
| **Process ID** | **Burst Time** | **Turn Around Time**  **(Completion Time – Arrival Time)** | **Waiting Time (Turn Around Time – Burst Time)** |
| P0 | 24 | 30 – 0 = 30 | 30 – 24 = 6 |
| P1 | 3 | 4 + 3 = 7 | 7 – 3 = 4 |
| P2 | 3 | 7 + 3 = 10 | 10 – 3 = 7 |
| Average |  | 15.66666667 | 5.6666667 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| P0 | P1 | P2 | P0 | P0 | P0 | P0 | P0 |

0 4 7 10 14 18 22 26 30

|  |  |
| --- | --- |
| **Process ID** | **Burst Time** |
| **P0** | **2** |
| **P1** | **1** |
| **P2** | **6** |

### **Assume Time Quanta :1ms**

|  |  |  |  |
| --- | --- | --- | --- |
| **Process ID** | **Burst Time** | **Turn Around Time**  **(Completion Time – Arrival Time)** | **Waiting Time (Turn Around Time – Burst Time)** |
| P0 | 2 | 4 – 0 = 4 | 30 – 24 = 2 |
| P1 | 1 | 1 + 1 = 2 | 7 – 3 = 1 |
| P2 | 6 | 4 + 5 = 9 | 10 – 3 = 3 |
| Average |  | 5.0000000 | 2.0000000 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P0 | P1 | P2 | P0 | P2 | P2 | P2 | P2 | P2 |

0 1 2 3 4 5 6 7 8 9

3.

|  |  |
| --- | --- |
| **Process ID** | **Burst Time** |
| **P0** | 7 |
| **P1** | 3 |
| **P2** | 2 |
| **P3** | 10 |
| **P4** | 8 |
|  |  |

### **Assume Time Quanta :3ms**

|  |  |  |  |
| --- | --- | --- | --- |
| **Process ID** | **Burst Time** | **Turn Around Time**  **(Completion Time – Arrival Time)** | **Waiting Time (Turn Around Time – Burst Time)** |
| P0 | 7 | 24 – 0 = 24 | 24 – 7 = 17 |
| P1 | 3 | 3 + 3 = 6 | 6 – 3 = 3 |
| P2 | 2 | 6 + 2 = 8 | 8 – 2 = 6 |
| P3 | 10 | 30 – 0 = 30 | 30 – 10 = 20 |
| P4 | 8 | 29 – 0 =29 | 29 – 8 = 21 |
| **Average** |  | **19.40000000** | **13.4000000** |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P0 | P1 | P2 | P3 | P4 | P0 | P3 | P4 | P0 | P3 | P4 | P3 |

0 3 6 8 11 14 17 20 23 24 27 29 30

### **Implementation**

### //Name:Jadhav Sahil

### //Batch: B2

### //PRN :2020016400783091

### //Date :28th July ,2021

### //Filename:P3\_RR\_SJ

### import java.util.Scanner;

### class P3\_RR\_SJ

### {

### public static void main(String args[])

### {

### Scanner input = new Scanner(System.in);

### int i, j, k, q, sum = 0;

### System.out.print("Enter number of process: ");

### int n = input.nextInt();

### int burstTime[] = new int[n];

### int waitingTime[] = new int[n];

### int turnAroundTime[] = new int[n];

### int a[] = new int[n];

### System.out.println("Enter Burst Time of each process: ");

### for (i = 0; i < n; i++)

### {

### System.out.print("Enter Burst Time for Process - P" + (i + 1) + ":");

### burstTime[i] = input.nextInt();

### a[i] = burstTime[i];

### }

### System.out.print("Enter Time quantum: ");

### q=input.nextInt();

### for (i = 0; i<n; i++)

### waitingTime[i] = 0;

### int timer = 0; // Current time

### // Keep traversing processes in round robin manner until all of them are not done.

### do{

### for (i = 0; i<n; i++)

### {

### // If burst time of a process is greater than 0 then only need to process further

### if (burstTime[i] > q)

### {

### // Increase the value of ti.e. shows how much time a process has been processed

### timer += q;

### // Decrease the burst time of current process by quantum

### burstTime[i] -= q;

### for (j = 0; j<n; j++)

### {

### if ((j!= i) && (burstTime[j] != 0))

### waitingTime[j] += q;

### }

### } // if ends

### // If burst time is smaller than or equal to quantum. Last cycle for this process

### else

### {

### // Increase the value of t i.e. shows how much time a process has been processed

### timer += burstTime[i];

### for (j = 0; j<n; j++)

### {

### if ((j != i) && (burstTime[j] != 0))

### waitingTime[j] += burstTime[i];

### }

### // As the process gets fully executed make its remaining burst time = 0

### burstTime[i] = 0;

### } // else ends

### }

### sum = 0;

### for (k = 0; k<n; k++)

### sum += burstTime[k];

### } while (sum != 0);

### // calculating turnaround time by adding waiting Time + burst Time

### for (i = 0; i<n; i++)

### turnAroundTime[i] = waitingTime[i] + a[i];

### float total = 0;

### for (int x : waitingTime)

### {

### total += x;

### }

### float averageWaitingTime = total / n;

### total = 0;

### for (int y : turnAroundTime)

### {

### total += y;

### }

### float averageTurnAroundTime = total / n;

### // print on console the order of processes scheduled using Round-robin Algorithm

### System.out.println("RR Algorithm: ");

### System.out.format("%20s%20s%20s%20s\n","Processld","BurstTime","Waiting Time","TurnAroundTime");

### for (i = 0; i<n; i++)

### {

### System.out.format("%20s%20d%20d%20d\n","P"+(i),a[i],

### waitingTime[i],turnAroundTime[i]);

### }

### System.out.format("%40s%20f%20f\n","Average",averageWaitingTime,averageTurnAroundTime);

### }

### }

### **Input:**

**Enter number of process : 3**

**Enter Burst Time for each process:**

**Enter Burst Time for process -P0: 24**

**Enter Burst Time for process -P1: 3**

**Enter Burst Time for process -P2: 3**

**Enter Time Quantum : 4**

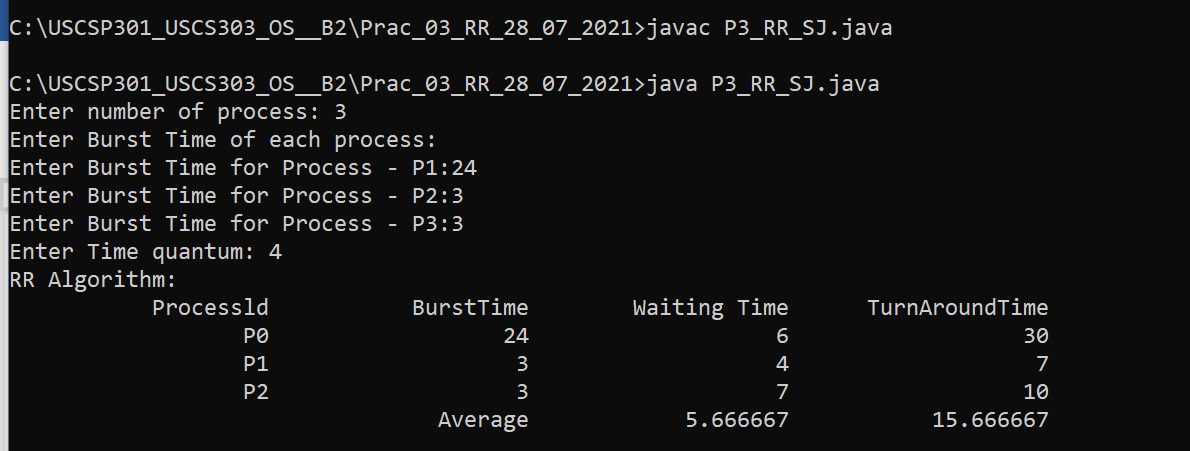
### **Output :**

|  |  |  |  |
| --- | --- | --- | --- |
| **Process ID** | **Burst Time** | **Turn Around Time**  **(Completion Time – Arrival Time)** | **Waiting Time (Turn Around Time – Burst Time)** |
| P0 | 24 | 30 – 0 = 30 | 30 – 24 = 6 |
| P1 | 3 | 4 + 3 = 7 | 7 – 3 = 4 |
| P2 | 3 | 7 + 3 = 10 | 10 – 3 = 7 |
| Average |  | 15.66666667 | 5.6666667 |

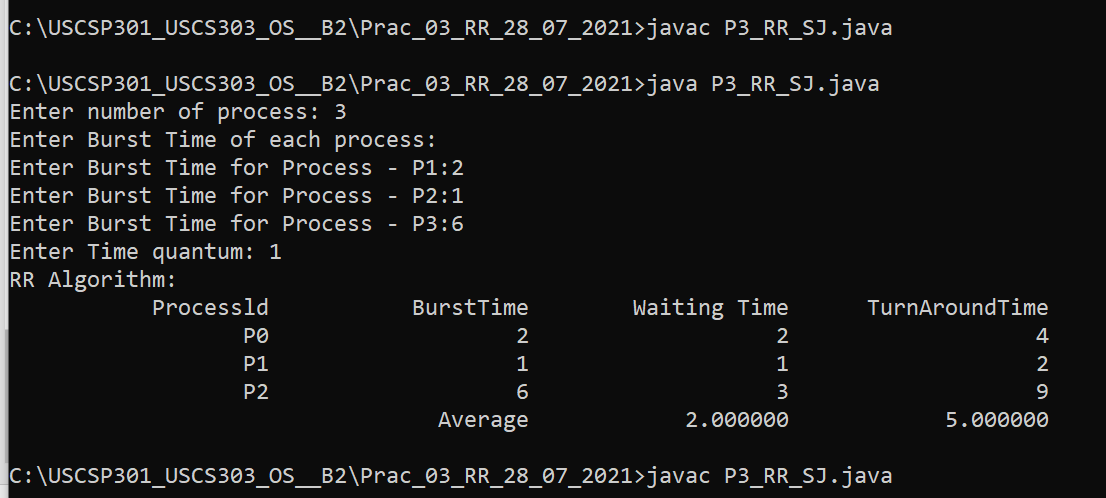
### 

### **Sample Output(Screenshots of all the examples) :**

### **Question 1:**



### **Question 2:**



**Question 3:**

