Mawlana Bhashani Science and Technology University

Lab-Report

Report No: 10

Course code: ICT-3110

Course title: Operating System Lab

Date of Performance:

Date of Submission: 09/09/2020

Submitted by

Name: Shakhera khanom

ID: IT-18033

3rd year 1st semester

Session: 2017-2018

Dept. of ICT

MBSTU.

Submitted To

Nazrul Islam

Assistant Professor

Dept. of ICT

MBSTU.

Experiment No: 10

Experiment Name: Implementation of Round Robin Scheduling Algorithm.

Objectives:

- What is Round Robin Scheduling Algorithm?
- How to implement in c?

✞ **Round Robin Scheduling Algorithm:**

→ Round Robin is the preemptive process scheduling algorithm

→ Each process is provided a fix time to execute, it is called a quantum.

→ Once a process is execute for a given time period, it is preempted and other process executes for a given time period.

Aim: To write a c program to implement the CPU scheduling round robin algorithm.

Description:

To aim is to calculate the average waiting time. There will be a time slice, each process should be executed within that time-slice and if not it will go to the waiting state so first check whether the burst time is less than the time-slice. If it is less than it assign the waiting time to the sum of the total times. If it is greater than the burst-time then subtract the time slot from the actual burst time and increment it by time-slot and the loop continues until all the processes are completed.

Algorithm:

Step 1: Start the process

Step 2: Accept the number of processes in the ready Queue and time quantum (or) time slice

Step 3: For each process in the ready Q, assign the process id and accept the

CPU burst time

Step 4: Calculate the no. of time slices for each process where

No. of time slice for process(n) = burst time process(n)/time slice

Step 5: If the burst time is less than the time slice then the no. of time slices =1.

Step 6: Consider the ready queue is a circular Q, calculate

a. Waiting time for process(n) = waiting time of process(n-1)+ burst time of process(n-1) + the time difference in getting the CPU from process(n-1)
b. Turnaround time for process(n) = waiting time of process(n) + burst time of process(n)+ the time difference in getting CPU from process(n).

Step 7: Calculate

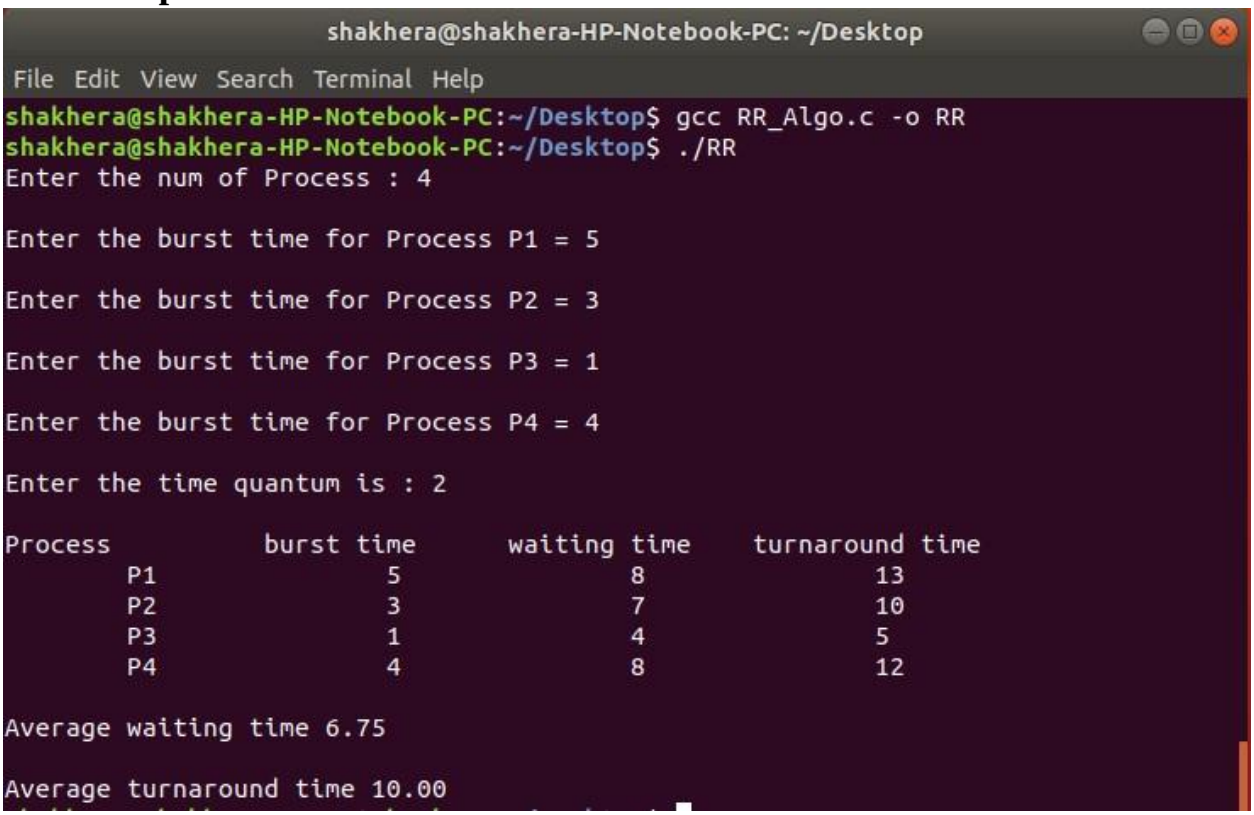
a. Average waiting time = Total waiting Time / Number of process
b. Average Turnaround time = Total Turnaround Time / Number of process **Step 8:** Stop the process

Corresponding Code:

```
#include<stdio.h>
int main()
{
   int n,i,j,time,max,burst_time[30],complete_t[30],waiting_t[30],turnaround_t[30];
   float avg_waiting_t,avg_turnaround_t,temp=0;
   printf("Enter the num of Process : ");
   scanf("%d",&n);
   for(i=0;i<n;i++)
   {
      printf("\nEnter the burst time for Process P%d = ",i+1);
      scanf("%d",&burst_time[i]);
      complete_t[i]=burst_time[i];
   }
   printf("\nEnter the time quantum is : ");
   scanf("%d",&time);
   max=burst_time[0];
   for(i=1;i<n;i++)
   {
      if(max<burst_time[i])
      {
         max=burst_time[i];
      }
   }
   for(i=0;i<(max/time)+1;i++)
   {
      for(j=0;j<n;j++)
      {
         if(burst_time[j]!=0)
         {
```

```
            if(burst_time[j]<=time)
            {
               turnaround_t[j]=temp+burst_time[j];
               temp=temp+burst_time[j];
               burst_time[j]=0;
            }
            else
            {
               burst_time[j]=burst_time[j]-time;
               temp=temp+time;
            }
         }
       }
    }
    printf("\nProcess \t burst time \t waiting time \t turnaround time \n");
    avg_waiting_t=avg_turnaround_t=0;
    for(i=0;i<n;i++)
    {
        waiting_t[i]=turnaround_t[i]-complete_t[i];
        avg_turnaround_t=avg_turnaround_t+turnaround_t[i];
        avg_waiting_t=avg_waiting_t+waiting_t[i];
    }
    for(i=0;i<n;i++)
    {
        printf("\tP%d\t\t%d\t\t%d\t\t%d\n",i+1,complete_t[i],waiting_t[i],turnaround_t[i]);
    }
    avg_waiting_t=avg_waiting_t/n;
    avg_turnaround_t=avg_turnaround_t/n;
    printf("\nAverage waiting time %.2f\n",avg_waiting_t);
    printf("\nAverage turnaround time %.2f\n",avg_turnaround_t);
    return 0;
}
```

Output:



Discussion:

This lab helps to learn Round Robin (RR) algorithm. In round robin scheduling , each ready task runs turn by turn only in a cyclic queue for a limited time slice.