Mawlana Bhashani Science and Technology University

Lab-Report

Report No:10

Lab Report Name: Implementation of Round Robin Scheduling Algorithm.

Course code: ICT-3110

Course title: Operating System Lab

Date of Performance: 18-09-2020

Date of Submission:

Submitted by

Name: Tazneen Akter

ID: IT-18056

3rd year 1st semester

Session: 2017-18

Submitted to

Nazrul Islam

Assistant Professor

Dept. of ICT, MBSTU.

Experiment No : 10

Experiment Name : Implementation of Round Robin scheduling algorithm.

Theory : 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.

• 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.

Working Process:

```
#include<stdio.h>
int main()
{
 int n,i,k,x=0,s=0,r=0,q=0,a[30],e[30],t[30];
 float m,p=0;
 printf("Enter the number of process: ");
 scanf("%d",&n);
```

```
printf("Enter the execution time: ");
for(i=0; i<n; i++)
{
scanf("%d",&a[i]);
e[i]=a[i];
}
printf("Enter the quanta: ");
scanf("%d",&q);
printf("After Round Robin sheduling: ");
for(i=0; i<n; i++)
{
if(x<a[i])
{
x=a[i];
}
}
k=x/q;
while(s<=k)
{
for(i=0; i<n; i++)
{
if(a[i]>0)
{
if(a[i]>q)
{
r=r+q;
a[i]=a[i]-q;
printf("P%d\t",i+1);
}else
{
r=r+a[i];
a[i]=a[i]-q;
printf("P%d ",i+1);
t[i]=r;
}
}
}
```

```
 s++;
 }
 printf("\n\nProcess  BurstTime   WaitingTime  TurnAroundTime\n");
 for(i=0; i<n; i++)
 {
 printf(" %d \t\t %d\t\t %d\t\t %d\t\t \n",i,e[i],x,t[i]);
 x=x+q;
 }
 m=x/n;
 printf("\nAverage waiting time=%f= ",m);
 printf("\nAverage turn around time= ");
 for(i=0; i<n; i++)
 p=p+t[i];
 p=p/n;
 printf("%f",p);
 printf("\n");
 return 0;
}
```

Output:

```
Enter the number of process: 4
Enter the execution time: 24 3 5 7
Enter the quanta: 4
After Round Robin sheduling: P1 P2 P3   P4      P1      P3 P4 P1        P1      P1      P1

Process  BurstTime   WaitingTime  TurnAroundTime
 0              24              24              39
 1              3               28              7
 2              5               32              20
 3              7               36              23

Average waiting time=10.000000=
Average turn around time= 22.250000

Process returned 0 (0x0)   execution time : 7.333 s
Press any key to continue.
```

Discussion: In this lab we have implemented Round Robin scheduling algorithm using C language. By solving this problem in future we can solve any problem of this algorithm.