Mawlana Bhashani Science and Technology University

Lab-Report

Report No:09

Course code:ICT-3110

Course title:Operating Systems Lab

Date of Performance:09-09-2020

Date of Submission:16-09-2020

Submitted by

Name:Meraz Ahmed

ID:IT-18005

3rd year 1st semester

Session: 2017-2018

Dept. of ICT

MBSTU.

Submitted To

Nazrul Islam

Assistant Professor

Dept. of ICT

MBSTU.

Experiment No: **09**

Experiment Name: **Implementation of Priority Scheduling Algorithm.**

Objectives:

- Here we will learn what is PriorityScheduling algorithm and how it work.

Priority scheduling Algorithm:

In priority scheduling algorithm each process has a priority associated with it and as each process hits the queue, it is stored in based on its priority so that process with higher priority are dealt with first. It should be noted that equal priority processes are scheduled in FCFS order.

To prevent high priority processes from running indefinitely the scheduler may decrease the priority of the currently running process at each clock tick (i.e., at each clock interrupt). If this action causes its priority to drop below that of the next highest process, a process switch occurs. Alternatively, each process may be assigned a maximum time quantum that it is allowed to run. When this quantum is used up, the next highest priority process is given a chance to run.

Example:

Process	P1	P2	P3	P4	P5
Burst time	5	13	8	6	12
Priority	1	3	0	4	2

PsudoCode:

```
#include<stdio.h>

int main()
{
int bt[20],p[20],wt[20],tat[20],pr[20],i,j,n,total=0,pos,temp,avg_wt,avg_tat;
printf("Enter Total Number of Process:");
scanf("%d",&n);

printf("\nEnter Burst Time and Priority\n");
for(i=0;i<n;i++)
```

```
{
printf("\nP[%d]\n",i+1);
printf("Burst Time:");
scanf("%d",&bt[i]);
printf("Priority:");
scanf("%d",&pr[i]);
p[i]=i+1;           //contains process number
}

//sorting burst time, priority and process number in ascending order using selection sort
for(i=0;i<n;i++)
{
pos=i;
for(j=i+1;j<n;j++)
{
if(pr[j]<pr[pos])
pos=j;
}

temp=pr[i];
pr[i]=pr[pos];
pr[pos]=temp;

temp=bt[i];
bt[i]=bt[pos];
bt[pos]=temp;

temp=p[i];
p[i]=p[pos];
p[pos]=temp;
}

wt[0]=0;    //waiting time for first process is zero

//calculate waiting time
for(i=1;i<n;i++)
{
wt[i]=0;
for(j=0;j<i;j++)
wt[i]+=bt[j];

total+=wt[i];
}

avg_wt=total/n;      //average waiting time
total=0;
```

```
printf("\nProcess\t    Burst Time    \tWaiting Time\tTurnaround Time");
for(i=0;i<n;i++)
{
tat[i]=bt[i]+wt[i];     //calculate turnaround time
total+=tat[i];
printf("\nP[%d]\t\t  %d\t\t    %d\t\t\t%d",p[i],bt[i],wt[i],tat[i]);
}

avg_tat=total/n;     //average turnaround time
printf("\n\nAverage Waiting Time=%d",avg_wt);
printf("\nAverage Turnaround Time=%d\n",avg_tat);

return 0;
}
```

Output:

```
Enter Total Number of Process:5

Enter Burst Time and Priority

P[1]
Burst Time:5
Priority:1

P[2]
Burst Time:13
Priority:3

P[3]
Burst Time:8
Priority:0

P[4]
Burst Time:6
Priority:4

P[5]
Burst Time:12
Priority:2

Process     Burst Time          Waiting Time    Turnaround Time
P[3]              8                 0                  8
P[1]              5                 8                  13
P[5]              12                13                 25
P[2]              13                25                 38
P[4]              6                 38                 44

Average Waiting Time=16
Average Turnaround Time=25

Process returned 0 (0x0)   execution time : 80.441 s
Press any key to continue.
```