Mawlana Bhashani Science and Technology University

Lab-Report

Report No: 09

Course code: ICT-3110

Course title: Operating Systems Lab

Date of Performance:

Date of Submission: 15/09/2020

Submitted by

Name: Md Ahadul Haque

ID:IT-18045

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**.**

Theory :

priorities can be either dynamic or static. Static priorities are allocated during creation, whereas dynamic priorities are assigned depending on the behavior of the processes while in the system. To illustrate, the scheduler could favor input/output (I/O) intensive tasks, which lets expensive requests to be issued as soon as possible.

Implementation:

1. priority scheduling program in C, we consider the **arrival time** of the processes..

2. The CPU can leave the process midway. The current state of the process will be saved by the context switch**.**

3. The system can then search for another process with a higher priority in the ready queue or waiting queue and start its execution.

4. Once the CPU comes back to the previous incomplete process, the job is resumed from where it was earlier paused.

Working Process :

Code for Priority Scheduling Algorithm –

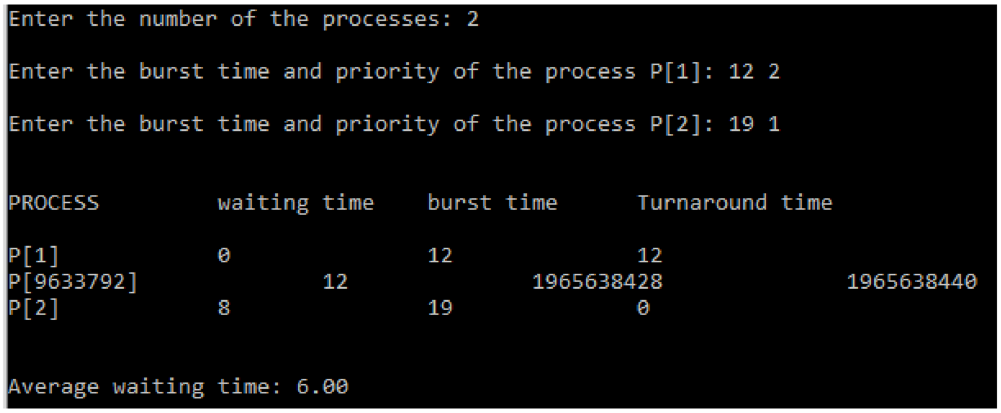
```
#include <stdio.h>
int main()
{

    int bt[20],wt[20],p[20],tat[20],priority[20];
    float avwt=0,avtat=0;

    int i,j,n,temp,key;

    printf("\nEnter the number of the processes: ");

    scanf("%d",&n);

    for(i=1;i<=n;i++)

    {

      printf("\nEnter the burst time and priority of the process P[%d]: ",i);

      scanf("%d",&bt[i]);
      scanf("%d",&priority[i]);
      p[i]=i;

    }
```

```
for(i=0;i<n;i++)
{
  key=i;
  for(j=i+1;j<n;j++)
  {
    if(priority[j]<priority[key])
    {
      key=j;
    }
  }
  temp=bt[i];
  bt[i]=bt[key];
  bt[key]=temp;

  temp=priority[i];
  priority[i]=priority[key];
  priority[key]=temp;

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


wt[0]=0;
tat[0]=bt[0];
avtat=tat[0];
```

```
for(i=1;i<n;i++)
{

    wt[i]=wt[i-1]+bt[i-1];

    tat[i]=tat[i-1]+bt[i];

    avwt+=wt[i];

    avtat+=tat[i];

}

avwt=avwt/n;
avtat=avtat/n;

printf("\n\nPROCESS\t\twaiting time\tburst time\tTurnaround time\n");

printf("\n");

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

    printf("P[%d]\t\t%d\t\t%d\t\t%d\n",p[i],wt[i],bt[i],tat[i]);

}
```

```
    printf("\n\nAverage waiting time: %.2f",avwt);
    printf("\n\nAverage Turn around time is: %.2f",avtat);
    printf("\n");

    return 0;

}
```

Output:



```
Enter the number of the processes: 2

Enter the burst time and priority of the process P[1]: 12 2

Enter the burst time and priority of the process P[2]: 19 1


PROCESS         waiting time    burst time      Turnaround time

P[1]            0               12              12
P[9633792]              12              1965638428              1965638440
P[2]            8               19              0


Average waiting time: 6.00
```

Discussion :

The priority of process is selected on the basis of memory requirement, user preference or the requirement of time. Processes are executed on the basis of priority. So high priority does not need to wait for long which saves time. It is a user friendly algorithm. it has reasonable support for priority.