SVKM'S NMIMS Deemed-to-be-University
Mukesh Patel School of Technology Management & Engineering
Department of Computer Engineering

Course Code		Program	B.Tech.
Semester	V	Year	III
Name of the Faculty	Prof. Mohini Reddy	Class	
Course Title	Operating Systems	Academic year	2022-23

PART A
(PART A : TO BE REFFERED BY STUDENTS)

Experiment No. 06

A.1—Aim:
Study various Process Scheduling Algorithm and implementation of **Priority Scheduling** algorithm for scheduling using 5 Process count.

A.2--- Prerequisite:
Concepts of Process & Process Scheduling

A.3--- Outcome:
After successful completion of this experiment students will be able to:
1. Understand the basics of Process & Process Scheduling.
2. Implement Preemptive & Non-Preemptive Priority Scheduling Algorithm

A.4--- Theory:

The basic idea is straightforward: each process is assigned a priority, and priority is allowed to run. Equal-Priority processes are scheduled in FCFS order. The shortest-Job-First (SJF) algorithm is a special case of general priority scheduling algorithm. An SJF algorithm is simply a priority algorithm where the priority is the inverse of the (predicted) next CPU burst. That is, the longer the CPU burst, the lower the priority and vice versa.

Priority can be defined either internally or externally. Internally defined priorities use some measurable quantities or qualities to compute priority of a process.

Examples of Internal priorities are

- Time limits.
- Memory requirements.
- File requirements,
  for example, number of open files.
- CPU Vs I/O requirements.

Externally defined priorities are set by criteria that are external to operating system such as

- The importance of process.

- Type or amount of funds being paid for computer use.
- The department sponsoring the work.
- Politics.

Priority scheduling can be either pre-emptive or non-pre-emptive

- A preemptive priority algorithm will preemptive the CPU if the priority of the newly arrival process is higher than the priority of the currently running process.
- A non-preemptive priority algorithm will simply put the new process at the head of the ready queue.

A major problem with priority scheduling is indefinite blocking or starvation. A solution to the problem of indefinite blockage of the low-priority process is *aging*. Aging is a technique of gradually increasing the priority of processes that wait in the system for a long period of time.

A.5--- Procedure:
Task:
1. Study Preemptive & Non-Preemptive Priority Scheduling Algorithm
2. Implement Preemptive & Non-Preemptive Priority Scheduling Algorithm with 5 processes.
3. Save and close the file and name it as **EXP5_ your Roll no.**

SVKM'S NMIMS Deemed-to-be-University
Mukesh Patel School of Technology Management & Engineering
Department of Computer Engineering

Course Code		Program	B.Tech.
Semester	V	Year	III
Name of the Faculty	Prof. Mohini Reddy	Class	
Course Title	Operating Systems	Academic year	2022-23

PART B

(PART B: TO BE COMPLETED BY STUDENTS)

Roll No:B090	Name:Vedant Sahai
Class: Btech CE	Batch:B2
Date of Experiment	Date of Submission
Grade:	

B.1 Work done by student

Non preemptive priority Scheduling(C++)

```
//C++ Implementation of Non-Preemptive Priority Scheduling
Algorithm
#include <iostream>
using namespace std;

        int main()
        {
        int n = 7;          //Number of Processes
        int CPU = 0;        //CPU Current time
        int allTime = 0; // Time needed to finish all processes

        int arrivaltime[n] = {0, 1, 3, 4, 5, 6, 10};
        int bursttime[n] = {8,2,4,1,6,5,1};
        int priority[n] = {3,4,4,5,2,6,1};
        int ATt[n];
        int NoP = n; //number of Processes
        int PPt[n];
        int waitingTime[n];
        int turnaroundTime[n];
        int ct[n];
        int i = 0;

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

Course Code		Program	B.Tech.
Semester	V	Year	III
Name of the Faculty	Prof. Mohini Reddy	Class	
Course Title	Operating Systems	Academic year	2022-23

```
        PPt[i] = priority[i];
        ATt[i] = arrivaltime[i];
        }

        int LAT = 0; //LastArrivalTime
        for (i = 0; i < n; i++)
        if (arrivaltime[i] > LAT)
        LAT = arrivaltime[i];

        int MAX_P = 0; //Max Priority
        for (i = 0; i < n; i++)
        if (PPt[i] > MAX_P)
        MAX_P = PPt[i];

        int ATi = 0;       //Pointing to Arrival Time indix
        int P1 = PPt[0]; //Pointing to 1st priority Value
        int P2 = PPt[0]; //Pointing to 2nd priority Value

        //finding the First Arrival Time and Highest priority
Process
        int j = -1;
        while (NoP > 0 && CPU <= 1000)
        {
        for (i = 0; i < n; i++)
        {
        if ((ATt[i] <= CPU) && (ATt[i] != (LAT + 10)))
        {
        if (PPt[i] != (MAX_P + 1))
        {
        P2 = PPt[i];
        j = 1;

        if (P2 < P1)
        {
        j = 1;
        ATi = i;
        P1 = PPt[i];
        P2 = PPt[i];
        }
        }
        }
        }
```

Course Code		Program	B.Tech.
Semester	V	Year	III
Name of the Faculty	Prof. Mohini Reddy	Class	
Course Title	Operating Systems	Academic year	2022-23

```
        if (j == -1)
        {
        CPU = CPU + 1;
        continue;
        }
        else
        {
        waitingTime[ATi] = CPU - ATt[ATi];
        CPU = CPU + bursttime[ATi];
        turnaroundTime[ATi] = CPU - ATt[ATi];
        ATt[ATi] = LAT + 10;
        j = -1;
        PPt[ATi] = MAX_P + 1;
        ATi = 0;        //Pointing to Arrival Time index
        P1 = MAX_P + 1; //Pointing to 1st priority Value
        P2 = MAX_P + 1; //Pointing to 2nd priority Value
        NoP = NoP - 1;
        }
        }

        cout << "\nProcess BurstTime Priority ArrivalTime
Completiontime WaitingTime Turnaround Time\n";
        for (i = 0; i < n; i++)
        {
        ct[i] = turnaroundTime[i] + arrivaltime[i];
        cout << "   " << "P" << i + 1 << "\t   " << bursttime[i] <<
"\t    " << priority[i] << "\t\t" << arrivaltime[i] << "\t\t"
<<ct[i]<<"\t    " << waitingTime[i] << "\t\t" << turnaroundTime[i]
<< endl;
        }

        float AvgWT = 0;  //Average waiting time
        float AVGTaT = 0; // Average Turn around time
        for (i = 0; i < n; i++)
        {
        AvgWT = waitingTime[i] + AvgWT;
        AVGTaT = turnaroundTime[i] + AVGTaT;
        }

        cout << "Average waiting time = " << AvgWT / n << endl;
        cout << "Average turnaround time = " << AVGTaT / n << endl;
        }
```

```
Process BurstTime Priority ArrivalTime  Completiontime WaitingTime Turnaround Time
  P1      8         3           0            8          0           8
  P2      2         4           1            17         14          16
  P3      4         4           3            21         14          18
  P4      1         5           4            22         17          18
  P5      6         2           5            14         3           9
  P6      5         6           6            27         16          21
  P7      1         1           10           15         4           5
Average waiting time = 9.71429
Average turnaround time = 13.5714
```

PREEMPTIVE PRIORITY SCHEDULING(C++)

```
#include<iostream>
#include<algorithm>
using namespace std;
        struct node{
        char pname;
        int btime;
        int atime;
        int priority;
        int restime=0;
        int ctime=0;
        int wtime=0;
        }a[1000],b[1000],c[1000];
        void insert(int n){
        int i;
        for(i=0;i<n;i++){
        cin>>a[i].pname;
        cin>>a[i].priority;
        cin>>a[i].atime;
        cin>>a[i].btime;
        a[i].wtime=-a[i].atime+1;
        }
        }
        bool btimeSort(node a,node b){
        return a.btime < b.btime;
        }
        bool atimeSort(node a,node b){
        return a.atime < b.atime;
        }
        bool prioritySort(node a,node b){
        return a.priority < b.priority;
```

Course Code		Program	B.Tech.
Semester	V	Year	III
Name of the Faculty	Prof. Mohini Reddy	Class	
Course Title	Operating Systems	Academic year	2022-23

```
}
int k=0,f=0,r=0;
void disp(int nop,int qt){
int n=nop,q;
sort(a,a+n,atimeSort);
int ttime=0,i;
int j,tArray[n];
int alltime=0;
bool moveLast=false;
for(i=0;i<n;i++){
alltime+=a[i].btime;
}
alltime+=a[0].atime;
for(i=0;ttime<=alltime;){
j=i;
while(a[j].atime<=ttime&&j!=n){
b[r]=a[j];
j++;
r++;
}
if(r==f){
c[k].pname='i';
c[k].btime=a[j].atime-ttime;
c[k].atime=ttime;
ttime+=c[k].btime;
k++;
continue;
}
i=j;
if(moveLast==true){
sort(b+f,b+r,prioritySort);
// b[r]=b[f];
// f++;
// r++;
}
j=f;
if(b[j].btime>qt){
c[k]=b[j];
c[k].btime=qt;
k++;
b[j].btime=b[j].btime-qt;
ttime+=qt;
moveLast=true;
```

Course Code		Program	B.Tech.
Semester	V	Year	III
Name of the Faculty	Prof. Mohini Reddy	Class	
Course Title	Operating Systems	Academic year	2022-23

```
for(q=0;q<n;q++){
if(b[j].pname!=a[q].pname){
a[q].wtime+=qt;
}
}
}
else{
c[k]=b[j];
k++;
f++;
ttime+=b[j].btime;
moveLast=false;
for(q=0;q<n;q++){
if(b[j].pname!=a[q].pname){
a[q].wtime+=b[j].btime;
}
}
}
if(f==r&&i>=n)
break;
}
tArray[i]=ttime;
ttime+=a[i].btime;
for(i=0;i<k-1;i++){
if(c[i].pname==c[i+1].pname){
c[i].btime+=c[i+1].btime;
for(j=i+1;j<k-1;j++)
c[j]=c[j+1];
k--;
i--;
}
}

int rtime=0;
for(j=0;j<n;j++){
rtime=0;
for(i=0;i<k;i++){
if(c[i].pname==a[j].pname){
a[j].restime=rtime;
break;
}
rtime+=c[i].btime;
}
```

```
}
float averageWaitingTime=0;
float averageResponseTime=0;
float averageTAT=0;

cout<<"\nGantt Chart\n";
rtime=0;
for (i=0; i<k; i++){
if(i!=k)
cout<<"|  "<<'P'<< c[i].pname << "    ";
rtime+=c[i].btime;
for(j=0;j<n;j++){
if(a[j].pname==c[i].pname)
a[j].ctime=rtime;
}
}
cout<<"\n";
rtime=0;
for (i=0; i<k+1; i++){
cout << rtime << "\t";
tArray[i]=rtime;
rtime+=c[i].btime;
}
cout<<"\n";
cout<<"\n";
cout<<"P.Name Priority AT\tBT\tCT\tTAT\tWT\tRT\n";
for (i=0; i<nop&&a[i].pname!='i'; i++){
if(a[i].pname=='\0')
break;
cout <<'P'<< a[i].pname << "\t";
cout << a[i].priority << "\t";
cout << a[i].atime << "\t";
cout << a[i].btime << "\t";
cout << a[i].ctime << "\t";
cout << a[i].wtime+a[i].ctime-rtime+a[i].btime << "\t";
averageTAT+=a[i].wtime+a[i].ctime-rtime+a[i].btime;
cout << a[i].wtime+a[i].ctime-rtime << "\t";
averageWaitingTime+=a[i].wtime+a[i].ctime-rtime;
cout << a[i].restime-a[i].atime << "\t";
averageResponseTime+=a[i].restime-a[i].atime;
cout <<"\n";
}
cout<<"Average Waiting time:
```

Course Code		Program	B.Tech.
Semester	V	Year	III
Name of the Faculty	Prof. Mohini Reddy	Class	
Course Title	Operating Systems	Academic year	2022-23

```
"<<(float)averageWaitingTime/(float)n<<endl;
        cout<<"Average TA time:
"<<(float)averageTAT/(float)n<<endl;
        }
        int main(){
        int nop,choice,i,qt;
        cout<<"Enter number of processes\n";
        cin>>nop;
        cout<<"Enter process, priority, AT, BT\n";
        insert(nop);
        disp(nop,1);
        return 0;
        }
```

```
Enter number of processes
7
Enter process, priority, AT, BT
1 3 0 8
2 4 1 2
3 4 3 4
4 5 4 1
5 2 5 6
6 6 6 5
7 1 10 1

Gantt Chart
|  P1    |  P5    |  P7    |  P5    |  P1    |  P2    |  P3    |  P4    |  P6
0        5        10       11       12       15       17       21       22       27

P.Name Priority AT       BT       CT       TAT      WT       RT
P1       3        0        8        15       15       7        0
P2       4        1        2        17       16       14       14
P3       4        3        4        21       18       14       14
P4       5        4        1        22       18       17       17
P5       2        5        6        12       7        1        0
P6       6        6        5        27       21       16       16
P7       1        10       1        11       1        0        0
Average Waiting time: 9.85714
Average TA time: 13.7143
```

Course Code		Program	B.Tech.
Semester	V	Year	III
Name of the Faculty	Prof. Mohini Reddy	Class	
Course Title	Operating Systems	Academic year	2022-23

B.2 Conclusion:

Successfully learnt and implemented Priority scheduling algorithm.

B.3 Questions of Curiosity:

Q1. What is the Average Waiting time and Turnaround time for the implemented Non-Preemptive Priority Scheduling example?

```
Average waiting time = 9.71429
Average turnaround time = 13.5714
```

Q2. What is the Average Waiting time and Turnaround time for the implemented Preemptive Priority Scheduling example?

```
Average Waiting time: 9.85714
Average TA time: 13.7143
```

Q3. Which algorithm out of above two is more advantageous? Justify your answer.

PRIORITY PREEMPTIVE SCHEDULING	PRIORITY NON PREEMPTIVE SCHEDULING
If a process with higher priority than the process currently being executed arrives, the CPU is preemeted and given to the higher priority process.	Once resources are allocated to a process, the process holds it till it completes its burst time even if a process with higher priority is added to the queue.
Preemptive scheduling is more flexible.	Non-preemptive scheduling is rigid.
The waiting time for the process having the highest priority will always be zero.	The waiting time for the process having the highest priority may not be zero.
It is more expensive and difficult to implement. Also a lot of time is wasted in switching.	It is cheaper to implement and faster as less switching is required.
It is useful in applications where high priority processes cannot be kept waiting.	It can be used in various hardware applications where waiting will not cause any serious issues.