ATMA RAM SANATAN DHARMA COLLEGE
University of Delhi

Operating Systems

Practical File

Submitted By

Aditi sharma

College Roll N0. 21/18055

BSc(Hons) Computer Science

Submitted to

Dr. Parul Jain

Department of Computer Science

Q 9) Write a program to implement non-preemptive priority-based scheduling algorithm.

Code –

```
#include <stdio.h>
#include <stdlib.h>
struct process
{
 int pid;
 int priority;
 int burstTime;
 int arrivalTime;
 int waitingTime;
 int turnAroundTime;
};
int comparisonDesc(const void *a, const void *b)
{
 return ((struct process *)a)->priority < ((struct process *)b)->priority;
}
int comparisonAsc(const void *a, const void *b)
{
 return ((struct process *)a)->pid > ((struct process *)b)->pid;
}
void computeWaitingTime(struct process *processes, int processCount)
{
 qsort(processes, processCount, sizeof(struct process), comparisonDesc);
 processes[0].waitingTime = 0;
 for (int i = 0; i < processCount - 1; i++)
 processes[i + 1].waitingTime =
 processes[i].burstTime +
 processes[i].waitingTime;
}
void computeTurnAroundTime(struct process *processes, int processCount)
{
for (int i = 0; i < processCount; i++)
 processes[i].turnAroundTime =
 processes[i].burstTime +
 processes[i].waitingTime;
 qsort(processes, processCount, sizeof(struct process), comparisonAsc);
}
void printAverageTimes(struct process *processes, int processCount)
{
 float totalWaitingTime = 0.0f;
 float totalTurnAroundTime = 0.0f;
 computeWaitingTime(processes, processCount);
 computeTurnAroundTime(processes, processCount);
 printf("Process ID\tPriority\tBurst Time\tArrival Time\tWaiting
Time\tTurn Around Time\n");
 printf("-----------------------------------------------------------");
```

```
 printf("----------------------------------------\n");
 for (int i = 0; i < processCount; i++)
 {
 totalWaitingTime += processes[i].waitingTime;
 totalTurnAroundTime += processes[i].turnAroundTime;
 printf("%d\t\t%d\t\t%d\t\t%d\t\t%d\t\t%d\n",
 processes[i].pid,
 processes[i].priority,
 processes[i].burstTime,
 processes[i].arrivalTime,
 processes[i].waitingTime,
 processes[i].turnAroundTime);
 }
 printf("\nAverage Waiting Time = %.2f",
 totalWaitingTime / processCount);
 printf("\nAverage Turn-Around time = %.2f\n",
 totalTurnAroundTime / processCount);
}
int main(void)
{
 int processCount;
 printf("Enter Number of Processes: ");
 scanf("%i", &processCount);
 struct process processes[processCount];
 for (int i = 0; i < processCount; i++)
 {
 processes[i].pid = i + 1;
 printf("Burst Time for Process %i: ", i + 1);
scanf("%d", &processes[i].burstTime);
 printf("Arrival Time for Process %i: ", i + 1);
 scanf("%d", &processes[i].arrivalTime);
 printf("Priority for Process %i: ", i + 1);
 scanf("%d", &processes[i].priority);
 }
 printf("\n");
 printAverageTimes(processes, processCount);
 return 0;
}
```

Output :-

```
Enter Number of Processes: 5
Burst Time for Process 1: 3
Arrival Time for Process 1: 0
Priority for Process 1: 3
Burst Time for Process 2: 5
Arrival Time for Process 2: 0
Priority for Process 2: 4
Burst Time for Process 3: 1
Arrival Time for Process 3: 0
Priority for Process 3: 1
Burst Time for Process 4: 7
Arrival Time for Process 4: 0
Priority for Process 4: 7
Burst Time for Process 5: 4
Arrival Time for Process 5: 0
Priority for Process 5: 8
6               19
2               4               5               0       11              16
3               1               1               0       19              20
4               7               7               0       11
5               8               4               0       4
Average Waiting Time = 10.00
Average Turn-Around time = 14.00
```

Q10) Write a program to implement pre-emptive priority based scheduling algorithm.

Code –

```
 #include<iostream>
#include<stdio.h>
using namespace std;
int main()
{
int i,j,n,time,sum_wait=0,sum_turnaround=0,smallest;
int at[10],bt[10],pt[10],rt[10],remain;
cout<<"\nEnter number of Processes: ";
cin>>n;
remain=n;
cout<<"\nEnter arrival time, burst time and priority for:\n ";
for(i=0;i<n;i++)
{
cout<<"\nProcess "<<i+1<<": ";
cin>>at[i];
cin>>bt[i];
cin>>pt[i];
rt[i]=bt[i];
}
pt[9]=11;
cout<<"\n\nProcess\t|Turnaround time|waiting time\n";
for(time=0;remain!=0;time++)
{
smallest=9;
for(i=0;i<n;i++)
{
if(at[i]<=time && pt[i]<pt[smallest] && rt[i]>0)
{
smallest=i;
}
}
rt[smallest]--;
if(rt[smallest]==0)
{
remain--;
cout<<" P:"<<smallest+1<<"\t|\t "<<time+1-at[smallest]<<"\t\t|\t"<<time+1-
at[smallest]-
bt[smallest]<<"\n";
sum_wait+=time+1-at[smallest];
sum_turnaround+=time+1-at[smallest]-bt[smallest];
}
}
cout<<"\nAverage Waiting Time: "<<sum_wait/n;
cout<<"\nAverage Turn Around Time: "<<sum_turnaround/n<<endl;
return 0;
```

```
}
```

Output :-

```
Enter number of Processes: 4

Enter arrival time, burst time and priority for:

Process 3: 2 4 2

Process 4: 3 6 4


Process |Turnaround time|waiting time
 P:1    |        5             |       0
 P:3    |        7             |       3
 P:2    |        12            |       8
 P:4    |        16            |       10

Average Waiting Time: 10
Average Turn Around Time: 5
```