Lab Manual – 6

CSC 2209 OPERATING SYSTEM

CPU Scheduling

EXPERIMENT NO.1

Simulate the following CPU scheduling algorithms

1. **FIRST COME FIRST SERVE (FCFS)**

AIM: To write a c program to simulate the CPU scheduling algorithm First Come First Serve (FCFS).

DESCRIPTION

To calculate the average waiting time using the FCFS algorithm first the waiting time of the first process is kept zero and the waiting time of the second process is the burst time of the first process and the waiting time of the third process is the sum of the burst times of the first and the second process and so on. After calculating all the waiting times the average waiting time is calculated as the average of all the waiting times. FCFS mainly says first come first serve the algorithm which came first will be served first.

ALGORITHM

Step 1: Start the process

Step 2: Accept the number of processes in the ready Queue

Step 3: For each process in the ready Q, assign the process name and the burst time

Step 4: Set the waiting of the first process as 0‘and its burst time as its turnaround time

Step 5: for each process in the Ready Q calculate

a). Waiting time (n) = waiting time (n-1) + Burst time (n-

1) b). Turnaround time (n)= waiting time(n)+Burst time(n)

Step 6: Calculate

a) Average waiting time = Total waiting Time / Number of process

b) Average Turnaround time = Total Turnaround Time / Number of process

Step 7: Stop the process

SOURCE CODE:

```
#include<stdio.h>
#include<conio.h>
void main()
{
char pn[10][10];
int arr[10],bur[10],star[10],finish[10],tat[10],wt[10],i,n;
int totwt=0,tottat=0;
clrscr();
printf("Enter the number of processes:");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("Enter the Process Name, Arrival Time & Burst Time:");
scanf("%s%d%d",&pn[i],&arr[i],&bur[i]);
}
for(i=0;i<n;i++)
{
if(i==0)
{
star[i]=arr[i];
wt[i]=star[i]-arr[i];
finish[i]=star[i]+bur[i];
```

```
tat[i]=finish[i]-arr[i];
}
else
{
star[i]=finish[i-ı];
wt[i]=star[i]-arr[i];
finish[i]=star[i]+bur[i];
tat[i]=finish[i]-arr[i];
}
}
printf("\nPName    Arrtime    Burtime    Start    TAT  Finish");
for(i=o;i<n;i++)
{
printf("\n%s\t%6d\t\t%6d\t%6d\t%6d\t%6d",pn[i],arr[i],bur[i],star[i],tat[i],finish[i]);
totwt+=wt[i];
tottat+=tat[i];
}
printf("\nAverage Waiting time:%f",(float)totwt/n);
printf("\nAverage Turn Around Time:%f",(float)tottat/n);
getch();
}
```

OUTPUT:

Input:

Enter the number of processes: 3

Enter the Process Name, Arrival Time & Burst Time: 1 2 3

Enter the Process Name, Arrival Time & Burst Time: 2 5 6

Enter the Process Name, Arrival Time & Burst Time: 3 6 7

Output:

PName	Arrtime	Burtime	Srart	TAT	Finish
1	2	3	2	3	5
2	5	6	5	6	4
3	6	7	6	7	10

Average Waiting Time: 3.333

Average Turn Around Time: 7.000

2. SHORTEST JOB FIRST(SJF):

AIM: To write a program to stimulate the CPU scheduling algorithm Shortest job first

(Non- Preemption)

DESCRIPTION:

To calculate the average waiting time in the shortest job first algorithm the sorting of the process based on their burst time in ascending order then calculate the waiting time of each process as the sum of the bursting times of all the process previous or before to that process.

ALGORITHM:

Step 1: Start the process
Step 2: Accept the number of processes in the ready Queue

Step 3: For each process in the ready Q, assign the process id and accept the CPU burst time

Step 4: Start the Ready Q according the shortest Burst time by sorting according to lowest to highest burst time.

Step 5: Set the waiting time of the first process as ‗0' and its turnaround time as its burst time.

Step 6: Sort the processes names based on their Burt time

Step 7: For each process in the ready queue, calculate

a) Waiting time(n)= waiting time (n-1) + Burst time (n-1)

b) Turnaround time (n)= waiting time(n)+Burst time(n)

Step 8: Calculate

c)Average waiting time = Total waiting Time / Number of process

d) Average Turnaround time = Total Turnaround Time / Number of

process Step 9: Stop the process

SOURCE CODE :

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
int et[20],at[10],n,i,j,temp,st[10],ft[10],wt[10],ta[10];
int totwt=0,totta=0;
float awt,ata;
char pn[10][10],t[10];
clrscr();
printf("Enter the number of process:");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("Enter process name, arrival time & execution time:");
flushall();
scanf("%s%d%d",pn[i],&at[i],&et[i]);
}
for(i=0;i<n;i++)
for(j=0;j<n;j++)
{
if(et[i]<et[j])
{
temp=at[i];
at[i]=at[j];
```

```
at[j]=temp;
temp=et[i];
et[i]=et[j];
et[j]=temp;
strcpy(t,pn[i]);
strcpy(pn[i],pn[j]);
strcpy(pn[j],t);
}
}
for(i=0;i<n;i++)
{
if(i==0)
st[i]=at[i];
else
st[i]=ft[i-1];
wt[i]=st[i]-at[i];
ft[i]=st[i]+et[i];
ta[i]=ft[i]-at[i];
totwt+=wt[i];
totta+=ta[i];
}
awt=(float)totwt/n;
ata=(float)totta/n;
printf("\nPname\tarrivaltime\texecutiontime\twaitingtime\ttatime");
for(i=0;i<n;i++)
printf("\n%s\t%5d\t\t%5d\t\t%5d\t\t%5d",pn[i],at[i],et[i],wt[i],ta[i]);
```

```
printf("\nAverage waiting time is:%f",awt);
printf("\nAverage turnaroundtime is:%f",ata);
getch();
}
```

OUTPUT:

Input:

Enter the number of processes: 3

Enter the Process Name, Arrival Time & Burst Time: 1 4 6

Enter the Process Name, Arrival Time & Burst Time: 2 5 15

Enter the Process Name, Arrival Time & Burst Time: 3 6 11

Output:

Pname	arrivaltime	executiontime	waitingtime	tatime
1	4	6	0	6
3	6	11	4	15
2	5	15	16	31

Average Waiting Time: 6.6667

Average Turn Around Time: 17.3333

3. Priority

AIM: To write a c program to simulate the CPU scheduling priorityalgorithm.

DESCRIPTION:

To calculate the average waiting time in the priority algorithm, sort the burst times according to their priorities and then calculate the average waiting time of the processes. The waiting time of each process is obtained by summing up the burst times of all the previous processes.

ALGORITHM:

Step 1: Start the process

Step 2: Accept the number of processes in the ready Queue

Step 3: For each process in the ready Q, assign the process id and accept the CPU burst time

Step 4: Sort the ready queue according to the priority number.

Step 5: Set the waiting of the first process as ‗0' and its burst time as its turnaround time

Step 6: Arrange the processes based on process priority

Step 7: For each process in the Ready Q calculate

Step 8: for each process in the Ready Q calculate

a) Waiting time(n)= waiting time (n-1) + Burst time (n-1)

b) Turnaround time (n)= waiting time(n)+Burst time(n)

Step 9: Calculate

c)Average waiting time = Total waiting Time / Number of process

d) Average Turnaround time = Total Turnaround Time / Number of process Print the results

in an order.

Step10: Stop

SOURCE CODE :

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
int et[20],at[10],n,i,j,temp,p[10],st[10],ft[10],wt[10],ta[10];
int totwt=0,totta=0;
float awt,ata;
char pn[10][10],t[10];
clrscr();
printf("Enter the number of process:");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("Enter process name,arrivaltime,execution time & priority:");
flushall();
scanf("%s%d%d%d",pn[i],&at[i],&et[i],&p[i]);
}
for(i=0;i<n;i++)
for(j=0;j<n;j++)
{
if(p[i]<p[j])
{
temp=p[i];
p[i]=p[j];
```

```
p[j]=temp;
temp=at[i];
at[i]=at[j];
at[j]=temp;
temp=et[i];
et[i]=et[j];
et[j]=temp;
strcpy(t,pn[i]);
strcpy(pn[i],pn[j]);
strcpy(pn[j],t);
}
}
for(i=o;i<n;i++)
{
if(i==o)
{
st[i]=at[i];
wt[i]=st[i]-at[i];
ft[i]=st[i]+et[i];
ta[i]=ft[i]-at[i];
}
else
{
st[i]=ft[i-ı];
wt[i]=st[i]-at[i];
ft[i]=st[i]+et[i];
```

```
ta[i]=ft[i]-at[i];

}

totwt+=wt[i];

totta+=ta[i];

}

awt=(float)totwt/n;

ata=(float)totta/n;

printf("\nPname\tarrivaltime\texecutiontime\tpriority\twaitingtime\ttatime");

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

printf("\n%s\t%5d\t\t%5d\t\t%5d\t\t%5d\t\t%5d",pn[i],at[i],et[i],p[i],wt[i],ta[i]);

printf("\nAverage waiting time is:%f",awt);

printf("\nAverage turnaroundtime is:%f",ata);

getch();

}
```

OUTPUT:

Input:

Enter the number of processes: 3
Enter the Process Name, Arrival Time, execution time & priority: 1 2 3 1
Enter the Process Name, Arrival Time, execution time & priority: 2 4 5 2
Enter the Process Name, Arrival Time, execution time & priority: 3 5 6 3

Output:

Pname	arrivaltime	executiontime	priority	waitingtime	tatime
1	2	3	1	0	3
2	4	5	2	1	6
3	5	6	3	5	11

Average Waiting Time: 2.0000
Average Turn Around Time: 6.6667

4. Round Robin:

AIM: To simulate the CPU scheduling algorithm round-robin.

DESCRIPTION:

To aim is to calculate the average waiting time. There will be a time slice, each process should be executed within that time-slice and if not it will go to the waiting state so first check whether the burst time is less than the time-slice. If it is less than it assign the waiting time to the sum of the total times. If it is greater than the burst-time then subtract the time slot from the actual burst time and increment it by time-slot and the loop continues until all the processes are completed.

ALGORITHM:

Step 1: Start the process

Step 2: Accept the number of processes in the ready Queue and time quantum (or) time slice

Step 3: For each process in the ready Q, assign the process id and accept the CPU burst time

Step 4: Calculate the no. of time slices for each process where No. of time slice for process (n) = burst time process (n)/time slice

Step 5: If the burst time is less than the time slice then the no. of time slices =1.

Step 6: Consider the ready queue is a circular Q, calculate

a) Waiting time for process (n) = waiting time of process(n-1)+ burst time of process(n-1) + the time difference in getting the CPU fromprocess(n-1)

b) Turnaround time for process(n) = waiting time of process(n) + burst time of process(n)+ the time difference in getting CPU from process(n).

Step 7: Calculate

c)Average waiting time = Total waiting Time / Number of process

d)Average Turnaround time = Total Turnaround Time / Number ofprocess

Step 8: Stop the process

SOURCE CODE :

```
#include<stdio.h>
#include<conio.h>
void main()
{
int et[30],ts,n,i,x=0,tot=0;
char pn[10][10];
clrscr();
printf("Enter the no of processes:");
scanf("%d",&n);
printf("Enter the time quantum:");
scanf("%d",&ts);
for(i=0;i<n;i++)
{
printf("enter process name & estimated time:");
scanf("%s %d",pn[i],&et[i]);
}
printf("The processes are:");
for(i=0;i<n;i++)
printf("process %d: %s\n",i+1,pn[i]);
for(i=0;i<n;i++)
tot=tot+et[i];
while(x!=tot)
{
for(i=0;i<n;i++)
{
```

```
if(et[i]>ts)
{
x=x+ts;
printf("\n %s -> %d",pn[i],ts);
et[i]=et[i]-ts;
}
else
if((et[i]<=ts)&&et[i]!=0)
{
x=x+et[i];
printf("\n %s -> %d",pn[i],et[i]);
et[i]=0;}
}
}
printf("\n Total Estimated Time:%d",x);
getch();
}
```

OUTPUT:

Input:

Enter the no of processes: 2

Enter the time quantum: 3

Enter the process name & estimated time: p1 12

Enter the process name & estimated time: p2 15

Output:

p1 -> 3

p2 -> 3

p1 -> 3

p2 -> 3

p1 -> 3

p2 -> 3

p1 -> 3

p2 -> 3

p2 -> 3

Total Estimated Time: 27