**OPERATING SYSTEMS**

**LAB MANUAL**

**OPERATING SYSTEMS LAB**

# EXPERIMENT

## OBJECTIVE

Write a C program to simulate the following non-preemptive CPU scheduling algorithms to find turnaround time and waiting time for the above problem.

* + 1. FCFS b) SJF c) Round Robin d) Priority

## DESCRIPTION

Assume all the processes arrive at the same time.

#### FCFS CPU SCHEDULING ALGORITHM

For FCFS scheduling algorithm, read the number of processes/jobs in the system, their CPU burst times. The scheduling is performed on the basis of arrival time of the processes irrespective of their other parameters. Each process will be executed according to its arrival time. Calculate the waiting time and turnaround time of each of the processes accordingly.

#### SJF CPU SCHEDULING ALGORITHM

For SJF scheduling algorithm, read the number of processes/jobs in the system, their CPU burst times. Arrange all the jobs in order with respect to their burst times. There may be two jobs in queue with the same execution time, and then FCFS approach is to be performed. Each process will be executed according to the length of its burst time. Then calculate the waiting time and turnaround time of each of the processes accordingly.

#### ROUND ROBIN CPU SCHEDULINGALGORITHM

For round robin scheduling algorithm, read the number of processes/jobs in the system, their CPU burst times, and the size of the time slice. Time slices are assigned to each process in equal portions and in circular order, handling all processes execution. This allows every process to get an equal chance. Calculate the waiting time and turnaround time of each of the processes accordingly.

#### PRIORITY CPU SCHEDULING ALGORITHM

For priority scheduling algorithm, read the number of processes/jobs in the system, their CPU burst times, and the priorities. Arrange all the jobs in order with respect to their priorities. There may be two jobs in queue with the same priority, and then FCFS approach is to be performed. Each process will be executed according to its priority. Calculate the waiting time and turnaround time of each of the processes accordingly.

## PROGRAM

* + 1. ***FCFS CPU SCHEDULING ALGORITHM*** #include<stdio.h> #include<conio.h>

main()

{

int bt[20], wt[20], tat[20], i, n; float wtavg, tatavg;

clrscr();

printf("\nEnter the number of processes -- "); scanf("%d", &n);

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

{

printf("\nEnter Burst Time for Process %d -- ", i); scanf("%d", &bt[i]);

}

wt[0] = wtavg = 0; tat[0] = tatavg = bt[0]; for(i=1;i<n;i++)

{

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

tat[i] = tat[i-1] +bt[i]; wtavg = wtavg + wt[i]; tatavg = tatavg + tat[i];

}

printf("\t PROCESS \tBURST TIME \t WAITING TIME\t TURNAROUND TIME\n");

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

printf("\n\t P%d \t\t %d \t\t %d \t\t %d", i, bt[i], wt[i], tat[i]); printf("\nAverage Waiting Time -- %f", wtavg/n);

printf("\nAverage Turnaround Time -- %f", tatavg/n); getch();

}

|  |  |  |  |
| --- | --- | --- | --- |
| ***INPUT*** |  |  |  |
| Enter the number of processes -- | | 3 |  |
| Enter Burst Time for Process 0 -- | | 24 |  |
| Enter Burst Time for Process 1 -- | | 3 |  |
| Enter Burst Time for Process 2 -- | | 3 |  |
| ***OUTPUT*** |  |  |  |
| PROCESS | BURST TIME | WAITING TIME | TURNAROUND TIME |
| P0 | 24 | 0 | 24 |
| P1 | 3 | 24 | 27 |
| P2 | 3 | 27 | 30 |

Average Waiting Time-- 17.000000

Average Turnaround Time -- 27.000000

* + 1. ***SJF CPU SCHEDULING ALGORITHM*** #include<stdio.h> #include<conio.h>

main()

{

int p[20], bt[20], wt[20], tat[20], i, k, n, temp; float wtavg, tatavg;

clrscr();

printf("\nEnter the number of processes -- "); scanf("%d", &n);

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

{

p[i]=i;

printf("Enter Burst Time for Process %d -- ", i); scanf("%d", &bt[i]);

}

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

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

if(bt[i]>bt[k])

{

temp=bt[i]; bt[i]=bt[k]; bt[k]=temp;

}

wt[0] = wtavg = 0;

temp=p[i]; p[i]=p[k]; p[k]=temp;

tat[0] = tatavg = bt[0]; for(i=1;i<n;i++)

{

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

tat[i] = tat[i-1] +bt[i]; wtavg = wtavg + wt[i]; tatavg = tatavg + tat[i];

}

printf("\n\t PROCESS \tBURST TIME \t WAITING TIME\t TURNAROUND TIME\n");

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

printf("\n\t P%d \t\t %d \t\t %d \t\t %d", p[i], bt[i], wt[i], tat[i]); printf("\nAverage Waiting Time -- %f", wtavg/n);

printf("\nAverage Turnaround Time -- %f", tatavg/n); getch();

}

|  |  |  |  |
| --- | --- | --- | --- |
| ***INPUT*** |  |  |  |
| Enter the number of processes -- | | 4 |  |
| Enter Burst Time for Process 0 -- | | 6 |  |
| Enter Burst Time for Process 1 -- | | 8 |  |
| Enter Burst Time for Process 2 -- | | 7 |  |
| Enter Burst Time for Process 3 -- | | 3 |  |
| ***OUTPUT*** |  |  |  |
| PROCESS | BURST TIME | WAITING TIME | TURNAROUND TIME |
| P3 | 3 | 0 | 3 |
| P0 | 6 | 3 | 9 |
| P2 | 7 | 9 | 16 |
| P1 | 8 | 16 | 24 |

Average Waiting Time -- 7.000000

Average Turnaround Time -- 13.000000

#### ROUND ROBIN CPU SCHEDULING ALGORITHM

#include<stdio.h> main()

{

int i,j,n,bu[10],wa[10],tat[10],t,ct[10],max; float awt=0,att=0,temp=0;

clrscr();

printf("Enter the no of processes -- "); scanf("%d",&n);

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

{

printf("\nEnter Burst Time for process %d -- ", i+1); scanf("%d",&bu[i]);

ct[i]=bu[i];

}

printf("\nEnter the size of time slice -- "); scanf("%d",&t);

max=bu[0]; for(i=1;i<n;i++)

if(max<bu[i])

max=bu[i]; for(j=0;j<(max/t)+1;j++)

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

if(bu[i]!=0)

if(bu[i]<=t)

{

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

{

}

else

{

}

tat[i]=temp+bu[i]; temp=temp+bu[i]; bu[i]=0;

bu[i]=bu[i]-t; temp=temp+t;

wa[i]=tat[i]-ct[i]; att+=tat[i];

awt+=wa[i];

}

printf("\nThe Average Turnaround time is -- %f",att/n); printf("\nThe Average Waiting time is -- %f ",awt/n);

printf("\n\tPROCESS\t BURST TIME \t WAITING TIME\tTURNAROUND TIME\n");

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

printf("\t%d \t %d \t\t %d \t\t %d \n",i+1,ct[i],wa[i],tat[i]);

getch();

}

#### INPUT

Enter the no of processes – 3

Enter Burst Time for process 1 – 24 Enter Burst Time for process 2 -- 3 Enter Burst Time for process 3 -- 3

Enter the size of time slice – 3

#### OUTPUT

The Average Turnaround time is – 15.666667 The Average Waiting time is -- 5.666667

|  |  |  |  |
| --- | --- | --- | --- |
| PROCESS | BURST TIME | WAITING TIME | TURNAROUND TIME |
| 1 | 24 | 6 | 30 |
| 2 | 3 | 4 | 7 |
| 3 | 3 | 7 | 10 |

#### PRIORITY CPU SCHEDULING ALGORITHM

#include<stdio.h> main()

{

int p[20],bt[20],pri[20], wt[20],tat[20],i, k, n, temp; float wtavg, tatavg;

clrscr();

printf("Enter the number of processes --- "); scanf("%d",&n);

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

{

p[i] = i;

printf("Enter the Burst Time & Priority of Process %d --- ",i); scanf("%d %d",&bt[i], &pri[i]);

}

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

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

if(pri[i] > pri[k])

{

temp=p[i]; p[i]=p[k]; p[k]=temp;

temp=bt[i]; bt[i]=bt[k]; bt[k]=temp;

temp=pri[i]; pri[i]=pri[k]; pri[k]=temp;

}

wtavg = wt[0] = 0;

tatavg = tat[0] = bt[0];

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

{

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

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

wtavg = wtavg + wt[i]; tatavg = tatavg + tat[i];

}

printf("\nPROCESS\t\tPRIORITY\tBURST TIME\tWAITING TIME\tTURNAROUND TIME");

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

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

printf("\nAverage Waiting Time is --- %f",wtavg/n); printf("\nAverage Turnaround Time is --- %f",tatavg/n); getch();

}

|  |  |
| --- | --- |
| ***INPUT*** |  |
| Enter the number of processes -- 5  Enter the Burst Time & Priority of Process 0 --- 10 | 3 |
| Enter the Burst Time & Priority of Process 1 --- 1 | 1 |
| Enter the Burst Time & Priority of Process 2 --- 2 | 4 |
| Enter the Burst Time & Priority of Process 3 --- 1 | 5 |
| Enter the Burst Time & Priority of Process 4 --- 5 | 2 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***OUTPUT***  PROCESS | PRIORITY | BURST TIME | WAITING TIME | TURNAROUND TIME |
| 1 | 1 | 1 | 0 | 1 |
| 4 | 2 | 5 | 1 | 6 |
| 0 | 3 | 10 | 6 | 16 |
| 2 | 4 | 2 | 16 | 18 |
| 3 | 5 | 1 | 18 | 19 |

Average Waiting Time is --- 8.200000 Average Turnaround Time is --- 12.000000

# EXPERIMENT

## OBJECTIVE

**\***Write a C program to simulate multi-level queue scheduling algorithm considering the following scenario. All the processes in the system are divided into two categories – system processes and user processes. System processes are to be given higher priority than user processes. The priority of each process ranges from 1 to 3. Use fixed priority scheduling for all the processes.

## DESCRIPTION

Multi-level queue scheduling algorithm is used in scenarios where the processes can be classified into groups based on property like process type, CPU time, IO access, memory size, etc. In a multi-level queue scheduling algorithm, there will be 'n' number of queues, where 'n' is the number of groups the processes are classified into. Each queue will be assigned a priority and will have its own scheduling algorithm like round-robin scheduling or FCFS. For the process in a queue to execute, all the queues of priority higher than it should be empty, meaning the process in those high priority queues should have completed its execution. In this scheduling algorithm, once assigned to a queue, the process will not move to any other queues.

## PROGRAM