**OPERATING SYSTEM PROJECT**

Lovely Professional University

Phagwara, Punjab.

**PROBLEM ASSIGNED:**

 Consider a scheduling approach which is non pre-emptive similar to shortest job next in nature. The priority of each job is dependent on its estimated run time, and also the amount of time it has spent waiting. Jobs gain higher priority the longer they wait, which prevents indefinite postponement. The jobs that have spent a long time waiting compete against those estimated to have short run times. The priority can be computed as :

Priority = 1+ Waiting time / Estimated run time

Using the data given below compute the waiting time and turnaround time for each process and average waiting time and average turnaround time.

|  |  |  |
| --- | --- | --- |
| Process | Arrival time | Burst time |
| P1 | 0 | 20 |
| P2 | 5 | 36 |
| P3 | 13 | 19 |
| P4 | 17 | 42 |

**ABOUT THE PROBLEM IN TERMS OF OS**

The problem is based on CPU scheduling algorithms. These algorithms are used by the processors. There are some basic concepts which are used for the execution of these algorithms which are:

* Process execution consists of a cycle of CPU execution and I/O wait

• CPU burst and I/O burst alternate

• CPU burst distribution varies greatly from process to process, and from computer to computer, but follows similar curves

• Maximum CPU utilization obtained with multiprogramming

• CPU scheduler selects another process when current one is in I/O burst

Here is the term used cpu scheduler. Let us dicuss what is cpu scheduler

CPU scheduler selects from among the processes in ready queue, and allocates the CPU to one of them . CPU scheduling decisions may take place when a process:

• switches from running to waiting state (e.g., wait for I/O)

• switches from running to ready state (e.g., when an interrupt occurs)

• switches from waiting to ready (e.g., at completion of I/O)

• terminates

In the above problem, we have given with 4 processors are we have to use scheduling algorithm (priority scheduling) for the required solution. Here it is clearly given that non preemptive scheduling approach will be used .as we know that in non preemptive scheduling approach once the CPU has been allocated to a process, the process keeps it until terminates or waiting for I/O and it is also called cooperative scheduling.

**Algorithm for the above problem:**

• Priority scheduling is used for the problem. In this algorithm , it selects the ready process with highest priority

• a priority number is associated with each process, smaller integer, higher priority

• the CPU is allocated to the process with the highest priority

• SJF is special case of priority scheduling

• priority is the inverse of predicted next CPU burst time

• Priority scheduling can be preemptive or non preemptive but here as it is already mentioned non preemptive is used

C0DE AND SOLUTION:

|  |  |  |
| --- | --- | --- |
| Process | Arrival time | Burst time |
| P1 | 0 | 20 |
| P2 | 5 | 36 |
| P3 | 13 | 19 |
| P4 | 17 | 42 |

First calculate the waiting time for the processes and then calculate the priority by using the formula given in the question-

Gantt chart:

|  |  |  |  |
| --- | --- | --- | --- |
| P1 | P2 | P3 | P4 |

0 20 56 75 117

Now, we calculate the waiting time:

Formula= initial – arrival time(for every process)

For p1=0

P2=15

P3=43

P4=58

Calculate priority of each process by the formula given that is,

Priority = 1+ Waiting time / Estimated run time

P1 = 1+0/20= 1

P2= 0.3

P3= 0.6

P4= 0.5

Multiply it by 10

Therefore,

Priority for

P1=10

P2= 3

P3= 6

P4= 5

|  |  |  |  |
| --- | --- | --- | --- |
| PROCESS | A.T | B.T | PRIORITY |
| P1 | 0 | 20 | 10 |
| P2 | 5 | 36 | 3 |
| P3 | 13 | 19 | 6 |
| P4 | 17 | 42 | 5 |

Code used:

#include<stdio.h>

int main()

{ int i,n,p[10]={1,2,3,4,5,6,7,8,9,10},min,k=1,burst=0,pri[10];

int bt[10],temp,temp1,j,at[10],wt[10],rt[10],tt[10],ta=0,sum=0;

float wavg,tavg,tsum,wsum;

printf("\nenter the No. processes ");

scanf("%d",&n);

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

printf("\tenter the burst time of %d process",i+1);

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

printf("\tEnter the arrival time of %d process ",i+1);

scanf("%d",&at[i]);

printf("\tEnter the priority time of %d process",i+1);

scanf("%d",&pri[i]);

}

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

{

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

{

if(at[i]<at[j])/\*sorting acc to arrival time\*/

{

temp=p[j];

p[j]=p[i];

p[i]=temp;

temp=at[j];

at[j]=at[i];

at[i]=temp;

temp1=bt[j];

bt[j]=bt[i];

bt[i]=temp1;

}

}

}

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

{ burst=burst+bt[j];

min=bt[k];

for(i=k;i<n;i++)/\*main logic\*/

{ min=pri[k];

if (burst>=at[i])

{

if(pri[i]<min)

{

temp=p[k];

p[k]=p[i];

p[i]=temp;

temp=at[k];

at[k]=at[i];

at[i]=temp;

temp1=bt[k];

bt[k]=bt[i];

bt[i]=temp1;

temp=pri[k];

pri[k]=pri[i];

pri[i]=temp;

}

}

}

k++;

}

wt[0]=0;

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

{

sum=sum+bt[i-1];

wt[i]=sum-at[i];

}

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

{

wsum=wsum+wt[i];

}

wavg=wsum/n;

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

{

ta=ta+bt[i];

tt[i]=ta-at[i];

}

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

{

tsum=tsum+tt[i];

}

tavg=tsum/n;

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

{

rt[i]=wt[i];

}

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

printf("\n RESULT:-\t\t\t VARIOUS TIMES");

printf("\nprocess\t burst\t arrival\tpriority " );

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

{

printf("\n p%d",p[i]);

printf("\t %d",bt[i]);

printf("\t %d",at[i]);

printf("\t\t %d",pri[i]);

}

printf("\nwaiting time\tturnaround time\tresponce time");

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

{

printf("\n %d",wt[i]);

printf("\t\t %d",tt[i]);

printf("\t\t%d",rt[i]);

}

printf("\nAVERAGE WAITING TIME:- %f ms",wavg);

printf("\nAVERAGE TURN AROUND TIME:- %f ms" ,tavg);

printf("\nAVERAGE RESPONSE TIME:- %f ms\n",wavg);

}

EXECUTION:



