**LOVELY PROFESSIONAL UNIVERSITY**

**Student Name: Vivek Mihsra**

**Student Id: 11813660**

**Email Address****:vivekmishragta@gmail.com**

**Section: K18XC**

**Roll no 44**

**GitHub Link:**[**https://github.com/gizmo-guardain/Operating-system-project-cse-316**](https://github.com/gizmo-guardain/Operating-system-project-cse-316)

Problem

Considering the arrival time and burst time requirement of the process the scheduler schedules the processes by interrupting the processor after every 6 units of time and does consider the completion of the process in this iteration. The scheduler than checks for the number of process waiting for the processor and allots the processor to the process but interrupting the processor every 10 unit of time and considers the completion of the processes in this iteration. The scheduler checks the number of processes waiting in the queue for the processor after the second iteration and gives the processor to the process which needs more time to complete than the other processes to go in the terminated state.

#include<stdio.h>

void rr(int no,int remt[10],int Cur\_t,int arT[10], int bsT[10]);

int main()

{

int processNo,currentTime,remainingProcess,indicator=0,wait=0,turnAroundTime=0,arrivalTime[10],burstTime[10],remainingTime[10],x=1;

printf("Number of processes ");

int n;

scanf("%d",&n);

remainingProcess = n;

printf("\nEnter the arrival time and burst time of the processes\n");

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

{

printf("\nProcess P%d\n",processNo+1);

printf("Arrival time = ");

scanf("%d",&arrivalTime[processNo]);

printf("Burst time = ");

scanf("%d",&burstTime[processNo]);

remainingTime[processNo]=burstTime[processNo];

}

printf("Time quantum for first round is 6.\n");

int timeQuantum=6;

currentTime=0;

for(processNo=0;remainingProcess!=0;)

{

if(remainingTime[processNo]<=timeQuantum && remainingTime[processNo]>0)

{

currentTime+=remainingTime[processNo];

remainingTime[processNo]=0;

indicator=1;

}

else if(remainingTime[processNo]>0)

{

remainingTime[processNo]-=timeQuantum;

currentTime+=timeQuantum;

}

if(remainingTime[processNo]==0 && indicator==1)

{ printf("%d",processNo);

remainingProcess--;

printf("P %d",processNo+1);

printf("\t\t\t%d",currentTime-arrivalTime[processNo]);

printf("\t\t\t%d\n",currentTime-burstTime[processNo]-arrivalTime[processNo]);

wait+=currentTime-arrivalTime[processNo]-burstTime[processNo];

turnAroundTime+=currentTime-arrivalTime[processNo];

indicator=0;

}

if(processNo==n-1){

x++;

if(x==2){

processNo=0;

timeQuantum=6;

printf("Time quantum for second round is 10. \n");

}

else{

break;

}

}

else if(currentTime >= arrivalTime[processNo+1]){

processNo++;

}

else{

processNo=0;

}

}

rr(n,remainingTime,currentTime,arrivalTime,burstTime);

return 0;

}

void rr(int no,int remainingTime[10],int currenttime,int arrivalTime[10], int burstTime[10]){

float avg\_wait,avg\_tut;

int i,j,n=no,temp,btime[20],Proc\_no[20],w\_time[20],tut\_t[20],total=0,loc;

printf("Third round with more burst time.\n");

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

{

btime[i]=remainingTime[i];

w\_time[i]=currenttime-arrivalTime[i]-btime[i];

Proc\_no[i]=i+1;

}

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

{

loc=i;

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

{

if(btime[j]<btime[loc]){

loc=j;

}

}

temp=btime[i];

btime[i]=btime[loc];

btime[loc]=temp;

temp=Proc\_no[i];

Proc\_no[i]=Proc\_no[loc];

Proc\_no[loc]=temp;

}

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

{

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

w\_time[i]+=btime[j];

}

total+=w\_time[i];

}

avg\_wait=(float)total/n;

total=0;

printf("\nProcess\t\tBurst time\t\twaiting time\t\tTurnaround Time");

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

{

tut\_t[i]=btime[i]+w\_time[i];

total=total + tut\_t[i];

printf("\nP%d\t\t\t%d\t\t\t%d\t\t\t%d",Proc\_no[i],btime[i],w\_time[i],tut\_t[i]);

}

avg\_tut=(float)total/n;

printf("\n\nAverage waiting time = %f",avg\_wait);

printf("\n Average turnaround time = %f\n",avg\_tut);

}

Problem in terms of Operating System

This problem is an example of Round-Robin Scheduling where time quantum is changed on each iteration.

Algorithm Used By Me

Basically the algorithm used by me was increasing the time quantum of the processes as per the iteration. Otherwise the concept of round robin scheduler was used for the burst time and arrival time to get the result of Waiting Time and Turn Around Time

Description

The arrival time and burst time requirement of the process the scheduler schedules the processes by interrupting the processor after every 6 units of time and does consider the completion of the process in this iteration. The scheduler than checks for the number of process waiting for the processor and allots the processor to the process but interrupting the processor every 10 unit of time and considers the completion of the processes in this iteration. The scheduler checks the number of processes waiting in the queue for the processor after the second iteration and gives the processor to the process which needs more time to complete than the other processes to go in the terminated state.

Boundary Conditions

If unfavorable arrival time and burst time are given than the program halts at the moment

Test Case And Output(Waiting Time and TurnAround Time)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PID | Arrival Time | Burstime | Waiting Time | Turn Around Time |
| P1 | 0 | 20 | 58 | 60 |
| P2 | 5 | 36 | 39 | 46 |
| P3 | 13 | 19 | 49 | 67 |
| P4 | 26 | 42 | 31 | 61 |

Github Link <https://github.com/gizmo-guardain/Operating-system-project-cse-316>