Student Name: Rutvik Surani

Student ID Email Address: rutvik.surani2012@gmail.com

GitHub Link: https://github.com/rutvik2012/Operating-System.git

Code:

|  |
| --- |
| #include<stdio.h> |
|  | #include<conio.h> |
|  | int main() |
|  | { |
|  | printf("\*------Welcome to multilevel Queue-------\*\n"); |
|  | printf("1. Shortest Time Remaining First\n"); |
|  | printf("2. Round Robin\n"); |
|  | int xc=0; |
|  | printf("Enter your choice:\n"); |
|  | scanf("%d",&xc); |
|  | if(xc==1) |
|  | { |
|  | int a[10],b[10],x[10]; |
|  | int waiting[10],turnaround[10],completion[10]; |
|  | int i,j,smallest,count=0,time,n; |
|  | double avg=0,tt=0,end; |
|  | printf("\nEnter the number of Processes: "); |
|  | scanf("%d",&n); |
|  | for(i=0;i<n;i++) |
|  | { |
|  | printf("\nEnter arrival time of process %d : ",i+1); |
|  | scanf("%d",&a[i]); |
|  | printf("\nEnter burst time of process %d : ",i+1); |
|  | scanf("%d",&b[i]); |
|  | } |
|  | for(i=0;i<n;i++) |
|  | x[i]=b[i]; |
|  | b[9]=9999; |
|  | //printf("time => process number"); |
|  | for(time=0;count!=n;time++) |
|  | { |
|  | smallest=9; |
|  | for(i=0;i<n;i++) |
|  | { |
|  | if(a[i]<=time && b[i]<b[smallest] && b[i]>0 ) |
|  | smallest=i; |
|  | } |
|  | b[smallest]--; |
|  | //printf("\n%d => p%d",time+1,smallest); |
|  | if(b[smallest]==0) |
|  | { |
|  | count++; |
|  | end=time+1; |
|  | completion[smallest] = end; |
|  | waiting[smallest] = end - a[smallest] - x[smallest]; |
|  | turnaround[smallest] = end - a[smallest]; |
|  | // printf("\n %d %d %d",smallest,wt[smallest],ttp[smallest]); |
|  | } |
|  | } |
|  | printf("pid \t burst \t arrival \twaiting \tturnaround \tcompletion"); |
|  | for(i=0;i<n;i++) |
|  | { |
|  | printf("\n %d \t %d \t %d\t\t%d \t\t%d\t\t%d",i+1,x[i],a[i],waiting[i],turnaround[i],completion[i]); |
|  | avg = avg + waiting[i]; |
|  | tt = tt + turnaround[i]; |
|  | } |
|  | printf("\n %If %If",avg,tt); |
|  | printf("\n\nAverage waiting time = %lf\n",avg/n); |
|  | printf("Average Turnaround time = %lf",tt/n); |
|  | getch(); |
|  | } |
|  |  |
|  | else if(xc==2){ |
|  |  |
|  | int i, limit, total = 0, x, counter = 0, time\_quantum; |
|  | int wait\_time = 0, turnaround\_time = 0, arrival\_time[10], burst\_time[10], temp[10]; |
|  | float average\_wait\_time, average\_turnaround\_time; |
|  | printf("\nEnter Total Number of Processes:\t"); |
|  | scanf("%d", &limit); |
|  | x = limit; |
|  | for(i = 0; i < limit; i++) |
|  | { |
|  | printf("nEnter Details of Process[%d]\n", i + 1); |
|  |  |
|  | printf("Arrival Time:\t"); |
|  |  |
|  | scanf("%d", &arrival\_time[i]); |
|  |  |
|  | printf("Burst Time:\t"); |
|  |  |
|  | scanf("%d", &burst\_time[i]); |
|  |  |
|  | temp[i] = burst\_time[i]; |
|  | } |
|  |  |
|  | printf("nEnter Time Quantum:\t"); |
|  | scanf("%d", &time\_quantum); |
|  | printf("\nProcess ID\t\tBurst Time\t Turnaround Time\t Waiting Time\n"); |
|  | for(total = 0, i = 0; x != 0;) |
|  | { |
|  | if(temp[i] <= time\_quantum && temp[i] > 0) |
|  | { |
|  | total = total + temp[i]; |
|  | temp[i] = 0; |
|  | counter = 1; |
|  | } |
|  | else if(temp[i] > 0) |
|  | { |
|  | temp[i] = temp[i] - time\_quantum; |
|  | total = total + time\_quantum; |
|  | } |
|  | if(temp[i] == 0 && counter == 1) |
|  | { |
|  | x--; |
|  | printf("\nProcess[%d]\t\t%d\t\t %d\t\t\t %d", i + 1, burst\_time[i], total - arrival\_time[i], total - arrival\_time[i] - burst\_time[i]); |
|  | wait\_time = wait\_time + total - arrival\_time[i] - burst\_time[i]; |
|  | turnaround\_time = turnaround\_time + total - arrival\_time[i]; |
|  | counter = 0; |
|  | } |
|  | if(i == limit - 1) |
|  | { |
|  | i = 0; |
|  | } |
|  | else if(arrival\_time[i + 1] <= total) |
|  | { |
|  | i++; |
|  | } |
|  | else |
|  | { |
|  | i = 0; |
|  | } |
|  | } |
|  |  |
|  | average\_wait\_time = wait\_time \* 1.0 / limit; |
|  | average\_turnaround\_time = turnaround\_time \* 1.0 / limit; |
|  | printf("\n\nAverage Waiting Time:\t%f", average\_wait\_time); |
|  | printf("\nAvg Turnaround Time:\t%fn", average\_turnaround\_time); |
|  | return 0; |
|  | } |
|  | else{ |
|  | printf("wromg choice"); |
|  |  |
|  | } |
|  | } |

1.Explain the problem in terms of operating system concept? (Max 200 word) Description:

Multilevel Queue Scheduling divides the processes in groups based on the process type, CPU consumption, Memory size, input-output access, and many more. There will be ‘n’ no. of ques made for different groups of processes. Similarly, in the given problem there are two queues formed. 1st is Smallest Remaining Time First (SRTF) and the 2nd is Round Robin (RR). The SRTF scheduling queue has been given a higher priority as they run INTERACTIVE PROCESSES compared to RR scheduling queue which holds lower priority to run BATCH PROCESSES. In the RR scheduling queue the TIME QUANTUM is fixed as 2 units. The scheduler will take input from the users for 3 things:

* No. of processes
* Arrival time of each process
* Burst time of each process.

(The scheduler also asks for the time quantum which will always be take as 2 units)

As for the output the scheduler needs to display the calculated AVERAGE TURNAROUND TIME and AVERAGE WAITING TIME, these both calculations will be done using the following formulas:

* Turnaround time=Burst time + Waiting time
* Turnaround time=Completion time – Arrival time
* Waiting time= Turnaround time – Burst time
* Completion time= Arrival time + Turnaround time

2.Write the algorithm for proposed solution of the assigned problem. Algorithm

3.Calculate complexity of implemented algorithm. (Student must specify complexity of each line of code along with overall complexity) Description (purpose of use):

4.Explain all the constraints given in the problem. Attach the code snippet of the implemented constraint. Code snippet:

* In the designed module of multilevel queue, the first queue of pre-emptive shortest remaining processing time first has the higher priority than the second queue which is Round robin. The priority is giving to the queue based on how the scheduling algorithm handles the incoming process and looks at various factor of waiting time, turnaround time and many more. Similarly, in our scheduler SRTF is one of the most efficient scheduling algorithm, it has minimum turnaround time, waiting time, and maximum CPU utilization, throughput.

5.If you have implemented any additional algorithm to support the solution, explain the need and usage of the same. Description:

Before designing a multilevel scheduler, I have implemented SRTF scheduling algorithm and Round Robin scheduling algorithm differently. And for the final multilevel schedular I have inculcate both the queues in it.

6. Explain the boundary conditions of the implemented code. Description:

7.Explain all the test cases applied on the solution of assigned problem. Description:

8.Have you made minimum 5 revisions of solution on GitHub? GitHub Link:

Yes