NAME-PRAHARSH KUMAR

ID-11902509

ROLLNO-B-29

GITHUB LINK-<https://github.com/praharsh77/os-project/>

EMAIL-praharshcrazy4@gmail.com

4- QUESTION->

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 Write a program to implement such an algorithm.

SOLUTION CODE->

|  |
| --- |
| #include<stdio.h> |
|  | #include<conio.h> |
|  | int main() |
|  | { |
|  | printf("\t\t\t----------------------- Scheduling -----------------------\n\n\n\n"); |
|  | long int n,i=0,j=0; |
|  | printf("Enter Number of Processes : "); |
|  | scanf("%ld",&n ); |
|  | double priority[n],avg\_waiting,avg\_turnaround,burstTime[n],arrivalTime[n],waitingTime[n],turnaroundTime[n], process[n], temp, completionTime[n],min,sum=0,sum2=0,wait\_final, turnaround\_final, wait\_avg, turnaround\_avg; |
|  | for(i=0;i<n;i++) |
|  | { |
|  | printf("\nEnter Burst Time for Process [%d] : ", i+1 ); |
|  | scanf("%lf", &burstTime[i]); |
|  | printf("Enter Arrival Time for Process [%d] : ", i+1 ); |
|  | scanf("%lf", &arrivalTime[i] ); |
|  | process[i]=i+1; |
|  | } |
|  |  |
|  | printf("\n\n\t\t\t -------------- Entered Values are --------------\n\n"); |
|  | printf("\t\t\t---------------------------------------\n"); |
|  | printf("\t\t\t| Process | Arrival Time | Burst Time |\n"); |
|  | printf("\t\t\t---------------------------------------\n"); |
|  | for(i=0;i<n;i++) |
|  | { |
|  | printf("\t\t\t| P[%0.0lf] | %0.0lf | %0.0lf |\n",process[i],arrivalTime[i],burstTime[i]); |
|  | } |
|  | printf("\t\t\t---------------------------------------\n"); |
|  |  |
|  |  |
|  | printf("\n\n\t\t\t-------- Sorting Processes according to Arrivaltime --------\n"); |
|  |  |
|  | for(i=0;i<n;i++) |
|  | { |
|  | for(j=0;j<n;j++) |
|  | { |
|  | if(arrivalTime[i]<arrivalTime[j]) |
|  | { |
|  |  |
|  | temp = burstTime[j]; |
|  | burstTime[j] = burstTime[i]; |
|  | burstTime [i] = temp; |
|  |  |
|  | temp = process[j]; |
|  | process[j] = process[i]; |
|  | process[i] = temp; |
|  |  |
|  | temp = arrivalTime[j]; |
|  | arrivalTime[j] = arrivalTime[i]; |
|  | arrivalTime[i] = temp; |
|  |  |
|  | } |
|  | } |
|  | } |
|  | printf("\n\n\t\t\t -------------- Now Values are --------------\n\n"); |
|  | printf("\t\t\t---------------------------------------\n"); |
|  | printf("\t\t\t| Process | Arrival Time | Burst Time |\n"); |
|  | printf("\t\t\t---------------------------------------\n"); |
|  | for(i=0;i<n;i++) |
|  | { |
|  | printf("\t\t\t| P[%0.0lf] | %0.0lf | %0.0lf |\n",process[i],arrivalTime[i],burstTime[i]); |
|  | } |
|  | printf("\t\t\t---------------------------------------\n"); |
|  |  |
|  |  |
|  | /\*Arranging the table according to Burst time, |
|  | Execution time and Arrival Time |
|  | Arrival time <= Execution time |
|  | \*/ |
|  |  |
|  |  |
|  | long int k = 1; |
|  | double b\_time = 0; |
|  | for(j=0;j<n;j++) |
|  | { |
|  | b\_time = b\_time + burstTime[j]; |
|  | min = burstTime[k]; |
|  |  |
|  | for(i=k;i<n;i++) |
|  | { |
|  | if((b\_time >= arrivalTime[i])&&(burstTime[i]<min)) |
|  | { |
|  | temp = burstTime[k]; |
|  | burstTime[k] = burstTime[i]; |
|  | burstTime[i] = temp; |
|  |  |
|  | temp = arrivalTime[k]; |
|  | arrivalTime[k] = arrivalTime[i]; |
|  | arrivalTime[i] = temp; |
|  |  |
|  | temp = process[k]; |
|  | process[k] = process[i]; |
|  | process[i] = temp; |
|  | } |
|  | } |
|  | k++; |
|  | } |
|  | waitingTime[0] = 0; |
|  | for(i=1;i<n;i++) |
|  | { |
|  | sum += burstTime[i-1]; |
|  | waitingTime[i] = sum - arrivalTime[i]; |
|  | wait\_final += waitingTime[i]; |
|  | } |
|  | wait\_avg = wait\_final/n; |
|  | for(i=0;i<n;i++) |
|  | { |
|  | sum2 += burstTime[i]; |
|  | turnaroundTime[i] = sum2 - arrivalTime[i]; |
|  | turnaround\_final += turnaroundTime[i]; |
|  | } |
|  | turnaround\_avg=turnaround\_final/n; |
|  | printf("\n\n\t\t\t -------------- Now Values are --------------\n\n"); |
|  | printf("\t\t\t-----------------------------------------------------------------------------\n"); |
|  | printf("\t\t\t| Process | Arrival Time | Burst Time | Waiting Time | Turn Around Time |\n"); |
|  | printf("\t\t\t-----------------------------------------------------------------------------\n"); |
|  | for(i=0;i<n;i++) |
|  | { |
|  | printf("\t\t\t| P[%0.0lf] | %0.0lf | %0.0lf | %0.0lf | %0.0lf |\n",process[i],arrivalTime[i],burstTime[i],waitingTime[i],turnaroundTime[i]); |
|  | } |
|  | printf("\t\t\t-----------------------------------------------------------------------------\n"); |
|  |  |
|  |  |
|  |  |
|  | /\*Now we have to prioritize the processes according to the formulae |
|  | Priority = 1+ Waiting time / Estimated run time |
|  | \*/ |
|  |  |
|  |  |
|  | completionTime[0] = burstTime[0]; |
|  | for(i=1;i<n;i++) |
|  | { |
|  | completionTime[i] = completionTime[i-1] + burstTime[i]; |
|  | } |
|  |  |
|  | for(i=0;i<n;i++) |
|  | { |
|  | priority[i] = 1+waitingTime[i]/completionTime[i]; |
|  | printf("%lf\n",priority[i]); |
|  | } |
|  |  |
|  |  |
|  | printf("\n\n\t\t\t -------------- Final Values are --------------\n\n"); |
|  | printf("\t\t\t-----------------------------------------------------------------------------\n"); |
|  | printf("\t\t\t| Process | Arrival Time | Burst Time | Waiting Time | Turn Around Time |\n"); |
|  | printf("\t\t\t-----------------------------------------------------------------------------\n"); |
|  | printf("\t\t\t| P[%0.0lf] | %0.0lf | %0.0lf | %0.0lf | %0.0lf |\n",process[0],arrivalTime[0],burstTime[0],waitingTime[0],turnaroundTime[0]); |
|  | for(i=n-1;i>0;i--) |
|  | { |
|  | printf("\t\t\t| P[%0.0lf] | %0.0lf | %0.0lf | %0.0lf | %0.0lf |\n",process[i],arrivalTime[i],burstTime[i],waitingTime[i],turnaroundTime[i]); |
|  | } |
|  | printf("\t\t\t-----------------------------------------------------------------------------\n"); |
|  |  |
|  | printf("\n\n\n\t\t\tAverage Turn Around Time : %lf",turnaround\_avg); |
|  | printf("\n\t\t\tAverage Waiting Time : %lf\n\n",wait\_avg); |
|  |  |
|  | getch(); |
|  | return 0; |
|  | } |