Pre-emptive Shortest Job First Algorithm

#include<stdio.h>

#include <string.h>

struct process

{

int pid;

int arrival\_time;

int burst\_time;

int start\_time;

int completion\_time;

int turnaround\_time;

int waiting\_time;

};

int main()

{

int n;

struct process p[100];

float avg\_turnaround\_time;

float avg\_waiting\_time;

int total\_turnaround\_time = 0;

int total\_waiting\_time = 0;

int total\_response\_time = 0;

int burst\_remaining[100];

int is\_completed[100];

memset(is\_completed,0,sizeof(is\_completed));

printf("Enter total number of processes : ");

scanf("%d",&n);

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

{

printf("Enter arrival time of process %d : ",i+1);

scanf("%d", &p[i].arrival\_time);

printf("Enter burst time of process %d : ",i+1);

scanf("%d", &p[i].burst\_time);

p[i].pid = i+1;

burst\_remaining[i] = p[i].burst\_time;

printf("\n");

}

int current\_time = 0;

int completed = 0;

int prev = 0;

while(completed != n)

{

int idx = -1;

int mn = 10000000;

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

if(p[i].arrival\_time <= current\_time && is\_completed[i] == 0)

{

if(burst\_remaining[i] < mn) {

mn = burst\_remaining[i];

idx = i;

}

if(burst\_remaining[i] == mn) {

if(p[i].arrival\_time < p[idx].arrival\_time) {

mn = burst\_remaining[i];

idx = i;

}

}

}

}

if(idx != -1)

{

if(burst\_remaining[idx] == p[idx].burst\_time)

{

p[idx].start\_time = current\_time;

}

burst\_remaining[idx] -= 1;

current\_time++;

prev = current\_time;

if(burst\_remaining[idx] == 0)

{

p[idx].completion\_time = current\_time;

p[idx].turnaround\_time = p[idx].completion\_time - p[idx].arrival\_time;

p[idx].waiting\_time = p[idx].turnaround\_time - p[idx].burst\_time;

total\_turnaround\_time += p[idx].turnaround\_time;

total\_waiting\_time += p[idx].waiting\_time;

is\_completed[idx] = 1;

completed++;

}

}

else {

current\_time++;

}

}

avg\_turnaround\_time = (float) total\_turnaround\_time / n;

avg\_waiting\_time = (float) total\_waiting\_time / n;

printf("\n\n");

printf("\nProcessID ArrivalTime BurstTime Start Time Completion Time WaitingTime TurnAroundTime");

printf("\n\n");

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

{

printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d\t\t%d\t\t%d\n", p[i].pid , p[i].arrival\_time , p[i].burst\_time , p[i].start\_time ,p[i].completion\_time, p[i].waiting\_time,p[i].turnaround\_time );

}

printf("\nAverage Waiting Time = %0.2f ",avg\_waiting\_time);

printf("\nAverage Turnaround Time = %0.2f ",avg\_turnaround\_time);

return 0

}

OUTPUT

Windows PowerShell

Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\ATHARVA> cd "d:\main\" ; if ($?) { gcc premsjf.c -o premsjf } ; if ($?) { .\premsjf }

Enter total number of processes : 5

Enter arrival time of process 1 : 2

Enter burst time of process 1 : 6

Enter arrival time of process 2 : 5

Enter burst time of process 2 : 2

Enter arrival time of process 3 : 1

Enter burst time of process 3 : 8

Enter arrival time of process 4 : 0

Enter burst time of process 4 : 3

Enter arrival time of process 5 : 4

Enter burst time of process 5 : 4

ProcessID ArrivalTime BurstTime Start Time Completion Time WaitingTime TurnAroundTime

1 2 6 3 15 7 13

2 5 2 5 7 0 2

3 1 8 15 23 14 22

4 0 3 0 3 0 3

5 4 4 4 10 2 6

Average Waiting Time = 4.60

Average Turnaround Time = 9.20

PS D:\main>