Round Robin Algorithm

#include<stdio.h>

int n,tq,time = 0;

int queue[100]={0},front = 0, rear = 0;

void sort( );

void rr();

int select(int t);

void push(int q);

int pop();

struct data

{

int at,bt1,bt,wt,tat;

}t;

int main()

{

int i;

float avg1=0,avg2=0;

printf("enter the no of processes: ");

scanf("%d",&n);

struct data aa[n];

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

{

printf("enter the process p[%d]'s arrival time and Burst time: ",i+1);

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

aa[i].bt1=aa[i].bt;

}

printf("enter time\_quantum: ");

scanf("%d",&tq);

sort( aa );

rr(aa);

printf("\n ProcessID Arrivaltime Bursttime waitingTime TurnAroundTime");

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

{

printf("\n|\tp[%d]\t|\t%d\t|\t%d \t|\t%d\t|\t%d\t|",i+1,aa[i].at,aa[i].bt1,aa[i].wt,aa[i].tat);

avg1+=aa[i].wt;

avg2+=aa[i].tat;

}

printf("\n average waiting time:%f \n average turnarond time :%f",avg1/n,avg2/n);

return 0;

}

void sort( struct data a[])

{

int i,j;

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

{

for(j=0;j<n-i-1;j++)

{

if(a[j].at>a[j+1].at)

{

t=a[j];

a[j]=a[j+1];

a[j+1]=t;

}

}

}

}

void rr(struct data a[])

{

int i=1,j=0,x;

x=n;

while(x!=0)

{

j=select(j);

if( a[j].bt<=tq && a[j].bt>0)

{

time = time + a[j].bt;

a[j].bt = 0;

}

else if(a[j].bt > 0)

{

a[j].bt = a[j].bt - tq;

time = time + tq;

}

while( a[i].at<=time && i<n )

{

push(i);

i++;

}

if(a[j].bt == 0)

{

x--;

a[j].wt = time - a[j].at - a[j].bt1;

a[j].tat= a[j].wt+a[j].bt1;

}

else

{

push(j);

}

}

}

int select(int j)

{

if(time == 0)

{

return j;

}

else

{

j=pop();

return j;

}

}

int pop()

{

int x;

x = queue[front++];

return x;

}

void push(int q)

{

queue[rear++]=q;

}

OUTPUT

Windows PowerShell

Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\ATHARVA> cd "c:\Users\ATHARVA\Desktop\ Operating systems\" ; if ($?) { gcc Round\_Robin.c -o Round\_Robin } ; if ($?) { .\Round\_Robin }

enter the no of processes: 6

enter the process p[1]'s arrival time and Burst time: 0 5

enter the process p[2]'s arrival time and Burst time: 1 6

enter the process p[3]'s arrival time and Burst time: 2 3

enter the process p[4]'s arrival time and Burst time: 3 1

enter the process p[5]'s arrival time and Burst time: 4 5

enter the process p[6]'s arrival time and Burst time: 6 4

enter time\_quantum: 4

ProcessID Arrivaltime Bursttime waitingTime TurnAroundTime

| p[1] | 0 | 5 | 12 | 17 |

| p[2] | 1 | 6 | 16 | 22 |

| p[3] | 2 | 3 | 6 | 9 |

| p[4] | 3 | 1 | 8 | 9 |

| p[5] | 4 | 5 | 15 | 20 |

| p[6] | 6 | 4 | 11 | 15 |

average waiting time:11.333333

average turnarond time :15.333333