Project Report

**Process Scheduling Simulator**

# **Group Members:**

* Ovaiz Ali 18K-0137 (2A)
* Zaeem Ahmed 18K-0166 (2A)
* Muhammad Hasan 18K-0294 (2A)

# **Objective:**

This is our operating systems project which aims to provide complete implementation of the process scheduling algorithms studied in our operating system course.

# **Project Description:**

The algorithms we have used in this simulator are FCFS (First Come First Serve), SJF (Shortest Job First), Priority scheduling (pre-emptive and non-pre-emptive) and Round Robin. The Burst time, Arrival Time, Priority and Time Quanta of processes are given as input. And we will calculate Waiting Time, Completion Time, Turn Around Time and average Waiting Time and average Turn Around Time.

# **Code:**

// Authors: Ovaiz Ali, Zaeem Ahmed, M.Hasan

// Purpose: Operating System Project with an

// implementation of process scheduling

// algorithms in an operating system.

// Version: 5.0

// Date: 1 June 2020

#include<stdio.h>

#include<stdlib.h>

#include <string.h>

#define clear() printf("\033[H\033[J")

typedef struct process{

char name[5];

int bt;

int at;

int prt;

int wt,ta;

int flag;

int ct;

}processes;

int n;

void b\_sort(processes temp[])

{

processes t;

int i,j;

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

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

if(temp[j].at > temp[j+1].at){

t = temp[j];

temp[j] = temp[j+1];

temp[j+1] = t;

}

}

}

void random(processes P[]){

int i;

int mxBT=30,mnBT=3,mxAT=0,mnAT=0;

printf("\n Enter total no. of processes : ");

scanf("%d",&n);

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

printf("\n PROCESS [%d]",i+1);

printf(" Enter process name : ");

scanf("%s",&P[i].name);

printf(" Enter priority : ");

scanf("%d",&P[i].prt);

P[i].bt = (rand() % mxBT - mnBT + 1) + mnBT;

mxAT = P[i].bt;

if(i==0)

P[i].at = 0;

else

P[i].at = (rand() % mxAT - mnAT + 1) + mnAT;

}

printf("\n PROCESS\tB.T.\tA.T.\tPRIORITY");

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

printf("\n %s\t\t%d\t%d\t%d",P[i].name,P[i].bt,P[i].at,P[i].prt);

}

void accept(processes P[]){

int i;

printf("\n Enter total no. of processes : ");

scanf("%d",&n);

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

printf("\n PROCESS [%d]",i+1);

printf(" Enter process name : ");

scanf("%s",&P[i].name);

printf(" Enter burst time : ");

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

printf(" Enter arrival time : ");

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

printf(" Enter priority : ");

scanf("%d",&P[i].prt);

}

printf("\n PROCESS\tB.T.\tA.T.\tPRIORITY");

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

printf("\n %s\t\t%d\t%d\t%d",P[i].name,P[i].bt,P[i].at,P[i].prt);

}

// FCFS Algorithm

void FCFS(processes P[]){

processes temp[10];

int sumw=0,sumt=0;

int x = 0;

float avgwt=0.0,avgta=0.0;

int i,j;

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

temp[i]=P[i];

printf("\n\t\tFCFS SCHEDULING\n");

b\_sort(temp);

printf("\n\n PROCESS\tB.T\tA.T");

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

printf("\n %s\t\t%d\t%d",temp[i].name,temp[i].bt,temp[i].at);

sumw = temp[0].wt = 0;

temp[0].ct = temp[0].bt-1;

sumt = temp[0].ta = temp[0].ct-temp[0].at;

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

temp[i].ct = temp[i-1].ct+temp[i].bt;

temp[i].wt = temp[i].ct-temp[i].bt;

temp[i].ta = temp[i].ct-temp[i].at;

sumw+=temp[i].wt;

sumt+=temp[i].ta;

}

avgwt = (float)sumw/n;

avgta = (float)sumt/n;

printf("\n\n PROCESS\tB.T\tA.T\tW.T\tC.T\tT.A.T");

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

printf("\n %s\t\t%d\t%d\t%d\t%d\t%d",temp[i].name,temp[i].bt,temp[i].at,temp[i].wt,temp[i].ct,temp[i].ta);

printf("\n\n GANTT CHART\n ");

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

printf("\t%s\t",temp[i].name);

printf("\n ");

printf("0\t");

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

x+=temp[i-1].bt;

printf("\t%d\t",x);

}

printf("\n\n Average waiting time = %0.2f\n Average turn-around = %0.2f.",avgwt,avgta);

}

//SJF Non Pre-emptive

void SJF\_NP(processes P[]){

processes temp[10];

processes t;

int sumw=0,sumt=0;

int x = 0;

float avgwt=0.0,avgta=0.0;

int i,j;

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

temp[i]=P[i];

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

temp[i].at = 0;

printf("\n\t\tSHORTEST JOB FIRST - NP SCHEDULING\n");

b\_sort(temp);

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

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

if(temp[j].bt > temp[j+1].bt){

t = temp[j];

temp[j] = temp[j+1];

temp[j+1] = t;

}

}

printf("\n\n PROCESS\tB.T\tA.T");

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

printf("\n %s\t\t%d\t%d",temp[i].name,temp[i].bt,temp[i].at);

sumw = temp[0].wt = 0;

temp[0].ct = temp[0].bt-1;

sumt = temp[0].ta = temp[0].ct-temp[0].at;

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

temp[i].ct = temp[i-1].ct+temp[i].bt;

temp[i].wt = temp[i].ct-temp[i].bt;

temp[i].ta = temp[i].ct-temp[i].at;

sumw+=temp[i].wt;

sumt+=temp[i].ta;

}

avgwt = (float)sumw/n;

avgta = (float)sumt/n;

printf("\n\n PROCESS\tB.T\tA.T\tW.T\tC.T\tT.A.T");

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

printf("\n %s\t\t%d\t%d\t%d\t%d\t%d",temp[i].name,temp[i].bt,temp[i].at,temp[i].wt,temp[i].ct,temp[i].ta);

printf("\n\n GANTT CHART\n ");

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

printf("\t%s\t",temp[i].name);

printf("\n ");

printf("0\t");

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

x+=temp[i-1].bt;

printf("\t%d\t",x);

}

printf("\n\n Average waiting time = %0.2f\n Average turn-around = %0.2f.",avgwt,avgta);

}

//Priority Non Pre-emptive

void PRT\_NP(processes P[])

{

processes temp[10];

processes t;

int sumw=0,sumt=0;

float avgwt=0.0,avgta=0.0;

int i,j;

int x = 0;

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

temp[i]=P[i];

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

temp[i].at = 0;

printf("\n\t\tPRIORITY - NP SCHEDULING\n");

b\_sort(temp);

for(i=2;i<n;i++)

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

if(temp[j].prt > temp[j+1].prt){

t = temp[j];

temp[j] = temp[j+1];

temp[j+1] = t;

}

}

printf("\n\n PROC.\tB.T.\tA.T.");

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

printf("\n %s\t%d\t%d",temp[i].name,temp[i].bt,temp[i].at);

sumw = temp[0].wt = 0;

temp[0].ct = temp[0].bt + temp[0].wt - 1;

sumt = temp[0].ta = temp[0].ct - temp[0].at;

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

temp[i].wt = (temp[i-1].bt + temp[i-1].at + temp[i-1].wt) - temp[i].at;;

temp[i].ct = (temp[i].wt + temp[i].bt - 1);

temp[i].ta = (temp[i].ct - temp[i].at);

sumw+=temp[i].wt;

sumt+=temp[i].ta;

}

avgwt = (float)sumw/n;

avgta = (float)sumt/n;

printf("\n\n PROC.\tB.T.\tA.T.\tPriority\tW.T\tC.T\tT.A.T");

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

printf("\n %s\t%d\t%d\t%d\t\t%d\t%d\t%d",temp[i].name,temp[i].bt,temp[i].at,temp[i].prt,temp[i].wt,temp[i].ct,temp[i].ta);

printf("\n\n GANTT CHART\n ");

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

printf(" %s ",temp[i].name);

printf("\n ");

printf("0\t");

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

x+=temp[i-1].bt;

printf("%d ",x);

}

printf("\n\n Average waiting time = %0.2f\n Average turn-around = %0.2f.",avgwt,avgta);

}

//Round Robin Scheduling

void RR(processes P[])

{

int pflag=0,t,tcurr=0,k,i,Q=0;

int sumw=0,sumt=0;

float avgwt=0.0,avgta=0.0;

processes temp1[10],temp2[10];

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

temp1[i]=P[i];

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

temp1[i].at = 0;

b\_sort(temp1);

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

temp2[i]=temp1[i];

printf("\n\t\tROUND ROBIN SCHEDULING\n");

printf("\n Enter quantum time : ");

scanf("%d",&Q);

for(k=0;;k++){

if(k>n-1)

k=0;

if(temp1[k].bt>0)

printf(" %d %s",tcurr,temp1[k].name);

t=0;

while(t<Q && temp1[k].bt > 0){

t++;

tcurr++;

temp1[k].bt--;

}

if(temp1[k].bt <= 0 && temp1[k].flag != 1){

if(temp2[k].bt < Q){

temp1[k].wt = tcurr - temp2[k].bt - temp1[k].at-1;

}

else{

temp1[k].wt = tcurr - Q -1;

}

temp1[k].ta = tcurr - temp1[k].at -1;

temp1[k].ct = tcurr-1;

pflag++;

temp1[k].flag = 1;

sumw+=temp1[k].wt;

sumt+=temp1[k].ta;

}

if(pflag == n)

break;

}

printf(" %d\n",tcurr);

printf("\n\n PROCESS\tB.T\tA.T\tW.T\tC.T\tT.A.T");

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

printf("\n %s\t\t%d\t%d\t%d\t%d\t%d",temp1[i].name,temp2[i].bt,temp2[i].at,temp1[i].wt,temp1[i].ct,temp1[i].ta);

avgwt = (float)sumw/n;

avgta = (float)sumt/n;

printf("\n\n Average waiting time = %0.2f\n Average turn-around = %0.2f.",avgwt,avgta);

}

void PRT\_P(processes P[]){

int i,t\_total=0,tcurr,b[10],j,x,min\_pr;

int sumw=0,sumt=0;

float avgwt=0.0,avgta=0.0;

processes temp[10],t;

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

temp[i]=P[i];

t\_total+=P[i].bt;

}

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

temp[i].at = 0;

printf("\n\t\tPRIORITY - P SCHEDULING\n");

b\_sort(temp);

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

b[i] = temp[i].bt;

i=j=0;

printf("\n GANTT CHART\n\n %d %s",i,temp[i].name);

for(tcurr=0;tcurr<t\_total;tcurr++)

{

if(b[i] > 0 && temp[i].at <= tcurr)

b[i]--;

if(b[i] == 0 && temp[i].at <= tcurr)

temp[i].ct = tcurr;

if(i!=j)

printf(" %d %s",tcurr,temp[i].name);

if(b[i]<=0 && temp[i].flag != 1)

{

temp[i].flag = 1;

temp[i].wt = (tcurr+1) - temp[i].bt - temp[i].at;

temp[i].ta = temp[i].ct - temp[i].at;

sumw+=temp[i].wt;

sumt+=temp[i].ta;

}

j=i;

min\_pr = 999;

for(x=0;x<n;x++){

if(temp[x].at <= (tcurr+1) && temp[x].flag != 1){

if(min\_pr != temp[x].prt && min\_pr > temp[x].prt){

min\_pr = temp[x].prt;

i=x;

}

}

}

}

printf(" %d",tcurr);

printf("\n\n PROCESS\tB.T\tA.T\tPriority\tW.T\tC.T\tT.A.T");

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

printf("\n %s\t\t%d\t%d\t%d\t\t%d\t%d\t%d",temp[i].name,temp[i].bt,temp[i].at,temp[i].prt,temp[i].wt,temp[i].ct,temp[i].ta);

avgwt = (float)sumw/n;

avgta = (float)sumt/n;

printf("\n\n Average waiting time = %0.2f\n Average turn-around = %0.2f.",avgwt,avgta);

}

int main(){

processes P[10];

int ch,n;

do{

printf("\n\n SIMULATION OF CPU SCHEDULING ALGORITHMS\n");

printf("\n Options:");

printf("\n 0. Enter process data.");

printf("\n 1. Enter with random process data.");

printf("\n 2. FCFS");

printf("\n 3. SJF (Non Pre-emptive)");

printf("\n 4. Priority Scheduling (Pre-emptive)");

printf("\n 5. Priority Scheduling (Non Pre-emptive)");

printf("\n 6. Round Robin");

printf("\n 7. Clear Screen");

printf("\n 8. Exit\n Select : ");

scanf("%d",&ch);

switch(ch){

case 0:

accept(P);

break;

case 1:

random(P);

break;

case 2:

FCFS(P);

break;

case 3:

SJF\_NP(P);

break;

case 4:

PRT\_P(P);

break;

case 5:

PRT\_NP(P);

break;

case 6:

RR(P);

break;

case 7:

clear();

break;

case 8:exit(0);

}

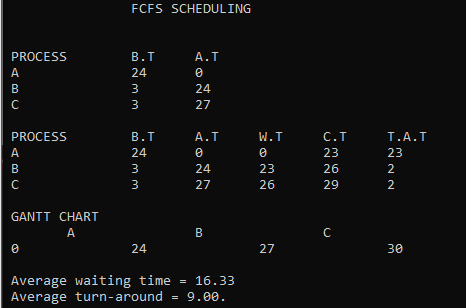
}while(ch != 8);

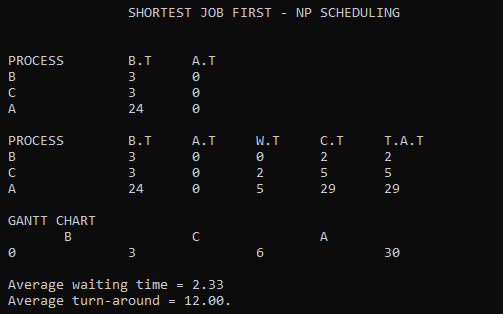
//getch();

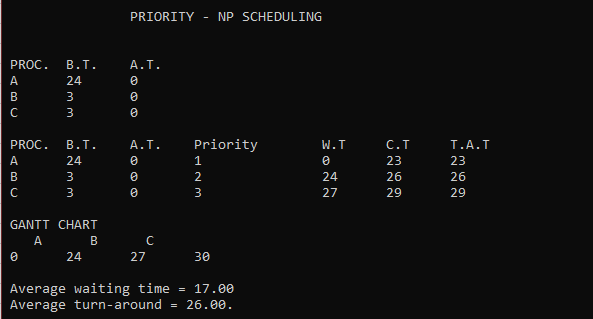
return 0;

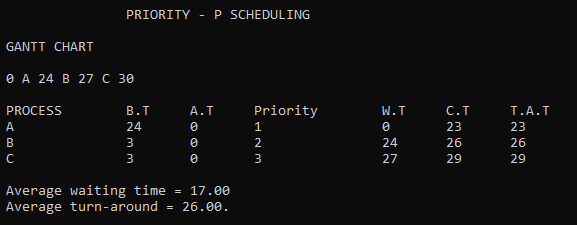
}

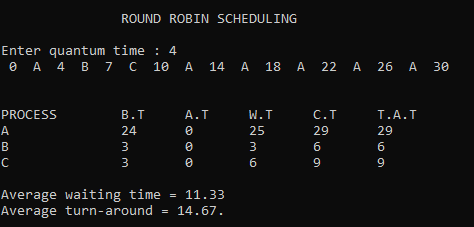
# **Outputs/Results:**











**Comparison via Graph:**

# **Conclusion:**

In view of the graph SJF (Shortest Job First) is best algorithm for process scheduling simulator because it gave least Waiting time and on the other side Priority scheduling is taking lot of waiting time as well as turn around time. FCFS (First Come First Serve) gives the least turnaround time but more waiting time than Round Robin algorithm.