OS WEEK 7

SCHEDULING ALGORITHMS

1.FCFS

#include <stdio.h>

#include <stdlib.h>

#define MAX 100

typedef struct

{

int pid;

int burst\_time;

int waiting\_time;

int turnaround\_time;

} Process;

void print\_table(Process p[], int n);

void print\_gantt\_chart(Process p[], int n);

int main()

{

Process p[MAX];

int i, j, n;

int sum\_waiting\_time = 0, sum\_turnaround\_time;

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

scanf("%d", &n);

printf("Enter burst time for each process:\n");

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

p[i].pid = i+1;

printf("P[%d] : ", i+1);

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

p[i].waiting\_time = p[i].turnaround\_time = 0;

}

// calculate waiting time and turnaround time

p[0].turnaround\_time = p[0].burst\_time;

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

p[i].waiting\_time = p[i-1].waiting\_time + p[i-1].burst\_time;

p[i].turnaround\_time = p[i].waiting\_time + p[i].burst\_time;

}

// calculate sum of waiting time and sum of turnaround time

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

sum\_waiting\_time += p[i].waiting\_time;

sum\_turnaround\_time += p[i].turnaround\_time;

}

// print table

puts(""); // Empty line

print\_table(p, n);

puts(""); // Empty Line

printf("Total Waiting Time : %-2d\n", sum\_waiting\_time);

printf("Average Waiting Time : %-2.2lf\n", (double)sum\_waiting\_time / (double) n);

printf("Total Turnaround Time : %-2d\n", sum\_turnaround\_time);

printf("Average Turnaround Time : %-2.2lf\n", (double)sum\_turnaround\_time / (double) n);

// print Gantt chart

puts(""); // Empty line

puts(" GANTT CHART ");

puts(" \*\*\*\*\*\*\*\*\*\*\* ");

print\_gantt\_chart(p, n);

return 0;

}

void print\_table(Process p[], int n)

{

int i;

puts("+-----+------------+--------------+-----------------+");

puts("| PID | Burst Time | Waiting Time | Turnaround Time |");

puts("+-----+------------+--------------+-----------------+");

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

printf("| %2d | %2d | %2d | %2d |\n"

, p[i].pid, p[i].burst\_time, p[i].waiting\_time, p[i].turnaround\_time );

puts("+-----+------------+--------------+-----------------+");

}

}

void print\_gantt\_chart(Process p[], int n)

{

int i, j;

// print top bar

printf(" ");

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

for(j=0; j<p[i].burst\_time; j++) printf("--");

printf(" ");

}

printf("\n|");

// printing process id in the middle

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

for(j=0; j<p[i].burst\_time - 1; j++) printf(" ");

printf("P%d", p[i].pid);

for(j=0; j<p[i].burst\_time - 1; j++) printf(" ");

printf("|");

}

printf("\n ");

// printing bottom bar

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

for(j=0; j<p[i].burst\_time; j++) printf("--");

printf(" ");

}

printf("\n");

// printing the time line

printf("0");

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

for(j=0; j<p[i].burst\_time; j++) printf(" ");

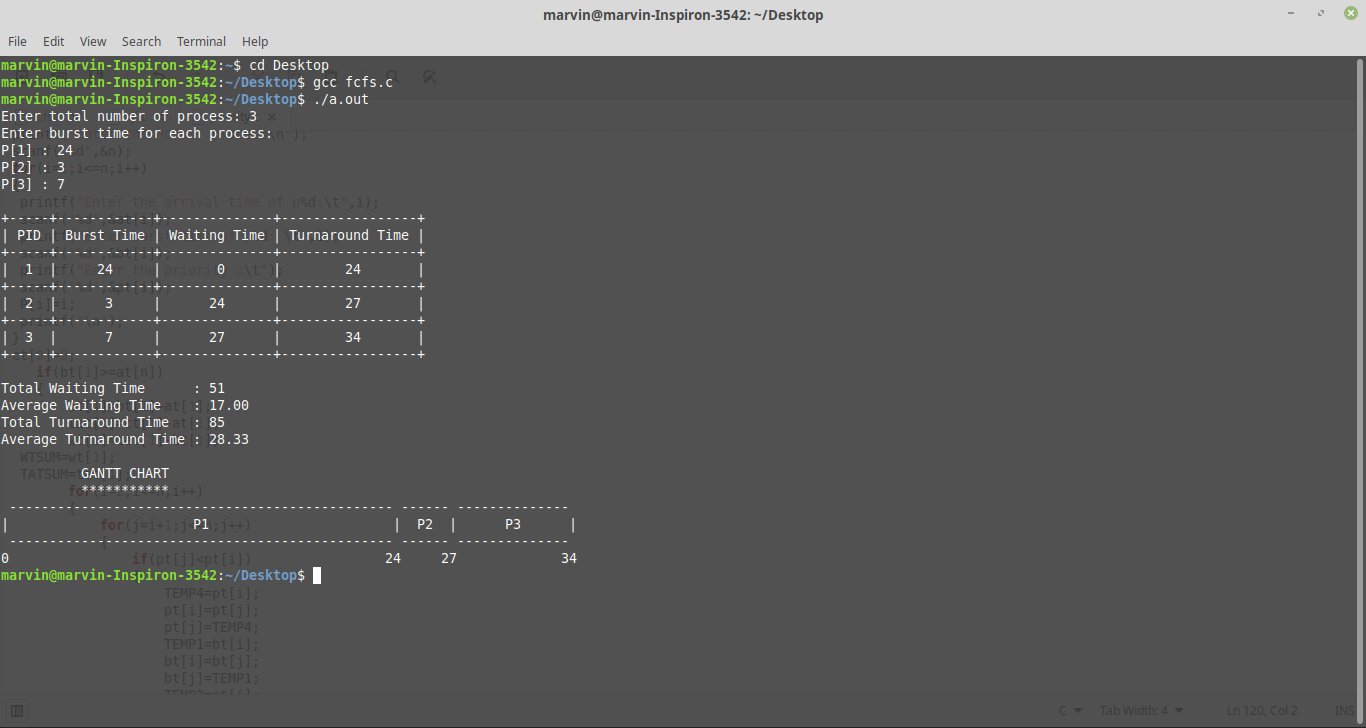
if(p[i].turnaround\_time > 9) printf("\b"); // backspace : remove 1 space

printf("%d", p[i].turnaround\_time);

}

printf("\n");

}



2.SJF

#include <stdio.h>

#include <stdlib.h>

#define MAX\_PROCESS 100

struct process {

int pid;

int burst\_time;

int waiting\_time;

};

typedef struct process Process;

double average\_waiting\_time;

int total\_waiting\_time;

void sort\_process\_by\_burst\_time(Process p[], int n);

void calculate\_waiting\_time(Process p[], int n);

void print\_gantt\_chart(Process p[], int n);

int main()

{

Process p[MAX\_PROCESS];

int n, i, j;

puts("SHORTEST JOB FIRST SCHEDULING ALGORITHM");

puts("=======================================");

printf("Enter total process: ");

scanf("%d", &n);

printf("Enter burst time for each process:\n");

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

printf("P[%d]: ", i+1);

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

p[i].pid = i+1;

}

sort\_process\_by\_burst\_time(p, n);

calculate\_waiting\_time(p, n);

average\_waiting\_time = (double) ( (double)total\_waiting\_time / (double) n );

puts("");

printf("Average Waiting Time: %.2lf\n",average\_waiting\_time);

printf("Gantt Chart:\n");

print\_gantt\_chart(p, n);

return 0;

}

void sort\_process\_by\_burst\_time(Process p[], int n)

{

int i, j;

Process temp;

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

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

if(p[j].burst\_time > p[j+1].burst\_time) {

temp = p[j];

p[j] = p[j+1];

p[j+1] = temp;

}

}

}

}

void calculate\_waiting\_time(Process p[], int n)

{

int i;

total\_waiting\_time = 0;

p[0].waiting\_time = 0;

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

p[i].waiting\_time = p[i-1].waiting\_time + p[i-1].burst\_time;

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

}

}

void print\_gantt\_chart(Process p[], int n)

{

int i, j;

int last = p[n-1].burst\_time + ( n== 1 ? 0 : p[n-1].waiting\_time);

// printing top bar

printf(" ");

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

for(j=0; j<p[i].burst\_time; j++) printf("--");

printf(" ");

}

printf("\n|");

// middle position

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

for(j=0; j<p[i].burst\_time-1; j++) printf(" ");

printf("p%d", p[i].pid);

for(j=0; j<p[i].burst\_time-1; j++) printf(" ");

printf("|");

}

printf("\n ");

// bottom bar

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

for(j=0; j<p[i].burst\_time; j++) printf("--");

printf(" ");

}

printf("\n");

// printing waiting time

int minus = 0;

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

if(p[i].waiting\_time>9) printf(" ");

printf("%d", p[i].waiting\_time);

if(p[i+1].waiting\_time>9){

minus = 1;

}

if(i+1 == n ) if (last>9) minus = 1;

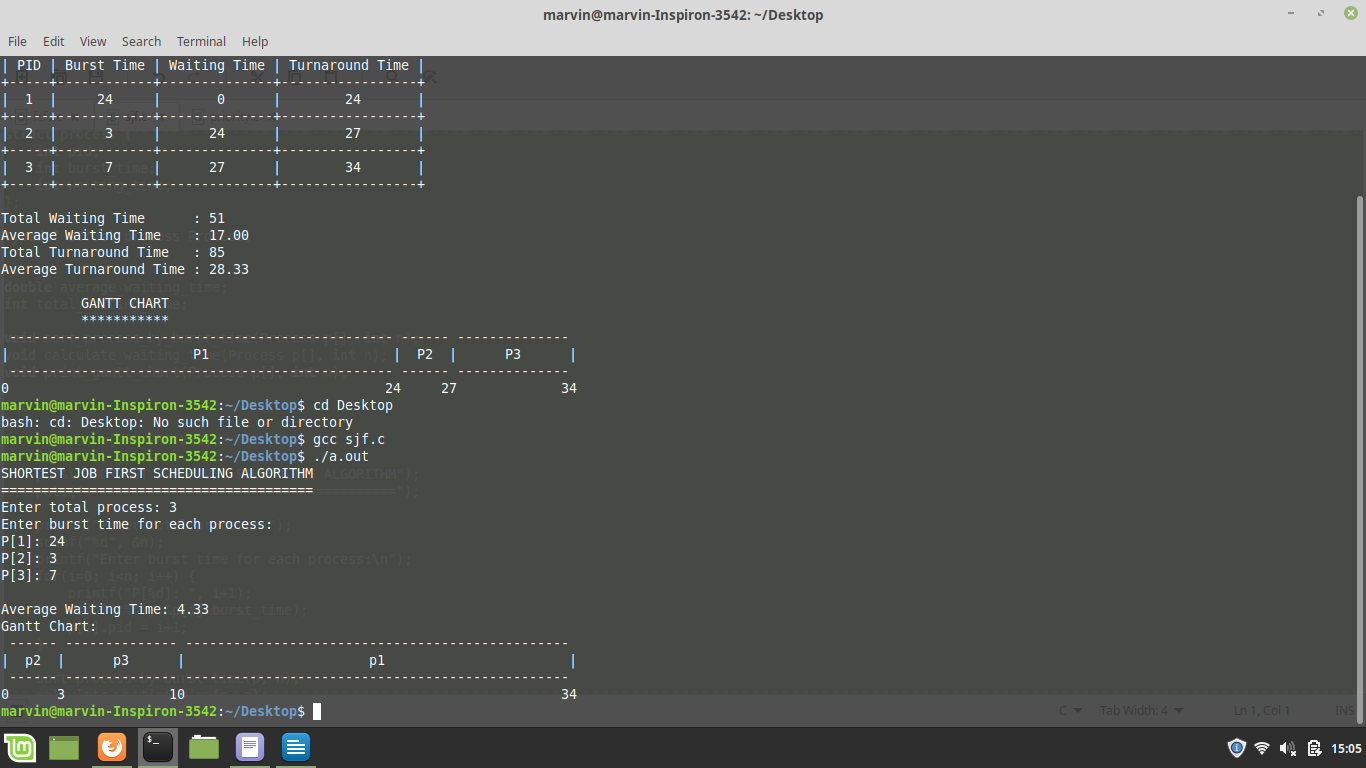
for(j=0; j<p[i].burst\_time-minus; j++) printf(" ");

}

if(last>9) printf(" ");

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

}



3.priority

void main()

{

int n,i,j,TEMP,TEMP1,TEMP2,TEMP3,TEMP4;

float WTSUM=0,TATSUM=0;

int bt[10],at[10],P[10],ct[10],tat[10],wt[10],pt[10];

printf("enter the no of process\n");

scanf("%d",&n);

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

{

printf("Enter the arrival time of p%d:\t",i);

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

printf("Enter burst time of p%d: \t",i);

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

printf("Enter the priority :\t");

scanf("%d",&pt[i]);

P[i]=i;

printf("\n");

}

ct[0]=0;

if(bt[1]>=at[n])

{

ct[1]=bt[1]+at[1];

tat[1]=ct[1]-at[1];

wt[1]=tat[1]-bt[1];

WTSUM=wt[1];

TATSUM=tat[1];

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

{

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

{

if(pt[j]<pt[i])

{

TEMP4=pt[i];

pt[i]=pt[j];

pt[j]=TEMP4;

TEMP1=bt[i];

bt[i]=bt[j];

bt[j]=TEMP1;

TEMP2=at[i];

at[i]=at[j];

at[j]=TEMP2;

TEMP3=P[i];

P[i]=P[j];

P[j]=TEMP3;

}

}

if(ct[i-1]<at[i])

{

TEMP=at[i]-ct[i-1];

ct[i]=ct[i-1]+bt[i]+TEMP;

TEMP1=bt[i];

}

else

{

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

}

tat[i]=ct[i]-at[i];

wt[i]=tat[i]-bt[i];

WTSUM=WTSUM+wt[i]+wt[1];

TATSUM=TATSUM+tat[i]+tat[1];

}

}

if(at[n]==0)

{

ct[0]=0;

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

{

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

{

if(pt[j]<pt[i])

{

TEMP4=pt[i];

pt[i]=pt[j];

pt[j]=TEMP4;

TEMP1=bt[i];

bt[i]=bt[j];

bt[j]=TEMP1;

TEMP2=at[i];

at[i]=at[j];

at[j]=TEMP2;

}

}

if(ct[i-1]<at[i])

{

TEMP=at[i]-ct[i-1];

ct[i]=ct[i-1]+bt[i]+TEMP;

TEMP1=bt[i];

}

else

{

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

}

tat[i]=ct[i]-at[i];

wt[i]=tat[i]-bt[i];

WTSUM=WTSUM+wt[i];

TATSUM=TATSUM+tat[i];

}

}

printf("\n\n\n");

printf("\tPROCESS\tBT\tAT\tPT\tTATshyam\tWT\n");

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

{

printf("\tP%d\t%d\t%d\t%d \t%d\t%d\n\n",P[i],bt[i],at[i],pt[i],tat[i],wt[i]);

}

printf("Average\_waiting\_time = %f\n",WTSUM/n);

printf("Average\_turn\_around\_time= %f\n",TATSUM/n);

printf("\n\n");

printf("Gantt chart\n");

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

{

printf("\tP%d\t",P[i]);

}

printf("\n");

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

{

printf("\t%d\t",ct[i]);

}

}

