/\*Develop a menu driven C program to implement the CPU Scheduling Algorithms

FCFS and SJF (Non-Preemptive and Preemptive)\*/

#include<stdio.h>

#include <stdlib.h>

#include<string.h>

//Struture representing each process

struct Job{

char \*PID;

double arrivalTime;

double burstTime;

double dummy;

double waitTime;

double turnTime;

double responseTime;

int nope;

};

typedef struct Job Process;

//Initialising the data members of each process

void initialise(Process \*p){

p->PID=(char\*)malloc(100\*sizeof(char));

p->arrivalTime=0.0;

p->burstTime=0.0;

p->dummy=0.0;

p->waitTime=0.0;

p->turnTime=0.0;

p->responseTime=-1.0;

p->nope=0;

}

//Accepting data of each process

void acceptProcess(Process \*p){

printf("\n Enter Process ID: ");scanf(" %s",p->PID);

printf("\n Enter arrival time: ");scanf("%lf",&p->arrivalTime);

printf("\n Enter burst time: ");scanf("%lf",&p->burstTime);

p->dummy=p->burstTime;

}

//Display Processes

void displayProcesses(Process p[],int number\_of\_processes){

for(int i=0;i<number\_of\_processes;i++)

printf("\t%s",p[i].PID);

printf("\n");

}

//Sorting using Insertion sort

void sortOnArrivalTime(Process p[],int start\_index,int end\_index){

for(int i=start\_index;i<end\_index;i++){

Process key=p[i];

int j=i-1;

for(;j>=start\_index&&key.arrivalTime<p[j].arrivalTime;j--)

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

p[j+1]=key;

}

}

//Display Gantt Chart

void displayGanttChart(char \*Gantt\_Chart[],int number\_of\_interval,double start\_times[],double end\_times[]){

//Display top line

printf("\n Gantt\_Chart:\n");

for(int i=0;i<number\_of\_interval;i++){

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

}

//Display order of processes

printf("\n|");

for(int i=0;i<number\_of\_interval;i++)

printf("\_\_\_\_\_%4s\_\_\_\_\_|",Gantt\_Chart[i]);

printf("\n");

//Display time line

int i=0;

for(i=0;i<number\_of\_interval;i++)

printf("%-15.0lf",start\_times[i]);

printf("%-15.0lf",end\_times[i-1]);

printf("\n\n");

}

//Print Wait Time

void printWaitTime(Process P[],int number\_of\_processes){

int i=0;

double sum=0.0;

printf("\n Wait Time:\n");

for(i=0;i<number\_of\_processes;i++){

printf(" %-5.2lf",P[i].waitTime);

sum+=P[i].waitTime;

}

printf("\nAverage: %-5.2lf",sum/number\_of\_processes);

printf("\n");

}

//Print Turnaround Time

void printTurnTime(Process P[],int number\_of\_processes){

int i=0;

double sum=0.0;

printf("\n Turnaround Time:\n");

for(i=0;i<number\_of\_processes;i++){

printf(" %-5.2lf",P[i].turnTime);

sum+=P[i].turnTime;

}

printf("\nAverage: %-5.2lf",sum/number\_of\_processes);

printf("\n");

}

//Print Response Time

void printRespTime(Process P[],int number\_of\_processes){

int i=0;

double sum=0.0;

printf("\n Response Time:\n");

for(i=0;i<number\_of\_processes;i++){

if(P[i].responseTime<0)

P[i].responseTime=0.0;

printf(" %-5.2lf",P[i].responseTime);

sum+=P[i].responseTime;

}

printf("\nAverage: %-5.2lf",sum/number\_of\_processes);

printf("\n");

}

//FCFS Scheduling

void FCFS(Process P[],int number\_of\_processes){

char \*Gantt\_Chart[100];

for(int i=0;i<100;i++)

Gantt\_Chart[i]=(char\*)malloc(10\*sizeof(char));

int interval=0;

double start\_times[100];

double end\_times[100];

sortOnArrivalTime(P,0,number\_of\_processes);

for(int i=0;i<number\_of\_processes;i++){

strcpy(Gantt\_Chart[interval],P[i].PID);

if(interval==0){

start\_times[interval]=0;

}

else{

start\_times[interval]=end\_times[interval-1];

}

end\_times[interval]=start\_times[interval]+P[i].burstTime;

P[i].waitTime=start\_times[interval]-P[i].arrivalTime;

P[i].turnTime=P[i].waitTime+P[i].burstTime;

P[i].responseTime=P[i].waitTime;

interval++;

}

displayGanttChart(Gantt\_Chart,interval,start\_times,end\_times);

printWaitTime(P,number\_of\_processes);

printTurnTime(P,number\_of\_processes);

printRespTime(P,number\_of\_processes);

}

//Sorting on Burst time

void sortOnBurstTime(Process p[],int number\_of\_processes){

for(int i=0;i<number\_of\_processes;i++){

Process key=p[i];

int j=i-1;

for(;j>=0&&key.burstTime<p[j].burstTime;j--)

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

p[j+1]=key;

}

}

//SJF Non-preemptive Scheduling

void Non\_PreSJF(Process P[],int number\_of\_processes){

//Total time of execution

double sum=0;

for(int i=0;i<number\_of\_processes;i++)

sum+=P[i].burstTime;

//Gantt chart

char \*Gantt\_Chart[100];

for(int i=0;i<100;i++)

Gantt\_Chart[i]=(char\*)malloc(10\*sizeof(char));

//Start and end times of processes

int interval=0;

double start\_times[100];

double end\_times[100];

for(int time=0;time<sum;){

Process tmp[100];

for(int i=0;i<100;i++)

initialise(&tmp[i]);

int tctr=0;

for(int i=0;i<number\_of\_processes;i++)

if(P[i].arrivalTime<=time&&P[i].burstTime){

tmp[tctr++]=P[i];

P[i].burstTime=0;

}

sortOnBurstTime(tmp,tctr);

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

strcpy(Gantt\_Chart[interval],tmp[i].PID);

if(interval==0){

start\_times[interval]=0;

}

else{

start\_times[interval]=end\_times[interval-1];

}

end\_times[interval]=start\_times[interval]+tmp[i].burstTime;

int j=0;

for(j=0;j<number\_of\_processes;j++){

if(strcmp(tmp[i].PID,P[j].PID)==0){

P[j].waitTime=start\_times[interval]-P[j].arrivalTime;

P[j].turnTime=P[j].waitTime+P[j].dummy;

P[j].responseTime=P[j].waitTime;

}

}

interval++;

}

time=end\_times[interval-1];

}

displayGanttChart(Gantt\_Chart,interval,start\_times,end\_times);

printWaitTime(P,number\_of\_processes);

printTurnTime(P,number\_of\_processes);

printRespTime(P,number\_of\_processes);

}

//SJF Preemptive Scheduling

void PreSJF(Process P[],int number\_of\_processes){

//Total time of execution

double sum=0;

for(int i=0;i<number\_of\_processes;i++)

sum+=P[i].burstTime;

//Gantt chart

char \*Gantt\_Chart[100];

for(int i=0;i<100;i++)

Gantt\_Chart[i]=(char\*)malloc(10\*sizeof(char));

//Start and end times of processes

int interval=0;

double start\_times[100];

double end\_times[100];

for(int time=0;time<sum;time++){

int flag=0;

Process tmp[100];

for(int i=0;i<100;i++)

initialise(&tmp[i]);

int tctr=0;

for(int i=0;i<number\_of\_processes;i++)

if(P[i].arrivalTime<=time&&P[i].burstTime){

tmp[tctr++]=P[i];

}

sortOnBurstTime(tmp,tctr);

for(int i=0;i<number\_of\_processes;i++){

if(strcmp(tmp[0].PID,P[i].PID)==0)

P[i].burstTime--;

}

if(interval==0){

strcpy(Gantt\_Chart[interval],tmp[0].PID);

start\_times[interval]=0;

flag=1;

interval++;

}

else{

if(strcmp(Gantt\_Chart[interval-1],tmp[0].PID)!=0){

end\_times[interval-1]=time;

strcpy(Gantt\_Chart[interval],tmp[0].PID);

start\_times[interval]=end\_times[interval-1];

flag=1;

interval++;

}

}

int j=0;

for(j=0;j<number\_of\_processes;j++){

if(flag&&strcmp(tmp[0].PID,P[j].PID)==0){

P[j].waitTime+=start\_times[interval-1]-P[j].arrivalTime;

if(P[j].waitTime>0.0){

P[j].nope++;

P[j].waitTime-=(P[j].dummy-P[j].burstTime-P[j].nope);

if(P[j].nope>1){

P[j].waitTime-=P[j].nope;

}

}

P[j].turnTime=P[j].waitTime+P[j].dummy;

if(P[j].responseTime<0.0)

P[j].responseTime=start\_times[interval-1]-P[j].arrivalTime;

}

}

}

end\_times[interval-1]=sum;

displayGanttChart(Gantt\_Chart,interval,start\_times,end\_times);

printWaitTime(P,number\_of\_processes);

printTurnTime(P,number\_of\_processes);

printRespTime(P,number\_of\_processes);

}

int main(){

printf("\n\t\tCPU SCHEDULING ALGORITHMS\n");

Process p[100];

int number\_of\_processes;

int algo\_option;

do{

printf("\nChoose your scheduling algorithm ");

printf("\n1. FCFS\n2. SJF\n0. Exit\n Your Choice: ");

scanf("%d",&algo\_option);

//FCFS Scheduling

if(algo\_option==1){

printf("\nEnter the number\_of\_processes:");scanf("%d",&number\_of\_processes);

printf("\nEnter the details of the processes:");

int i;

for(i=0;i<number\_of\_processes;i++){

initialise(&p[i]);

acceptProcess(&p[i]);

}

Process FCFSp[100];

for(i=0;i<number\_of\_processes;i++){

initialise(&FCFSp[i]);

FCFSp[i]=p[i];

}

printf("\n FCFS Scheduling Output:\n ");

FCFS(FCFSp,number\_of\_processes);

}

//SJF Scheduling

else if(algo\_option==2){

printf("\nEnter the number\_of\_processes:");scanf("%d",&number\_of\_processes);

printf("\nEnter the details of the processes:");

int i;

for(i=0;i<number\_of\_processes;i++){

initialise(&p[i]);

acceptProcess(&p[i]);

}

char preemp\_option;

printf("\n Use Pre-emption? y/n ");scanf(" %c",&preemp\_option);

//Non preemptive SJF Scheduling

if(preemp\_option=='n'||preemp\_option=='N'){

Process NSJFp[100];

for(i=0;i<number\_of\_processes;i++){

initialise(&NSJFp[i]);

NSJFp[i]=p[i];

}

printf("\n Non-preemptive SJF Scheduling Output:\n ");

Non\_PreSJF(NSJFp,number\_of\_processes);

}

//Preemptive SJF Scheduling

else if(preemp\_option=='y'||preemp\_option=='Y'){

Process SJFp[100];

for(i=0;i<number\_of\_processes;i++){

initialise(&SJFp[i]);

SJFp[i]=p[i];

}

printf("\n Preemptive SJF Scheduling Output:\n ");

PreSJF(SJFp,number\_of\_processes);

}

else{

printf("\n Invalid choice\n");

}

}

else if(algo\_option!=0){

printf("\n Invalid option\n");

}

else;

}while(algo\_option);

}

/\*

Output:

shivanirudh@shiva-ideapad:~/Desktop/Semester4/OSLAB/CPUScheduling$ gcc Scheduling.c -o a

shivanirudh@shiva-ideapad:~/Desktop/Semester4/OSLAB/CPUScheduling$ ./a

CPU SCHEDULING ALGORITHMS

Choose your scheduling algorithm

1. FCFS

2. SJF

0. Exit

Your Choice: 1

Enter the number\_of\_processes:5

Enter the details of the processes:

Enter Process ID: p1

Enter arrival time: 0

Enter burst time: 8

Enter Process ID: p2

Enter arrival time: 1

Enter burst time: 6

Enter Process ID: p3

Enter arrival time: 2

Enter burst time: 1

Enter Process ID: p4

Enter arrival time: 3

Enter burst time: 9

Enter Process ID: p5

Enter arrival time: 4

Enter burst time: 3

FCFS Scheduling Output:

Gantt\_Chart:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|\_\_\_\_\_ p1\_\_\_\_\_|\_\_\_\_\_ p2\_\_\_\_\_|\_\_\_\_\_ p3\_\_\_\_\_|\_\_\_\_\_ p4\_\_\_\_\_|\_\_\_\_\_ p5\_\_\_\_\_|

0 8 14 15 24 27

Wait Time:

0.00 7.00 12.00 12.00 20.00

Average: 10.20

Turnaround Time:

8.00 13.00 13.00 21.00 23.00

Average: 15.60

Response Time:

0.00 7.00 12.00 12.00 20.00

Average: 10.20

Choose your scheduling algorithm

1. FCFS

2. SJF

0. Exit

Your Choice: 2

Enter the number\_of\_processes:5

Enter the details of the processes:

Enter Process ID: p1

Enter arrival time: 0

Enter burst time: 8

Enter Process ID: p2

Enter arrival time: 1

Enter burst time: 6

Enter Process ID: p3

Enter arrival time: 2

Enter burst time: 1

Enter Process ID: p4

Enter arrival time: 3

Enter burst time: 9

Enter Process ID: p5

Enter arrival time: 4

Enter burst time: 3

Use Pre-emption? y/n n

Non-preemptive SJF Scheduling Output:

Gantt\_Chart:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|\_\_\_\_\_ p1\_\_\_\_\_|\_\_\_\_\_ p3\_\_\_\_\_|\_\_\_\_\_ p5\_\_\_\_\_|\_\_\_\_\_ p2\_\_\_\_\_|\_\_\_\_\_ p4\_\_\_\_\_|

0 8 9 12 18 27

Wait Time:

0.00 11.00 6.00 15.00 5.00

Average: 7.40

Turnaround Time:

8.00 17.00 7.00 24.00 8.00

Average: 12.80

Response Time:

0.00 11.00 6.00 15.00 5.00

Average: 7.40

Choose your scheduling algorithm

1. FCFS

2. SJF

0. Exit

Your Choice: 2

Enter the number\_of\_processes:5

Enter the details of the processes:

Enter Process ID: p1

Enter arrival time: 0

Enter burst time: 8

Enter Process ID: p2

Enter arrival time: 1

Enter burst time: 6

Enter Process ID: p3

Enter arrival time: 2

Enter burst time: 1

Enter Process ID: p4

Enter arrival time: 3

Enter burst time: 9

Enter Process ID: p5

Enter arrival time: 4

Enter burst time: 3

Use Pre-emption? y/n y

Preemptive SJF Scheduling Output:

Gantt\_Chart:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|\_\_\_\_\_ p1\_\_\_\_\_|\_\_\_\_\_ p2\_\_\_\_\_|\_\_\_\_\_ p3\_\_\_\_\_|\_\_\_\_\_ p2\_\_\_\_\_|\_\_\_\_\_ p5\_\_\_\_\_|\_\_\_\_\_ p2\_\_\_\_\_|\_\_\_\_\_ p1\_\_\_\_\_|\_\_\_\_\_ p4\_\_\_\_\_|

0 1 2 3 4 7 11 18 27

Wait Time:

10.00 4.00 0.00 15.00 0.00

Average: 5.80

Turnaround Time:

18.00 10.00 1.00 24.00 3.00

Average: 11.20

Response Time:

0.00 0.00 0.00 15.00 0.00

Average: 3.00

Choose your scheduling algorithm

1. FCFS

2. SJF

0. Exit

Your Choice: 0

\*/