//Priority

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

typedef struct

{

char ppid[10];

int at;

int bt;

int wt,tat,rt;

int pr;

}proces;

typedef struct

{

char ppid[100];

int start,end;

}gantt;

void input(proces \*p,int n);

void display(proces \*p,int n);

void schedule\_priority\_np(proces \*t,int n);

void schedule\_priority\_p(proces \*t,int n);

int smallest\_process(proces \*p,int n);

void main()

{

proces \*p;

int n;

int ch=0;

char cho;

while(ch!=3)

{

printf(" 1.Priority\_Non\_preemtive \n 2.Priority\_Preemtive\n 3.EXIT\n\n Choice : ");

scanf("%d",&ch);

if(ch==1)

{

printf("Enter the no. of proceses : ");

scanf("%d",&n);

p=(proces\*)malloc(sizeof(proces)\*n);

input(p,n);

schedule\_priority\_np(p,n);

display(p,n);

free(p);

}

else if(ch==2)

{

printf("Enter the no. of proceses : ");

scanf("%d",&n);

p=(proces\*)malloc(sizeof(proces)\*n);

input(p,n);

schedule\_priority\_p(p,n);

display(p,n);

free(p);

}

}

}

void input(proces \*p,int n)

{

int i;

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

{

printf("\n\nEnter proces %d\n",i+1);

printf("Enter proces id : ");

scanf("%s",p[i].ppid);

printf("Enter arrival time : ");

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

printf("Enter burst time : ");

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

printf("Enter priority : ");

scanf("%d",&(p[i].pr));

}

}

void display(proces \*p,int n)

{

int i;

float avg\_wt=0,avg\_rt=0,avg\_tat=0;

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

printf("proces ID Arrival Time Burst Time TurnaroundTime Waiting Time Response Time\n");

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

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

{

printf("%s %d %d %d %d %d \n",p[i].ppid,p[i].at,p[i].bt,p[i].tat,p[i].wt,p[i].rt);

avg\_wt+=p[i].wt;

avg\_rt+=p[i].rt;

avg\_tat+=p[i].tat;

}

avg\_wt/=n;

avg\_rt/=n;

avg\_tat/=n;

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

printf(" Average %f %f %f\n",avg\_tat,avg\_wt,avg\_rt);

}

void schedule\_priority\_np(proces \*t,int n)

{

proces temp[n];

int i,j,a=-1,b=0;

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

{

if(t[i].at>a)

a=t[i].at;

t[i].wt=0;

t[i].rt=-1;

t[i].tat=0;

}

int k=0;

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

{

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

{

if(t[j].at==i)

{

temp[k]=t[j];

k++;

if(k>=n)

break;

}

}

}

k=0;

int max=temp[0].at;

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

{

if(t[i].at>max)

max=t[i].at;

max+=t[i].bt;

}

a=0;

proces temporary[n];

j=0;

proces idle,small,previous;

strcpy(idle.ppid,"idle");

idle.at=-1;

idle.bt=-1;

idle.pr=-1;

int f=0;

gantt g[100];

int index,count=0,gun[100];

j=0;

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

{

if(temp[i].at<=a && temp[i].bt>0)

{

temporary[j]=temp[i];

j++;

}

}

if(j==0)

{

small=idle;

}

else

{

index=smallest\_process(temporary,j);

small=temporary[index];

}

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

while(a<max)

{

if(strcmp(small.ppid,previous.ppid))

{

gun[count]=a;

count++;

printf(" : %d %s ",gun[count-1],small.ppid);

g[f].start=gun[count-1];

strcpy(g[f].ppid,small.ppid);

}

if(!strcmp(small.ppid,idle.ppid))

{

a++;

}

else

{

a+=small.bt;

temporary[index].bt=0;

}

if(j!=0)

{

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

{

if(!strcmp(temporary[index].ppid,temp[i].ppid))

{

temp[i]=temporary[index];

}

}

}

previous=small;

j=0;

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

{

if(temp[i].at<=a && temp[i].bt>0)

{

temporary[j]=temp[i];

j++;

}

}

if(j==0)

{

small=idle;

}

else

{

index=smallest\_process(temporary,j);

small=temporary[index];

}

if(strcmp(small.ppid,previous.ppid))

{

gun[count]=a;

printf("%d :",gun[count]);

count++;

g[f].end=gun[count-1];

f++;

}

}

printf("\n");

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

{

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

{

if(!strcmp(temp[j].ppid,g[i].ppid))

{

if(temp[j].rt==-1)

{

temp[j].rt=g[i].start-temp[j].at;

temp[j].wt=temp[j].rt;

temp[j].tat=g[i].end-temp[j].at;

}

}

}

}

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

{

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

{

if(!strcmp(temp[k].ppid,t[i].ppid))

{

t[i].rt=temp[k].rt;

t[i].tat=temp[k].tat;

t[i].wt=temp[k].wt;

}

}

}

}

void schedule\_priority\_p(proces \*t,int n)

{

proces temp[n];

int i,j,a=-1,b=0;

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

{

if(t[i].at>a)

a=t[i].at;

t[i].wt=0;

t[i].rt=-1;

t[i].tat=0;

}

int k=0;

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

{

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

{

if(t[j].at==i)

{

temp[k]=t[j];

k++;

if(k>=n)

break;

}

}

}

k=0;

int max=temp[0].at;

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

{

if(t[i].at>max)

max=t[i].at;

max+=t[i].bt;

}

a=0;

proces temporary[n];

j=0;

proces idle,small,previous;

strcpy(idle.ppid,"idle");

idle.at=-1;

idle.bt=-1;

idle.pr=-1;

int f=0;

gantt g[100];

int index,count=0,gun[100];

j=0;

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

{

if(temp[i].at<=a && temp[i].bt>0)

{

temporary[j]=temp[i];

j++;

}

}

if(j==0)

{

small=idle;

}

else

{

index=smallest\_process(temporary,j);

small=temporary[index];

}

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

while(a<max)

{

if(strcmp(small.ppid,previous.ppid))

{

gun[count]=a;

count++;

printf(" : %d %s ",gun[count-1],small.ppid);

g[f].start=gun[count-1];

strcpy(g[f].ppid,small.ppid);

}

if(!strcmp(small.ppid,idle.ppid))

{

a++;

}

else

{

a++;

temporary[index].bt--;

}

if(j!=0)

{

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

{

if(!strcmp(temporary[index].ppid,temp[i].ppid))

{

temp[i]=temporary[index];

}

}

}

previous=small;

j=0;

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

{

if(temp[i].at<=a && temp[i].bt>0)

{

temporary[j]=temp[i];

j++;

}

}

if(j==0)

{

small=idle;

}

else

{

index=smallest\_process(temporary,j);

small=temporary[index];

}

if(strcmp(small.ppid,previous.ppid))

{

gun[count]=a;

printf("%d :",gun[count]);

count++;

g[f].end=gun[count-1];

f++;

}

}

printf("\n");

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

{

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

{

if(!strcmp(temp[j].ppid,g[i].ppid))

{

if(temp[j].rt==-1)

{

temp[j].rt=g[i].start-temp[j].at;

}

temp[j].tat=g[i].end-temp[j].at;

}

}

}

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

{

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

{

if(!strcmp(temp[k].ppid,t[i].ppid))

{

t[i].rt=temp[k].rt;

t[i].tat=temp[k].tat;

}

}

}

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

{

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

}

}

int smallest\_process(proces \*p,int n)

{

int b=p[0].pr;

int index=0;

int i;

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

{

if(p[i].pr<b)

{

b=p[i].pr;

index=i;

}

}

return index;

}

/\*

gml29:Desktop cse29$ gcc -o k scheduling2.c

gml29:Desktop cse29$ ./k

1.Priority\_Non\_preemtive

2.Priority\_Preemtive

3.EXIT

Choice : 1

Enter the no. of proceses : 3

Enter proces 1

Enter proces id : p1

Enter arrival time : 0

Enter burst time : 3

Enter priority : 3

Enter proces 2

Enter proces id : p2

Enter arrival time : 2

Enter burst time : 1

Enter priority : 2

Enter proces 3

Enter proces id : p3

Enter arrival time : 3

Enter burst time : 5

Enter priority : 1

GANTT CHART

: 0 p1 3 : : 3 p3 8 : : 8 p2 9 :

---------------------------------------------------------------------------------------------

proces ID Arrival Time Burst Time TurnaroundTime Waiting Time Response Time

---------------------------------------------------------------------------------------------

p1 0 3 3 0 0

p2 2 1 7 6 6

p3 3 5 5 0 0

---------------------------------------------------------------------------------------------

Average 5.000000 2.000000 2.000000

1.Priority\_Non\_preemtive

2.Priority\_Preemtive

3.EXIT

Choice : 2

Enter the no. of proceses : 3

Enter proces 1

Enter proces id : p1

Enter arrival time : 0

Enter burst time : 3

Enter priority : 3

Enter proces 2

Enter proces id : p2

Enter arrival time : 1

Enter burst time : 5

Enter priority : 2

Enter proces 3

Enter proces id : p3

Enter arrival time : 2

Enter burst time : 4

Enter priority : 1

GANTT CHART

: 0 p1 1 : : 1 p2 2 : : 2 p3 6 : : 6 p2 10 : : 10 p1 12 :

---------------------------------------------------------------------------------------------

proces ID Arrival Time Burst Time TurnaroundTime Waiting Time Response Time

---------------------------------------------------------------------------------------------

p1 0 3 12 9 0

p2 1 5 9 4 0

p3 2 4 4 0 0

---------------------------------------------------------------------------------------------

Average 8.333333 4.333333 0.000000

1.Priority\_Non\_preemtive

2.Priority\_Preemtive

3.EXIT

Choice : 3

gml29:Desktop cse29$

\*/

//RoundRobin

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

typedef struct

{

char ppid[10];

int at;

int bt;

int wt,tat,rt;

}proces;

int Quantum=0;

typedef struct

{

char ppid[100];

int start,end;

}gantt;

typedef struct

{

proces a[100];

int size,f,r;

}queue;

queue\* initialize(int s);

int isfull(queue \*q);

int isempty(queue \*q);

void enqueue(queue \*q,proces x);

proces dequeue(queue \*q);

proces view(queue \*q);

void input(proces \*p,int n);

void display(proces \*p,int n);

void RoundRobin(proces \*p,int n);

void disp(queue \*q)

{

int i;

if(q->f<q->r)

{

for(i=q->f;i<q->r;i++)

{

printf("%s",q->a[i].ppid);

}

}

else if(q->f>q->r)

{

for(i=q->f;i<q->size;i++)

{

printf("%s",q->a[i].ppid);

}

for(i=0;i<q->r;i++)

{

printf("%s",q->a[i].ppid);

}

}

}

queue\* initialize(int s)

{

queue \*q;

q=(queue\*)malloc(sizeof(q));

q->f=0;

q->r=0;

q->size=s+1;

return q;

}

int isfull(queue \*q)

{

if(((q->r==q->size-1)&&(q->f==0))||(q->r+1==q->f))

return 1;

return 0;

}

int isempty(queue \*q)

{

if(q->r==q->f)

return 1;

return 0;

}

void enqueue(queue \*q,proces x)

{

if(isfull(q))

{

printf("queue is full\n");

return;

}

q->a[q->r]=x;

q->r=(q->r+1)%q->size;

}

proces dequeue(queue \*q)

{

proces a;

strcpy(a.ppid,"idle");

a.bt=-1;

a.at=-1;

if(isempty(q))

{

return a;

}

int x=q->f;

q->f=(x+1)%q->size;

return q->a[x];

}

proces view(queue \*q)

{

proces a;

strcpy(a.ppid,"idle");

a.bt=-1;

a.at=-1;

if(isempty(q))

{

return a;

}

return q->a[q->f];

}

void main()

{

proces \*p;

int n;

int ch=0;

char cho;

while(ch!=2)

{

printf(" 1.RoundRobin \n 2.EXIT\n\n Choice : ");

scanf("%d",&ch);

if(ch==1)

{

printf("Enter the no. of proceses : ");

scanf("%d",&n);

p=(proces\*)malloc(sizeof(proces)\*n);

input(p,n);

RoundRobin(p,n);

display(p,n);

free(p);

}

}

}

void input(proces \*p,int n)

{

int i;

printf("Enter Time Slice : ");

scanf("%d",&Quantum);

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

{

printf("\n\nEnter proces %d\n",i+1);

printf("Enter proces id : ");

scanf("%s",p[i].ppid);

printf("Enter arrival time : ");

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

printf("Enter burst time : ");

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

p[i].rt=-1;

p[i].tat=-1;

p[i].wt=-1;

}

}

void display(proces \*p,int n)

{

int i;

float avg\_wt=0,avg\_rt=0,avg\_tat=0;

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

printf("proces ID Arrival Time Burst Time TurnaroundTime Waiting Time Response Time\n");

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

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

{

printf("%s %d %d %d %d %d \n",p[i].ppid,p[i].at,p[i].bt,p[i].tat,p[i].wt,p[i].rt);

avg\_wt+=p[i].wt;

avg\_rt+=p[i].rt;

avg\_tat+=p[i].tat;

}

avg\_wt/=n;

avg\_rt/=n;

avg\_tat/=n;

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

printf(" Average %f %f %f\n",avg\_tat,avg\_wt,avg\_rt);

}

void RoundRobin(proces \*t,int n)

{

queue \*rq,\*aq;

rq=initialize(100);

aq=initialize(100);

proces temp[n],p[n];

int i,j,k,a=t[0].at;

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

{

if(t[i].at>a)

{

a=t[i].at;

}

}

k=0;

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

{

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

{

if(i==t[j].at)

{

temp[k]=t[j];

k++;

if(k==n)

break;

}

}

}

int max=temp[0].at;

int flag[n];

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

{

flag[i]=0;

if(t[i].at>max)

max=t[i].at;

max+=t[i].bt;

}

a=0;

j=0;

int l=0;

proces idle,prev;

strcpy(idle.ppid,"idle");

idle.at=-1;

idle.bt=-1;

gantt g[100];

int gun[100],count=0;

int index=0;

int pr\_j=0;

while(a<max)

{

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

{

if(temp[i].at<=a && temp[i].bt>0 && flag[i]==0)

{

enqueue(aq,temp[i]);

flag[i]=1;

}

}

if(isempty(rq))

{

if(isempty(aq))

{

if(strcmp(prev.ppid,idle.ppid))

{

gun[count]=a;

count++;

printf(": %d idle ",gun[count-1]);

}

a+=1;

prev=idle;

}

}

if(isempty(rq))

{

if(!isempty(aq))

{

while(!isempty(aq))

{

enqueue(rq,dequeue(aq));

}

if(view(rq).bt>Quantum)

{

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

{

if(temp[i].at<=(a+Quantum) && temp[i].bt>0 && flag[i]==0)

{

enqueue(aq,temp[i]);

flag[i]=1;

}

}

}

else

{

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

{

if(temp[i].at<=view(rq).bt && temp[i].bt>0 && flag[i]==0)

{

enqueue(aq,temp[i]);

flag[i]=1;

}

}

}

}

}

if(!isempty(rq))

{

proces cpu=dequeue(rq);

gun[count]=a;

count++;

if(!strcmp(prev.ppid,idle.ppid))

{

printf("%d :",gun[count-1]);

}

prev=cpu;

if(cpu.bt>Quantum)

{

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

{

if(temp[i].at<=(a+Quantum) && temp[i].bt>0 && flag[i]==0)

{

enqueue(aq,temp[i]);

flag[i]=1;

}

}

}

else

{

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

{

if(temp[i].at<=cpu.bt && temp[i].bt>0 && flag[i]==0)

{

enqueue(aq,temp[i]);

flag[i]=1;

}

}

}

if(cpu.bt>Quantum)

{

cpu.bt=cpu.bt-=Quantum;

a+=Quantum;

if(isempty(aq))

{

enqueue(rq,cpu);

}

else

{

while(!isempty(aq))

enqueue(rq,dequeue(aq));

enqueue(rq,cpu);

}

}

else

{

a+=cpu.bt;

cpu.bt=0;

if(!isempty(aq))

{

while(!isempty(aq))

enqueue(rq,dequeue(aq));

}

}

gun[count]=a;

g[l].start=gun[count-1];

g[l].end=gun[count];

strcpy(g[l].ppid,cpu.ppid);

l++;

count++;

printf(": %d %s %d :",gun[count-2],cpu.ppid,gun[count-1]);

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

{

if(!strcmp(cpu.ppid,temp[i].ppid))

{

temp[i]=cpu;

}

}

}

}

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

{

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

{

if(!strcmp(g[i].ppid,temp[j].ppid))

{

if(temp[j].rt==-1)

{

temp[j].rt=g[i].start-temp[j].at;

}

temp[j].tat=g[i].end-temp[j].at;

}

}

}

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

{

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

{

if(!strcmp(temp[k].ppid,t[i].ppid))

{

t[i].rt=temp[k].rt;

t[i].tat=temp[k].tat;

}

}

}

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

{

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

}

printf("\n");

}

/\*

PS F:\SEM4\OS\Assignment3> gcc -o k RRwq.c PS F:\SEM4\OS\Assignment3> ./k 1.RoundRobin

2.EXIT

Choice : 1

Enter the no. of proceses : 4

Enter Time Slice : 2

Enter proces 1

Enter proces id : p1

Enter arrival time : 1

Enter burst time : 4

Enter proces 2

Enter proces id : p2

Enter arrival time : 2

Enter burst time : 5

Enter proces 3

Enter proces id : p3

Enter arrival time : 3

Enter burst time : 4

Enter proces 4

Enter proces id : p4

Enter arrival time : 4

Enter burst time : 3

: 0 idle 1 :: 1 p1 3 :: 3 p2 5 :: 5 p3 7 :: 7 p1 9 :: 9 p4 11 :: 11 p2 13 :: 13 p3 15 :: 15 p4 16 :: 16 p2 17 :

---------------------------------------------------------------------------------------------

proces ID Arrival Time Burst Time TurnaroundTime Waiting Time Response Time

---------------------------------------------------------------------------------------------

p1 1 4 8 4 0

p2 2 5 15 10 1

p3 3 4 12 8 2

p4 4 3 12 9 5

---------------------------------------------------------------------------------------------

Average 11.750000 7.750000 2.000000

\*/