DINING PHIL

#include<semaphore.h>

#include<pthread.h>

#include<stdio.h>

#include<string.h>

int think=0,hungry=1,eat=2;

int state[5];

int numberofphilosopher[5];

sem\_t mutex,self[5];

void test(int philosopher)

{

if (state[philosopher] == hungry && state[(philosopher+4)%5] != eat && state[(philosopher+1)%5] != eat)

{

state[philosopher] = eat;

printf("Philosopher %d takes fork %d and %d\n",philosopher+1,(philosopher+4)%5+1,philosopher+1);

printf("Philosopher %d is eat\n",philosopher+1);

sem\_post(&self[philosopher]);

}

}

void putdown(int philosopher)

{

sem\_wait(&mutex);

state[philosopher] = think;

printf("Philosopher %d is currently thinking\n",philosopher+1);

test((philosopher+4)%5);

test((philosopher+1)%5);

sem\_post(&mutex);

}

void \*philospherPROCESS(void \*num)

{

while(1)

{

int \*i = num;

pickup(\*i);

putdown(\*i);

}

}

void pickup(int philosopher)

{

sem\_wait(&mutex);

state[philosopher] = hungry;

printf("Philosopher %d is currently hungry\n",philosopher+1);

test(philosopher);

sem\_post(&mutex);

sem\_wait(&self[philosopher]);

}

int main()

{

int i;

for(i=0;i<5;i++)

numberofphilosopher[i]=i;

pthread\_t thread\_id[5];

sem\_init(&mutex,0,1);

for(i=0;i<5;i++)

sem\_init(&self[i],0,0);

for(i=0;i<5;i++)

{

pthread\_create(&thread\_id[i],NULL,philospherPROCESS,&numberofphilosopher[i]);

printf("Philosopher %d is currently thinking\n",i+1);

}

for(i=0;i<5;i++)

pthread\_join(thread\_id[i],NULL);

}

Fcfs no prempt

/\*done by M.S.SANJAY 15BCE0517\*/

#include<stdio.h>

void main()

{

int bt[100],p[100],wt[100],tat[100],i,j,n,pp,t;

float tot=0;

float awt,awtt;

printf("enter the total number of processes:");

scanf("%d",&n);

printf("enter burst time for all process");

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

{

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

p[i]=i+1;

}

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

{

pp=i;

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

{

if(bt[j]<bt[pp])

pp=j;

}

t=bt[i];

bt[i]=bt[pp];

bt[pp]=t;

t=p[i];

p[i]=p[pp];

p[pp]=t;

}

wt[0]=0;

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

{

wt[i]=0;

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

wt[i]+=bt[j];

tot+=wt[i];

}

awt=tot/n;

tot=0;

printf("\nprocess burst time waiting time turnaround time\n");

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

{

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

tot+=tat[i];

printf("\n%d\t %d\t\t %d\t%d",p[i],bt[i],wt[i],tat[i]);

}

awtt=tot/n;

printf("\n\nAverage Waiting Time=%f",awt);

printf("\nAverage Turnaround Time=%f\n",awtt);

}

Prempt

//SANJAY M.S.

//15BCE0517

#include<stdio.h>

void main()

{

float fcfstime,sjftime,prioritytime;

int flip;

float awt=0,att=0;

int n,b[10],w[10],t[10];

int i,j,pn[10],p[10];

printf("PRE EMPTIVE SCHEDULING ALGORITHMS\n\n");

printf("How many processes do u what?\n");

scanf("%d",&n);

printf("\nEnter burst time and priority for %d processes\n",n);

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

{printf("\nENTER BURST TIME:");

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

pn[i]=i+1;

printf("\nENTER PRIORITY TIME:");

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

}

w[0]=0;

int time=b[0];

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

{

w[i]=time;

time=time+b[i];

awt+=w[i];

}

awt=awt/n;

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

{

t[i]=b[i]+w[i];

att+=t[i];

}

att=att/n;

printf("For fcfs scheduling the output is\n");

printf("PROCESS'S NUMBER\t\tBURST TIME\t\tWAITING TIME\t\tTURNAROUND TIME\n");

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

{

printf("%d\t\t%d\t\t%d\t\t%d\n",i+1,b[i],w[i],t[i]);

}

printf("AVERAGE WAITING TIME IS=%f \t\t AVERAGE TURNARoUND TIME=%f \n",awt,att);

fcfstime=att;

att=0;awt=0;

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

{

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

{

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

{

flip=b[j];

b[j]=b[i];

b[i]=flip;

flip=pn[i];

pn[i]=pn[j];

pn[j]=flip;

flip=p[i];

p[i]=p[j];

p[j]=flip;

}

}

}

w[0]=0;

time=b[0];

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

{

w[i]=time;

time=time+b[i];

awt+=w[i];

}

awt=awt/n;

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

{

t[i]=b[i]+w[i];

att+=t[i];

}

att=att/n;

printf("for priority scheduling the output is\n");

printf("PROCESS'S NUMBER\t\tBURST TIME\t\tWAITING TIME\t\tTURNAROUND TIME\n");

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

{

printf("%d\t\t%d\t\t%d\t\t%d\n",pn[i],b[i],w[i],t[i]);

}

printf("AVERAGE WAITING TIME IS=%f \t\t AVERAGE TURNARoUND TIME=%f \n",awt,att);

prioritytime=att;

att=0;awt=0;

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

{

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

{

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

{

flip=b[j];

b[j]=b[i];

b[i]=flip;

flip=pn[i];

pn[i]=pn[j];

pn[j]=flip;

}

}

}

w[0]=0;

time=b[0];

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

{

w[i]=time;

time=time+b[i];

awt+=w[i];

}

awt=awt/n;

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

{

t[i]=b[i]+w[i];

att+=t[i];

}

att=att/n;

printf("For shortest job first the output is\n");

printf("PROCESS'S NUMBER\t\tBURST TIME\t\tWAITING TIME\t\tTURNAROUND TIME\n");

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

{

printf("%d\t\t%d\t\t%d\t\t%d\n",pn[i],b[i],w[i],t[i]);

}

printf("AVERAGE WAITING TIME IS=%f \t\t AVERAGE TURNARoUND TIME=%f \n",awt,att);

sjftime=att;

if(prioritytime<sjftime&&prioritytime<fcfstime)

printf("PRIORITY ALGORITHM IS BEST IN THIS CASE");

else if (sjftime<fcfstime&&sjftime<prioritytime)

printf("SHORTEST JOB FIRST ALGORITHM IS BEST IN THIS CASE");

else

printf("FIRST COME FIRST SERVE ALGORITHM IS BEST IN THIS CASE");

}

Sjf no prempt

/\*done by M.S.SANJAY 15BCE0517\*/

#include<stdio.h>

void main()

{

int bt[100],p[100],wt[100],tat[100],i,j,n,pp,t;

float tot=0;

float awt,awtt;

printf("enter the total number of processes:");

scanf("%d",&n);

printf("enter burst time for all process");

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

{

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

p[i]=i+1;

}

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

{

pp=i;

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

{

if(bt[j]<bt[pp])

pp=j;

}

t=bt[i];

bt[i]=bt[pp];

bt[pp]=t;

t=p[i];

p[i]=p[pp];

p[pp]=t;

}

wt[0]=0;

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

{

wt[i]=0;

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

wt[i]+=bt[j];

tot+=wt[i];

}

awt=tot/n;

tot=0;

printf("\nprocess burst time waiting time turnaround time\n");

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

{

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

tot+=tat[i];

printf("\n%d\t %d\t\t %d\t%d",p[i],bt[i],wt[i],tat[i]);

}

awtt=tot/n;

printf("\n\nAverage Waiting Time=%f",awt);

printf("\nAverage Turnaround Time=%f\n",awtt);

}

Prio non premptiv

#include<stdio.h>

int main()

{

int bt[20],p[20],wt[20],tat[20],pr[20],i,j,n,pos,temp;

float awt,atat;

float total=0;

printf("Enter Total Number of Process:");

scanf("%d",&n);

printf("\nEnter Burst Time and Priority\n");

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

{

printf("Burst Time:");

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

printf("Priority:");

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

p[i]=i+1;

}

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

{

pos=i;

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

{

if(pr[j]<pr[pos])

pos=j;

}

temp=pr[i];

pr[i]=pr[pos];

pr[pos]=temp;

temp=bt[i];

bt[i]=bt[pos];

bt[pos]=temp;

temp=p[i];

p[i]=p[pos];

p[pos]=temp;

}

wt[0]=0;

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

{

wt[i]=0;

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

wt[i]+=bt[j];

total+=wt[i];

}

awt=total/n;

total=0;

printf("\nprocess burst time waiting time turnaround time");

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

{

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

total+=tat[i];

printf("\n%d\t %d\t %d\t %d",p[i],bt[i],wt[i],tat[i]);

}

atat=total/n;

printf("\n\avg waiting Time=%f",awt);

printf("\navg turnaround Time=%f\n",atat);

return 0;

}

Rund robin

#include<stdio.h>

int main()

{ int flip;

int time,remain,flag=0,ts;

float sum\_wait=0,sum\_turnaround=0;

int i,j,n;

int at[10],bt[10],rt[10];

printf("enter totl number of processes:");

scanf("%d",&n);

remain=n;

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

{

printf("\nEnter burst time:");

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

printf("\nEnter arrival time:");

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

rt[i]=bt[i];

}

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

{

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

{

if(at[i]>at[j])

{

flip=at[i];

at[i]=at[j];

at[j]=flip;

flip=bt[i];

bt[i]=bt[j];

bt[j]=flip;

flip=rt[i];

rt[i]=rt[j];

rt[j]=flip;

}

}

}

printf("Enter time quanta:");

scanf("%d",&ts);

printf("\n\nProcess\t TURNAROUND time \tWAIT time\n\n");

for(time=0,i=0;remain!=0;)

{

if(rt[i]<=ts && rt[i]>0)

{

time+=rt[i];

rt[i]=0;

flag=1;

}

else if(rt[i]>0)

{

rt[i]-=ts;

time+=ts;

}

if(rt[i]==0 && flag==1)

{

remain--;

printf("%d\t%d\t%d\n",i+1,time-at[i],time-at[i]-bt[i]);

sum\_wait+=time-at[i]-bt[i];

sum\_turnaround+=time-at[i];

flag=0;

}

if(i==n-1)

i=0;

else if(at[i+1]<=time)

i++;

else

i=0;

}

printf("\nAvg sum\_wait = %f\n",sum\_wait/n);

printf("Avg sum\_turnaround = %f",sum\_turnaround/n);

return 0;

}

Prod cons

/\*DONE BY M.S.SANJAY 15BCE0517\*/

#include<pthread.h>

#include <stdio.h>

#include <semaphore.h>

int buffer[100];

sem\_t mutex,full,empty;

void \* producer()

{

int i=0,nextproduce,in=0;

for(i=0;i<50;i++)

{

sem\_wait(&empty);

sem\_wait(&mutex);

buffer[in]=i;

printf("\nproducer is producing item=%d",buffer[in]);

in=((in+1)%100);

sem\_post(&mutex);

sem\_post(&full);

}

}

void \* consumer()

{

int i,nextconsumed,out=0;

for(i=0;i<50;i++)

{

sem\_wait(&full);

sem\_wait(&mutex);

nextconsumed=buffer[out];

printf("\nconsumer is consuming item=%d",nextconsumed);

out=((out+1)%100);

sem\_post(&mutex);

sem\_post(&empty);

}

}

int main()

{

pthread\_t tid1;

pthread\_t tid2;

sem\_init(&mutex,0,1);

sem\_init(&full,0,0);

sem\_init(&empty,0,10);

pthread\_create(&tid1,NULL,producer,NULL);

pthread\_create(&tid2,NULL,consumer,NULL);

pthread\_join(tid1, NULL);

pthread\_join(tid2, NULL);

sem\_destroy(&mutex);

sem\_destroy(&full);

sem\_destroy(&empty);

return 0;

}

Reader writer

#include<pthread.h>

#include<semaphore.h>

#include<stdio.h>

sem\_t readermutex,writermutex;

int rc=0;

void \* writerprocess(void \*arg)

{

printf("\nwriterprocess %d is trying to enter into file",(int)arg);

sem\_wait(&writermutex);

printf("\nwriterprocess %d is writting into file",(int)arg);

printf("\nwriterprocess %d finished writing in file");

sem\_post(&writermutex);

}

void \*readerprocess(void \*arg)

{

printf("\nreaderprocess %d is trying to read the file",(int)arg);

sem\_wait(&readermutex);

rc++;

if(rc==1)

{

sem\_wait(&writermutex);

printf("\nreaderprocess %d is currently reading file",(int)arg);

}

sem\_post(&readermutex);

sem\_wait(&readermutex);

rc--;

if(rc==0)

{

printf("\nreaderprocess %d finished reading file",(int)arg);

sem\_post(&writermutex);

}

sem\_post(&readermutex);

}

int main()

{

int i=0,numread=0,numwrite;

sem\_init(&readermutex,0,1);

sem\_init(&writermutex,0,1);

pthread\_t readerprocesss\_thr[100],writerprocess\_thr[100];

printf("\nenter total no of threads for readerprocesss");

scanf("%d",&numread);

printf("\nenter total no of threads for writerprocesss");

scanf("%d",&numwrite);

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

{

pthread\_create(&readerprocesss\_thr[i],NULL,readerprocess,(void \*)i);

}

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

{

pthread\_create(&writerprocess\_thr[i],NULL,writerprocess,(void \*)i);

}

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

{

pthread\_join(writerprocess\_thr[i],NULL);

}

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

{

pthread\_join(readerprocesss\_thr[i],NULL);

}

sem\_destroy(&writermutex);

sem\_destroy(&readermutex);

return 0;

}