数组模拟短作业，队咧模拟时间片转轮，注释很清楚，就不赘述

代码：

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<time.h>

#include<set>

#include<unistd.h>

#define R "RUN" //运行中

#define F "FINISH" //已完成

#define W "WAITE" //等待中

#define T "TAKEN" //未提交

#define MAX 5

struct PCB{

char Name[10];//进程名

int Arrive;//到达时间

int Need;//需要运行时间

int Start;//开始时间

int End;//完成时间

int Turnover;//周转时间

int Priority;//优先级

int CpuTime;//所用cpu时间

float UseWeightTurnover;//带权周转时间

char Status[10];//进程状态

};

int CurrentTime;//当前时间

int finish;//已完成数量

char c; //算法选择

int visit[MAX]; //标记是否存在队列

//创建PCB

void CreatePCB(struct PCB\* pcb){

int fd\_stdin=dup(fileno(stdin));

freopen("d:\\input.txt","r",stdin);

printf("从文件中读入四个参数的数据：\n");

printf("作业号 优先级 到达时间 需要运行时间\n");

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

scanf("%s",&pcb[i].Name);//进程名

scanf("%d",&pcb[i].Priority);//优先级

scanf("%d",&pcb[i].Arrive);//到达时间

scanf("%d",&pcb[i].Need);//需要运行时间

pcb[i].Start=-1;

pcb[i].End=-1;

pcb[i].Turnover=0;

pcb[i].UseWeightTurnover=0.0;

pcb[i].CpuTime=0;

strcpy(pcb[i].Status, T);

printf("%s\t%2d\t%2d\t%5d\n",pcb[i].Name, pcb[i].Priority,pcb[i].Arrive,pcb[i].Need);

}

fclose(stdin);

fdopen(fd\_stdin,"r");

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

}

typedef struct Node

{

int data; /\*数据域\*/

struct Node \*next; /\*指针域\*/

}Node;

typedef struct

{

Node \*front;

Node \*rear;

}Queue;

void InitQueue(Queue \*q)

{

/\* 将Q初始化为一个空的链队列 \*/

q->front=(Node \*)malloc(sizeof(Node));

if(q->front!=NULL){

q->rear=q->front;

q->front->next=NULL;

}

}

/\*入队操作。\*/

void Enqueue(Queue \*q,int x)

{

/\* 将数据元素x插入到队列Q中 \*/

Node \*NewNode;

NewNode=(Node \*)malloc(sizeof(Node));

if(NewNode!=NULL){

NewNode->data=x;

NewNode->next=NULL;

q->rear->next=NewNode;

q->rear=NewNode;

}

}

/\*出队操作。\*/

int Dequeue(Queue \*q)

{

/\* 将队列Q的队头元素出队，并存放到x所指的存储空间中 \*/

Node \*p;

int x;

p=q->front->next;

q->front->next=p->next;

if(q->rear==p)

q->rear=q->front;

x=p->data;

free(p);

return x;

}

//打印

void Display(struct PCB\* pcb){

int i;

printf("当前时间为%d\n", CurrentTime);

printf("进程名 优先级 到达时间 需要运行时间 已用cpu时间 开始时间 完成时间 进程状态\n");

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

printf("%s\t%d\t%3d\t%4d\t\t%d\t%4d\t%6d\t %s\n",

pcb[i].Name,pcb[i].Priority,pcb[i].Arrive,

pcb[i].Need,pcb[i].CpuTime,pcb[i].Start, pcb[i].End,

pcb[i].Status);

}

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

}

//进程状态检测与修改

void StatusConfirm(struct PCB\* pcb){

int i;

for(i=0;i<MAX;i++){ //到达时间等于当前时间则状态变为等待中

if(CurrentTime>=pcb[i].Arrive&&strcmp(pcb[i].Status,T)==0){

strcpy(pcb[i].Status,W);

}

if(pcb[i].CpuTime==pcb[i].Need&&strcmp(pcb[i].Status,W)==0){

finish++;

strcpy(pcb[i].Status,F);

pcb[i].End=CurrentTime;

pcb[i].Turnover=pcb[i].End-pcb[i].Arrive;

pcb[i].UseWeightTurnover=pcb[i].Turnover\*1.0/pcb[i].Need;

}

}

}

//取就绪队列中运行时间最短的进程下标

int ShortIndex(struct PCB\* pcb){

int i;

int min;

int temp;

min=100;

temp=-1;

StatusConfirm(pcb);//更新进程状态

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

if(strcmp(pcb[i].Status,W)==0){

if(pcb[i].Need< min){ //有更小的运行时间则更新

min=pcb[i].Need;

temp=i;

}

}

}

return temp;

}

//比较各个进程之间的到达时间,按升序排列

void Sort(struct PCB\* pcb){

int i;

int j;

int min;

int minIndex;

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

min=pcb[i].Arrive;

minIndex=i;

for(j=i+1;j<MAX;j++){

if(pcb[j].Arrive<min){

min=pcb[j].Arrive;

minIndex = j;

}

}

struct PCB temp=pcb[i];

pcb[i]=pcb[minIndex];

pcb[minIndex]=temp;

}

}

//短作业优先

void ShortRuntime(struct PCB\* pcb)

{

Sort(pcb);

int index;

int first;

first=pcb[0].Arrive;

for(index=0,finish=0;finish!=MAX;CurrentTime++){ //进程调度位currentTime每次加1，直到进程全部被调度完成为止

StatusConfirm(pcb);

if(CurrentTime<first||ShortIndex(pcb)==-1) {//最快到达的进程还未到达或就绪队列中无进程

Display(pcb);

}

else{

index=ShortIndex(pcb);//取运行时间最短进程

pcb[index].Start=CurrentTime; //更新开始运行时间

strcpy(pcb[index].Status,R); //更新进程状态为运行中

while(true){//运行此进程

StatusConfirm(pcb);

if(pcb[index].Need==pcb[index].CpuTime){ //若所用的cpu时间等于所需运行时间

strcpy(pcb[index].Status,F); //进程状态变为已完成

finish++;

break;

}

else{

Display(pcb);

CurrentTime++;

}

pcb[index].CpuTime++;//已用时间片+1

StatusConfirm(pcb);

};

pcb[index].End=CurrentTime; //更新完成时间

pcb[index].Turnover=pcb[index].End-pcb[index].Arrive; //计算周转时间

pcb[index].UseWeightTurnover= pcb[index].Turnover\*1.0/pcb[index].Need;//计算带权周转时间

strcpy(pcb[index].Status,F);

CurrentTime--;

}

}

Display(pcb);

}

//时间片转轮法

void HighPriority(struct PCB\* pcb)

{

Queue LQ;

InitQueue(&LQ);//创建队列

Sort(pcb);//按到达时间排列

int index;

int first;

int j;

int i;

first=pcb[0].Arrive;

memset(visit,0,sizeof(visit));//标记数组置零

for(index=0,finish=0;finish!=MAX;CurrentTime++){ //进程调度位currentTime每次加1，直到进程全部被调度完成为止

StatusConfirm(pcb);

//getchar();

if(CurrentTime<first||ShortIndex(pcb)==-1) {//最快到达的进程未到达或就绪队列无进程

Display(pcb);

}

else{

if(CurrentTime==first){

Enqueue(&LQ,0);

visit[0]=1;

}

for(i=0,j=index+1;i<MAX;i++,j++){//将提交的进程入队

if(j==MAX){

j=0;

}

if(!visit[j]&&strcmp(pcb[j].Status,W)==0){

Enqueue(&LQ,j);

visit[j]=1;//已存在队列

}

}

index=Dequeue(&LQ);//取队头

visit[index]=0;//不存在队列中

if(pcb[index].Start==-1){//更新进程开始运行时间

pcb[index].Start=CurrentTime;

}

strcpy(pcb[index].Status,R);//进程状态为运行中

Display(pcb);

pcb[index].CpuTime++;//当前进程所用cpu时间增加

if(pcb[index].CpuTime==pcb[index].Need){//若所用cpu时间已达到所需运行时间

strcpy(pcb[index].Status,F);

pcb[index].End=CurrentTime+1;// 更新完成时间

pcb[index].Turnover=pcb[index].End-pcb[index].Arrive; //计算周转时间

pcb[index].UseWeightTurnover= pcb[index].Turnover\*1.0/pcb[index].Need;//计算带权周转时间

finish++;

}

else{

strcpy(pcb[index].Status,W);//时间片用完进程重新变为等待状态

}

}

}

Display(pcb);

}

//计算平均带权周转时间

float WeightTurnoverTimeCount(struct PCB\* pcb){

float sum = 0.0;

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

sum += pcb[i].UseWeightTurnover;

}

return sum /MAX;

}

//计算平均周转时间

float TurnOverTimeCount(struct PCB\* pcb){

float sum = 0.0;

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

sum += pcb[i].Turnover;

}

return sum /MAX;

}

//开始进程调度

void Start(struct PCB\* pcb){

int i;

printf("请选择进程调度算法：\n1、短作业优先调度算法。2、时间片转轮调度算法。\n\n");

scanf("%c",&c);

switch(c){

case '1':ShortRuntime(pcb);break;

case '2':HighPriority(pcb);break;

default:break;

}

printf("进程名 周转时间 带权周转时间\n");

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

printf("%s\t%d\t%.2f\n", pcb[i].Name,pcb[i].Turnover,pcb[i].UseWeightTurnover);

}

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

printf("平均周转时间为：%.2f\n",TurnOverTimeCount(pcb));

printf("平均带权周转时间为：%.2f\n", WeightTurnoverTimeCount(pcb));

}

//主函数

int main(){

struct PCB pcb[MAX];

CreatePCB(pcb);

Start(pcb);

return 0;

}