#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define WAIT "Wait"//就绪状态

#define RUN "Run"//运行状态

#define FINISH "Finish"//完成状态

#define JOBNUMBER 5 //设置进程测试数为5

typedef struct JCB{

char jobName[10];//作业名

int arriveTime;//到达时间

int runTime;//需要运行时间

int startTime;//开始时间

int endTime;//完成时间

int turnoverTime;//周转时间

float useWeightTurnoverTime;//带权周转时间

char processStatus[10];//进程状态

};

static int currentTime = 0;//当前时间

static int finishNumber = 0;//进程完成数量

char JobArray[JOBNUMBER][10];//存放数组名信息的二元数组

float priority[JOBNUMBER];//存放进程优先级的一元数组

//创建JCB

void createJCB(struct JCB\* jcb){

freopen("input.txt","r",stdin);

printf("从文件中读入三个参数的数据：\n");

printf("作业号 到达时间 需要运行时间\n");

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

scanf("%s", &jcb[i].jobName);//作业号

scanf("%d", &jcb[i].arriveTime);//到达时间

scanf("%d", &jcb[i].runTime);//需要运行时间

jcb[i].startTime = 0;

jcb[i].endTime = 0;

jcb[i].turnoverTime = 0;

jcb[i].useWeightTurnoverTime = 0.0;

strcpy(jcb[i].processStatus, WAIT);

printf("%s\t%d\t%d\n",jcb[i].jobName, jcb[i].arriveTime,jcb[i].runTime);

}

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

freopen("CON", "r", stdin);

}

//打印用途

void printJob(struct JCB\* jcb){

printf("当前时间为%d\n", currentTime);

printf("作业号 到达时间 需要运行时间 开始时间 完成时间 周转时间 带权周转时间 进程状态\n");

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

if(strcmp(jcb[i].processStatus, FINISH) == 0)//如果进程为finish状态，这样输出

printf("%s\t%d\t%4d\t\t%d\t%d\t %d\t %.2f\t %s\n", jcb[i].jobName, jcb[i].arriveTime, jcb[i].runTime, jcb[i].startTime, jcb[i].endTime, jcb[i].turnoverTime, jcb[i].useWeightTurnoverTime, jcb[i].processStatus);

else if(strcmp(jcb[i].processStatus, RUN) == 0)//如果进程为run状态，这样输出

printf("%s\t%d\t%4d\t\t%d\t运行中\t none\t none %s\n", jcb[i].jobName, jcb[i].arriveTime, jcb[i].runTime, jcb[i].startTime, jcb[i].processStatus);

else //如果进程为wait状态，这样输出

printf("%s\t%d\t%4d\t\t未运行\tnone\t none\t none %s\n", jcb[i].jobName, jcb[i].arriveTime, jcb[i].runTime, jcb[i].processStatus);

}

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

}

//计算平均带权周转时间

float weightTurnoverTimeCount(struct JCB\* jcb){

float sum = 0.0;

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

sum += jcb[i].useWeightTurnoverTime;

return sum / JOBNUMBER;

}

//计算平均周转时间

float turnOverTimeCount(struct JCB\* jcb){

float sum = 0.0;

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

sum += jcb[i].turnoverTime;

return sum / JOBNUMBER;

}

//比较各个进程之间的到达时间,按升序排列

void compare(struct JCB\* jcb){

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

int min = jcb[i].arriveTime, minIndex = i;

for(int j = i + 1; j < JOBNUMBER; j++){

if(jcb[j].arriveTime < min){

min = jcb[j].arriveTime;

minIndex = j;

}

}

struct JCB temp = jcb[i];

jcb[i] = jcb[minIndex];

jcb[minIndex] = temp;

}

}

//打印进程调度顺序，平均周转时间及平均带权周转时间

void printInfo(struct JCB\* jcb){

printf("1、进程调度顺序为：%s -> %s -> %s -> %s -> %s\n", JobArray[0], JobArray[1], JobArray[2], JobArray[3], JobArray[4]);

printf("2、平均周转时间为：%.2f\n",turnOverTimeCount(jcb));

printf("3、平均带权周转时间为：%.2f\n", weightTurnoverTimeCount(jcb));

printf("------------------测试完毕 版权归邓钦艺所有---------\n");

}

//两算法共同循环遍历部分

void loop(struct JCB\* jcb, int i){

jcb[i].startTime = currentTime;

jcb[i].endTime = jcb[i].startTime + jcb[i].runTime;

jcb[i].turnoverTime = jcb[i].endTime - jcb[i].arriveTime;

jcb[i].useWeightTurnoverTime = jcb[i].turnoverTime \* 1.0 / jcb[i].runTime;

strcpy(jcb[i].processStatus, RUN);

while(true){

if(currentTime == jcb[i].endTime){

strcpy(jcb[i].processStatus, FINISH);

finishNumber++;

if(finishNumber == JOBNUMBER)

printJob(jcb);

currentTime--;

break;

}

else{

printJob(jcb);

currentTime++;

}

}

}

//先来先服务调度算法

void firstComeFirstServed(struct JCB\* jcb){

createJCB(jcb);

compare(jcb);

int i = 0;

//进程调度currentTime每次加1，直到进程全部被调度完成为止

for(; finishNumber != JOBNUMBER; currentTime++){

if(currentTime < jcb[0].arriveTime)//当前时间小于第一个节点到来时间时，直接打印

printJob(jcb);

else{

strcpy(JobArray[i], jcb[i].jobName);

loop(jcb, i);

i++;

}

}

printInfo(jcb);//打印进程调度顺序，平均周转时间及平均带权周转时间

currentTime = 0;//静态变量当前时间置位

finishNumber = 0;//静态变量完成进程数量置位

}

//高响应比优先调度算法

void highestResponseRatioNext(struct JCB\* jcb){

createJCB(jcb);

compare(jcb);

int i = 0, j = 0;

for(; finishNumber != JOBNUMBER; currentTime++){

float maxPriority = 0.0;

int indexPriority = 0;

if(currentTime < jcb[0].arriveTime)//当前时间小于第一个节点到来时间时，直接打印

printJob(jcb);

else{

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

if(strcmp(jcb[i].processStatus, FINISH) != 0){

int waitTime = currentTime - jcb[i].arriveTime;

priority[i] = (waitTime + jcb[i].runTime) \* 1.0 / jcb[i].runTime;

if(priority[i] > maxPriority){

maxPriority = priority[i];

indexPriority = i;

}

}

}

strcpy(JobArray[j++], jcb[indexPriority].jobName);

loop(jcb, indexPriority);

}

}

printInfo(jcb);//打印进程调度顺序，平均周转时间及平均带权周转时间

currentTime = 0;//当前时间置位

finishNumber = 0;//完成进程数量置位

}

//菜单函数

void menu(struct JCB\* jcb){

int input;

while(true){

printf("------------3114005847 邓钦艺-----------------\n");

printf("| 1、先来先服务调度算法 |\n");

printf("| 2、响应比高者优先调度算法 |\n");

printf("| 3、退出 |\n");

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

printf("请输入序号以继续程序：");

scanf("%d", &input);

switch(input){

case 1:firstComeFirstServed(jcb);

break;

case 2:highestResponseRatioNext(jcb);

break;

case 3:

exit(0);

default:printf("输入有误，请重新输入!!!\n");

break;

}

}

}

//主函数

int main(){

struct JCB jcb[JOBNUMBER];

menu(jcb);

system("pause");

return 0;

}