

**操作系统课程设计报告**

**系 （院）：**  计算机科学学院

**专业班级：** 计科11402

**姓 名：** 程志平

**学 号：** 201403616

**指导教师：**  **钟宝荣**

**设计时间：** 2017.6.12 - 2017.6.23

**设计地点：** 4教软件实验室

目录

[一、课程设计目的 1](#_Toc486453293)

[二、课程设计的任务和要求 1](#_Toc486453294)

[三、模拟程序的描述 2](#_Toc486453295)

[四、需求分析 3](#_Toc486453296)

[五、总体设计 3](#_Toc486453297)

[六、详细设计与实现 4](#_Toc486453298)

[代码部分 4](#_Toc486453299)

[指令类Instructions 4](#_Toc486453300)

[进程类PCBs 4](#_Toc486453301)

[文件读取类fileReader 5](#_Toc486453302)

[主题类mainFrame 7](#_Toc486453303)

[结果展示部分 17](#_Toc486453304)

[窗体界面 17](#_Toc486453305)

[读取文件并设置 18](#_Toc486453306)

[开始调度及结果 18](#_Toc486453307)

[七、课程设计小结 19](#_Toc486453308)

**《操作系统》课程设计**

--进程调度模拟程序

# 一、课程设计目的

《操作系统原理》是计算机科学与技术专业的一门专业核心课程，也是研究生入学考试中计算机专业综合中所涉及的内容。该课程理论性强，纯粹的理论学习相对枯燥乏味，不易理解。通过课程设计，可加强学生对原理知识的理解。

# 二、课程设计的任务和要求

本次课程设计的题目是，**时间片轮转调度算法的模拟实现**。要求在充分理解时间片轮转调度算法原理的基础上，编写一个可视化的算法模拟程序。

具体任务如下：

1、根据需要，合理设计PCB结构，以适用于时间片轮转调度算法；

2、设计模拟指令格式，并以文件形式存储，程序能够读取文件并自动生成指令序列。

3、根据文件内容，建立模拟进程队列，并能采用时间片轮转调度算法对模拟进程进行调度。

任务要求：

1、进程的个数，进程的内容（即进程的功能序列）来源于一个进程序列描述文件。

2、需将调度过程输出到一个运行日志文件。

3、开发平台及语言不限。

4、要求设计一个Windows可视化应用程序。

# 三、模拟程序的描述

**模拟指令的格式：操作命令+操作时间**

● C ： 表示在CPU上计算

● I ： 表示输入

● O ： 表示输出

● W ： 表示等待

● H ： 表示进程结束

操作时间代表该操作命令要执行多长时间。这里假设I/O设备的数量没有限制，I和O设备都只有一类。

I，O，W三条指令实际上是不占有CPU的，执行这三条指令就应该将进程放入对应的等待队列（输入等待队列，输出等待队列 ，其他等待队列）。

例如，有一虚拟程序文件prc.txt描述如下：

P1

C10

I20

C40

I30

C20

O30

H00

P2

I10

C50

O20

H00

P3

C10

I20

W20

C40

O10

H00................

# 四、需求分析

程序要求实现时间片轮转调度算法

1. 从本地读取文本文件，文件中有进程名和指令队列，队列含有每条指令的类型和执行要用的时间片大小。
2. 读取数据时按照读取的内容进行区分。
3. 需要设计合理的进程pcb结构，和指令结构，来进行合理的时间调度。
4. 按照指令的类型，构造相应的队列，存放对应的指令。
5. 严格遵循时间片轮转调度算法进行程序的设计和实现。
6. 将进程的调度过程输出，并写出简要文字描述。

# 五、总体设计

※时间片大小固定，由用户手动输入。

※每个进程用一个pcb表示。Pcb包括进程名，指令列表，当前运行指令。

※每个指令用Instrcation实现。Instructions包括指令类型，指令运行时间，指令剩余时间。

※根据指令类型，创建相应的指令队列，分别为：AllQue，ReadyQue，BackupReadyQue，InputQue，OutputQue，OtherQue。

※读取文件时，按行读取，当第一个字母为P时，创建相应的进程对象，并将后面的指令添加到他的队列中去

※按照时间片轮转的规律进程调度



# 六、详细设计与实现

## 代码部分

### 指令类Instructions

**package** project;

**public** **class** Instrustions

{

**public** String IName; //指令类型

**public** **int** IRuntime; //指令运行时间

**public** **int** IRemainTime; //指令剩余运行时间

}

### 进程类PCBs

**package** project;

**import** java.awt.List;

**import** java.util.ArrayList;

**public** **class** PCBs

{

**public** String PName; //进程名称

**public** ArrayList<Instrustions> PInstrustions; //进程中的指令列表

**public** **int** CurrentInstruction; //当前运行指令索引

**public** PCBs()

{

PInstrustions = **new** ArrayList<Instrustions>();

}

}

### 文件读取类fileReader

package project;

import javax.swing.JFileChooser;

import javax.swing.filechooser.FileFilter;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileReader;

import java.io.IOException;

import java.io.InputStreamReader;

import javax.swing.\*;

import org.omg.CORBA.PUBLIC\_MEMBER;

public class fileReader { //文件读取

JFileChooser chooser;

String filename;

FileReader fileread;

BufferedReader bufferread;

public void read()

{

chooser=new JFileChooser();

chooser.setDialogTitle("请选择文件");

chooser.showOpenDialog(null);

chooser.setVisible(true);

filename=chooser.getSelectedFile().getAbsolutePath();

fileread=null;

bufferread=null;

try

{

fileread=new FileReader(filename);

bufferread=new BufferedReader(fileread);

String s="";

String all="";

while ((s = bufferread.readLine())!= null) {

all = s.substring(0, 1);switch (all) {

case "P":

PCBs p1 = new PCBs();

p1.PName = s;

Queens.AllQue.add(p1);

break;

case "C":

Instrustions a = new Instrustions();

a.IName = s.substring(0, 1);

a.IRemainTime = Integer.parseInt(s.substring(1, 3));

PCBs pcb1 = (PCBs)Queens.AllQue.get(Queens.AllQue.size() - 1);

pcb1.PInstrustions.add(a);

break;

case "I":

Instrustions b = new Instrustions();

b.IName = s.substring(0, 1);

b.IRemainTime = Integer.parseInt(s.substring(1, 3));

PCBs pcb2 = (PCBs) Queens.AllQue.get(Queens.AllQue.size() - 1);

pcb2.PInstrustions.add(b);

break;

case "O":

Instrustions c = new Instrustions();

c.IName = s.substring(0, 1);

c.IRemainTime = Integer.parseInt(s.substring(1, 3));

PCBs pcb3 = (PCBs) Queens.AllQue.get(Queens.AllQue.size() - 1);

pcb3.PInstrustions.add(c);

break;

case "W":

Instrustions d = new Instrustions();

d.IName = s.substring(0, 1);

d.IRemainTime = Integer.parseInt(s.substring(1, 3));

PCBs pcb4 = (PCBs) Queens.AllQue.get(Queens.AllQue.size() - 1);

pcb4.PInstrustions.add(d);

break;

case "H":

Instrustions e = new Instrustions();

e.IName = s.substring(0, 1);

e.IRemainTime = Integer.parseInt(s.substring(1, 3));

PCBs pcb5 = (PCBs) Queens.AllQue.get(Queens.AllQue.size() - 1);

pcb5.PInstrustions.add(e);

default:

break;

}

}

String allq="";

for(int i=0;i<Queens.AllQue.size();i++)

{

PCBs pcb=(PCBs)Queens.AllQue.get(i);

allq+=pcb.PName+"\r\n";

}

mainFrame.areaReadlyList.setText(allq);

}

catch(Exception e)

{

e.printStackTrace();

}

finally

{

try

{

bufferread.close();

fileread.close();

}

catch(Exception ee)

{

ee.printStackTrace();

}

}

}

}

### 主题类mainFrame

package project;

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

public class mainFrame implements ActionListener

{

Boolean Stoping = false;

int timePiece;

JFrame frame; //JFrame Panel创建

JPanel panel;

JButton buttonOpen;

JButton buttonStart;

JButton buttonStop; //button创建

JLabel labelTimePiece;

JLabel labelRunningPcb;

JLabel labelReadlyList;

JLabel labelStoredReadlyList;

JLabel labelInputWaitList;

JLabel labelOutputWaitList;

JLabel labelOtherWaitList; //label创建

JTextField textTimePiece;

JTextField textRunningPcb; //textfield创建

public static JTextArea areaReadlyList;

JTextArea areaStoredReadlyList;

JTextArea areaInputWaitList;

JTextArea areaOutputWaitList;

JTextArea areaOtherWaitList; //textarea创建

public mainFrame()

{

frame=new JFrame("时间片轮转算法");

panel=new JPanel();

frame.setContentPane(panel);

buttonOpen=new JButton("打开文件");

buttonStart=new JButton("开始调度");

buttonStop=new JButton("停止调度");

labelTimePiece=new JLabel("时间片大小:");

labelRunningPcb=new JLabel("当前运行进程:");

labelReadlyList=new JLabel("就 绪 队 列");

labelStoredReadlyList=new JLabel(" 后备就绪队列");

labelInputWaitList=new JLabel("输入等待队列");

labelOutputWaitList=new JLabel("输出等待队列");

labelOtherWaitList=new JLabel("其 他 队 列");

textTimePiece=new JTextField();

textRunningPcb=new JTextField();

areaReadlyList=new JTextArea();

areaStoredReadlyList=new JTextArea();

areaInputWaitList=new JTextArea();

areaOutputWaitList=new JTextArea();

areaOtherWaitList=new JTextArea();

//frame.add(panel);

panel.add(textTimePiece);

panel.add(textRunningPcb);

panel.add(buttonOpen);

panel.add(buttonStart);

panel.add(buttonStop);

//buttonStart.setEnabled(false);// 初始化开始键不可按

panel.add(areaInputWaitList);

panel.add(areaOtherWaitList);

panel.add(areaOutputWaitList);

panel.add(areaReadlyList);

panel.add(areaStoredReadlyList);

panel.add(labelInputWaitList);

panel.add(labelTimePiece);

panel.add(labelRunningPcb);

panel.add(labelReadlyList);

panel.add(labelStoredReadlyList);

panel.add(labelOutputWaitList);

panel.add(labelOtherWaitList);

panel.setLayout(null);

frame.setVisible(true);

frame.setSize(750,500);

buttonOpen.setBounds(40,20, 90, 30);

buttonStart.setBounds(150,20, 90, 30);

buttonStop.setBounds(260,20, 90, 30);

labelTimePiece.setBounds(425, 20, 120, 30);

labelTimePiece.setFont(new Font("Dialog",1,18));

textTimePiece.setBounds(560, 20, 90, 30);

labelRunningPcb.setBounds(228, 80, 140, 30);

labelRunningPcb.setFont(new Font("Dialog",1,18));

textRunningPcb.setBounds(373, 80, 100, 30);

labelReadlyList.setBounds(40,150,100,30);

labelReadlyList.setFont(new Font("Dialog",1,15));

labelStoredReadlyList.setBounds(175, 150, 100, 30);

labelStoredReadlyList.setFont(new Font("Dialog",1,15));

labelInputWaitList.setBounds(305, 150, 100, 30);

labelInputWaitList.setFont(new Font("Dialog",1,15));

labelOutputWaitList.setBounds(435, 150, 100, 30);

labelOutputWaitList.setFont(new Font("Dialog",1,15));

labelOtherWaitList.setBounds(565, 150, 100, 30);

labelOtherWaitList.setFont(new Font("Dialog",1,15)); ////五个标签

areaReadlyList.setBounds(40,180, 100, 250);

areaStoredReadlyList.setBounds(175, 180, 100, 250);

areaInputWaitList.setBounds(305,180, 100, 250);

areaOutputWaitList.setBounds(435, 180, 100, 250);

areaOtherWaitList.setBounds(565,180, 100, 250); //五个区域

buttonOpen.addActionListener(this);

buttonStop.addActionListener(this);

buttonStart.addActionListener(this);

}

public static void main(String[] args)

{

mainFrame jjj=new mainFrame();

}

public void actionPerformed(ActionEvent e) {

if(e.getSource()==buttonOpen)

{

fileReader fff=new fileReader();

fff.read();

}

if (e.getSource()==buttonStop) {

if(Stoping == false)

{

Stoping = true; //若为暂停是将布尔类型的Stop变为true

}

else

{

Stoping = false;

}

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*" +Stoping);

}

if(e.getSource()==buttonStart)

{

if (e.getSource()==buttonStart) {

areaReadlyList.setText("");

timePiece=Integer.parseInt(textTimePiece.getText()); // 获取输入的时间片大小

process();

new Thread(new Runnable() {

public void run() {

while(true){

try {

for(;Stoping == true;){ //无限循环，一直判断布尔量stop的值

System.out.println("调度暂停！！！！"); //若为false则一直输出暂停，直到stop的

//的值变为false。

}

Thread.sleep(timePiece); //线程休眠时间片大小的时间

processBackupReadyQue();

processInputQue();

processOutputQue();

processWaitQue();

showView();

} catch (Exception e) {

}

}

}

}).start();

}

}

}

public void processBackupReadyQue() //后备就绪队列

{

if(!Queens.BackupReadyQue.isEmpty()){ //若后备队列不为空

PCBs pcb = (PCBs) Queens.BackupReadyQue.get(0); //获取第一个进程

System.out.println("后备就绪进程名"+pcb.PName); //输出此进程的名字

Instrustions instructions = (Instrustions) pcb.PInstrustions.get(0);//进程队列的第一个命令

instructions.IRemainTime-=1; //需要时间减1

if(instructions.IRemainTime>0){ //若还需要的时间大于0时

Queens.BackupReadyQue.remove(0); //移除此进程

distribution(pcb); //根据指令类型进行操作

System.out.println("-时间片用完-从后备就绪队列中移除"+pcb.PName);

}else{

pcb.PInstrustions.remove(0); //若指令还需时间小于0

Queens.BackupReadyQue.remove(0);

distribution(pcb);

}

}

}

public void outputView2(){ //打印后备就绪队列

if(!Queens.BackupReadyQue.isEmpty()){

String allque = "";

for (int m = 0; m <Queens.BackupReadyQue.size(); m++) {

PCBs pcb1 = (PCBs) Queens.BackupReadyQue.get(m);

allque = allque + pcb1.PName + "\r\n";

}

System.out.println("后备就绪队列中有" + allque);

textRunningPcb.setText(allque.substring(0, 2));

areaStoredReadlyList.setText(allque.substring(0, allque.length()));

}else{

areaStoredReadlyList.setText(null);

}

}

public void processInputQue()

{

if(!Queens.InputQue.isEmpty()){ //输入队列不为空时，获取进程名

PCBs pcb = (PCBs) Queens.InputQue.get(0);

Instrustions instructions = (Instrustions) pcb.PInstrustions.get(0);//进程中指令队列的第一个指令

instructions.IRemainTime-=1;

if(instructions.IRemainTime<=0){

System.out.println("移除指令类型" + ((Instrustions)pcb.PInstrustions.get(0)).IName);

pcb.PInstrustions.remove(0); //时间小于0指令从指令列表中移除

System.out.println("输入队列移除" + ((Instrustions)pcb.PInstrustions.get(0)).IName);

Queens.InputQue.remove(0);// 时间小于0指令从输入队列移除

System.out.println("命令类型" + ((Instrustions)pcb.PInstrustions.get(0)).IName);

distribution(pcb);

}

}

}

public void outputView3(){ //打印输入队列

if(!Queens.InputQue.isEmpty()){

String allque = "";

for (int m = 0; m <Queens.InputQue.size(); m++) {

PCBs pcb1 = (PCBs) Queens.InputQue.get(m);

allque = allque + pcb1.PName + "\r\n";

}

areaInputWaitList.setText(allque);

System.out.println("输入队列中有" + allque);

}else{

areaInputWaitList.setText(null);

}

}

public void processOutputQue() //输出队列管理

{

if(!Queens.OutputQue.isEmpty())

{

PCBs pcb = (PCBs) Queens.OutputQue.get(0);

Instrustions instructions = (Instrustions) pcb.PInstrustions.get(0);

instructions.IRemainTime-=1;

if(instructions.IRemainTime<=0)

{

pcb.PInstrustions.remove(0);

Queens.OutputQue.remove(0);

distribution(pcb);

}

}

}

public void outputView4(){ //打印输出队列

if(!Queens.OutputQue.isEmpty()){

String allque = "";

for (int m = 0; m <Queens.OutputQue.size(); m++) {

PCBs pcb1 = (PCBs)Queens.OutputQue.get(m);

allque = allque + pcb1.PName + "\r\n";

}

areaOutputWaitList.setText(allque);

System.out.println("输出队列中有" + allque);

}else{

areaOutputWaitList.setText(null);

}

}

public void outputView5(){ //打印等待队列

if(!Queens.WaitQue.isEmpty()){

String allque = "";

for (int m = 0; m <Queens.WaitQue.size(); m++) {

PCBs pcb1 = (PCBs) Queens.WaitQue.get(m);

allque = allque + pcb1.PName + "\r\n";

}

areaOtherWaitList.setText(allque);

System.out.println("等待队列中有" + allque);

}else{

areaOtherWaitList.setText(null);

}

}

public void processWaitQue() //等待队列

{

if(!Queens.WaitQue.isEmpty()){

PCBs pcb = (PCBs) Queens.WaitQue.get(0);

Instrustions instructions = (Instrustions) pcb.PInstrustions.get(0);

instructions.IRemainTime-=1;

if(instructions.IRemainTime<=0){

pcb.PInstrustions.remove(0);

Queens.WaitQue.remove(0);

distribution(pcb);

}

}

}

public void showAllque(int index){ //输出所有队列名

if(!Queens.AllQue.isEmpty()){

String allque = "";

for (int m = index; m <Queens.AllQue.size(); m++) {

PCBs pcb1 = (PCBs)Queens.AllQue.get(m);

allque = allque + pcb1.PName + "\r\n";

}

areaReadlyList.setText(allque);

System.out.println("所有队列中有" + allque);

}else{

areaReadlyList.setText(null);

}

}

public void showView(){ //所有队列的输出

outputView2();

outputView3();

outputView4();

outputView5();

}

public void distribution(PCBs pcb) //按照指令名进行处置

{

Instrustions instructions = (Instrustions) pcb.PInstrustions.get(0);

if (instructions.IName.equals("C")) { //按照指令类型进行操作

Queens.BackupReadyQue.add(pcb);

}else if(instructions.IName.equals("I")){

Queens.InputQue.add(pcb);

}else if(instructions.IName.equals("O")){

Queens.OutputQue.add(pcb);

}else if(instructions.IName.equals("W")){

Queens.WaitQue.add(pcb);

}else if(instructions.IName.equals("H")) //若为H，表示此次进程结束

{

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*" +pcb.PName +"已完成工作"+ "\*\*\*\*\*\*\*\*\*\*\*");

}

}

public void process(){

new Thread(new Runnable() {

int i;

public void run() {

for (i = 0; i <Queens.AllQue.size(); i++)

{

try {

PCBs pcb = (PCBs) Queens.AllQue.get(i);

Instrustions instructions = (Instrustions) pcb.PInstrustions.get(0);

if (instructions.IName.equals("C")) {

textRunningPcb.setText(pcb.PName);

Queens.BackupReadyQue.add(pcb);

//showView();

}

if (instructions.IName.equals("I")) {

Queens.InputQue.add(pcb);

// showView();

}

if (instructions.IName.equals("O")) {

Queens.OutputQue.add(pcb);

// showView();

}

if (instructions.IName.equals("W")) {

Queens.WaitQue.add(pcb);

// showView();

}

if(i<Queens.AllQue.size()-1){

showAllque(i);

showView();

}else{

areaReadlyList.setText(null);

}

Thread.sleep(timePiece);

} catch (InterruptedException e) {

// TODO 自动生成的 catch 块

e.printStackTrace();

}

}

}

}).start();

}

}

## 结果展示部分

### 窗体界面

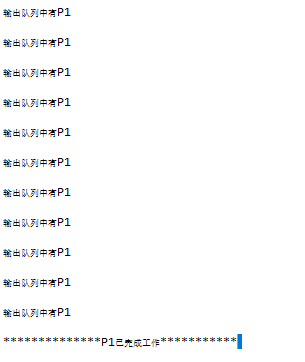


### 读取文件并设置



### 开始调度及结果





# 七、课程设计小结

通过这次的课程设计，我对进程调度算法，特别对轮转调度算法有了更深的认识。在学期初刚接触操作系统这门课程时，其实是觉得比较难找到切入点的，更不用谈里面的各种各样的算法。不过后来经过了思考，查看各类资料，摸清调度算法的各项步骤，其实自己都可以用一句话把算法概况。

这次课程设计的语言采用了java此门语言毕竟是大三上学期学的，离现在时间不多，所以加以深用起来会稍显吃力，不过凭着啃硬骨头的精神，还有以前遗留下来的编程基础，通过查阅教科书还是没有任何问题的。

学如逆水行舟，不进则退。课程设计是一次让我重新检查自己所学过的知识是否还熟练，是否有所缺漏的机会，并且只要通过自己动手去排除各种困难，最后自己还是获益匪浅的。

成绩: 教师签名：

年 月 日