**华南农业大学信息（软件）学院**

《操作系统分析与设计实习》成绩单

开设时间：2014学年第一学期

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 小组成员、组内分工及各成员成绩 | | | | | | | | |
| **学号** | 201230740312 | **姓名** | 刘杜康 | **分工** | 全部 | | **成绩** |  |
| **学号** |  | **姓名** |  | **分工** |  | | **成绩** |  |
| **学号** |  | **姓名** |  | **分工** |  | | **成绩** |  |
| 实验题目 | 题目四 单处理器系统的进程调度 | | | | | | | |
| 自我评价 | 这次的操作系统课程设计，根据题目难度的逐级递增，我选择的题目是单处理器系统的进程调度。分析完题目之后，感觉难度一般，思路也很清晰，事实上写代码花了2天多。在实现过程中，遇到比较复杂的问题就是在JTable单元格中实现JProgressBar的动态变化。还有就是对进程模拟过程中继续添加新进程后的进程处理，为了实现这个功能，我几乎把整个程序的框架都重构了，因为一开始没考虑到这一点。最后，觉得这次课程设计做完之后自己的收获还是挺大的，进一步加深了我对Java多线程的认识和操作系统中进程调度的理解。 | | | | | | | |
| 教师评语 | 评价指标：   * 题目内容和要求完成情况 优□ 良□ 中□ 差□ * 对算法原理的理解程度 优□ 良□ 中□ 差□ * 程序设计水平 优□ 良□ 中□ 差□ * 程序运行效果及正确性 优□ 良□ 中□ 差□ * 课程设计报告结构清晰 优□ 良□ 中□ 差□ * 报告中总结和分析详尽 优□ 良□ 中□ 差□ | | | | | | | |
|  |  | | | **教师签名** | | 张丽霞 | | |

# 一、 需求分析

编写程序完成单处理器系统中的进程调度， 要求实现时间片轮转、 优先数、 最短进程优先和最短剩余时间优先四种调度算法。 实验具体包括： 首先确定进程控制块的内容， 进程控制块的组成方式； 然后完成进程创建原语和进程调度原语； 最后编写主函数对所作工作进行测试。

模拟程序只对你所设置的“ 虚拟 PCB” 进行相应的调度模拟操作， 即每发生“ 调度”时， 显示出当前运行进程的“ 进程标识符”、“ 优先数”、“ 剩余运行时间” 等， 而不需要对系统中真正的 PCB 等数据进行修改。

主要考虑三个问题： 如何组织进程、 如何创建进程和如何实现处理器调度。

题目要求模拟实现四种调度方法， 要求和提示如下：

（ 1） 时间片轮转调度

时间片轮转调度的要求和提示已经在上述过程中进行了说明， 请参看。

（ 2） 优先数调度

要求动态改变优先数， 假设大数代表高优先级， 进程每运行一次优先数就减“ 1”， 即被调度时执行： 优先数－ 1， 剩余运行时间－ 1， 来模拟进程的一次运行。

进程运行一次后， 若剩余运行时间不等于 0， 则再将它加入队列（ 按优先数大小插入，

且置队首标志）； 若剩余运行时间等于 0， 则把它的状态修改成“终止”， 且退出队列。

若就绪进程队列不为空， 则重复调度， 直到所有进程都“ 终止”。

（ 3） 最短进程优先

按照进程执行时间的长短进行排队， 优先调度短进程， 因为该调度不抢占， 因此调度到

的进程就可以运行完， 其状态修改成“终止”， 且退出队列。若就绪进程队列不为空， 则重复调度， 直到所有进程都“ 终止”。

（ 4） 最短剩余时间优先

最短进程优先的抢占版本， 比较当前进程的剩余运行时间和新到达进程的预计运行时

间， 短者优先执行。 进程运行一次后， 若剩余运行时间不等于 0， 则再将它加入队列（按剩余时间长短插入， 且置队首标志）； 若剩余运行时间等于 0， 则把它的状态修改成“ 终止”，且退出队列。若就绪进程队列不为空， 则重复调度， 直到所有进程都“ 终止”。

# 二、概要设计

**1、主要数据结构**

public ArrayList<PCB> arraylist = new ArrayList<PCB>();保存随机创建的每个进程信息

public ArrayList<PCB> resetArraylist = new ArrayList<PCB>(); 存档随机创建的每个进程信息

，用于复位功能

private Set<Integer> set = new HashSet<Integer>();保存每个进程标识符，确保不会重复

Queue<PCB>[] queue = new LinkedList[5];按优先级保存要执行的进程

PCB结构：

public class PCB {

int name; // 进程标识符

int status; // 进程状态

int pri; // 进程优先数

int total;// 进程总时间

int time; // 剩余运行时间， 以时间片为单位， 当减至 0 时该进程终止

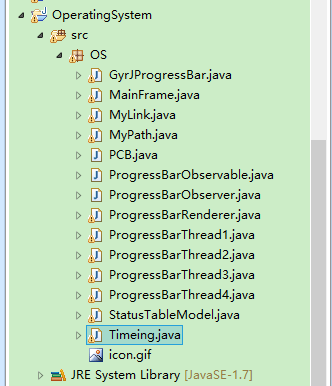
int next; // 下一个进程控制块的位置

int psw;//状态字

JProgressBar bar;//进度条

}

**2、程序的类说明：**



GyrJProgressBar用于创建随百分百变化产生颜色变化的进度条

MainFrame程序主界面

MyLink用于支持URL跳转

MyPath一个空白JPanel

PCB进程控制块

ProgressBarObservable用于实现观察者观察的对象，进度条的显示值

ProgressBarObserver用于实现观察者，进度条的显示值变化时更新statusTableModel

ProgressBarRenderer进度条的渲染器

ProgressBarThread1时间片轮转调度实现类

ProgressBarThread2优先数调度实现类

ProgressBarThread3最短进程优先实现类

ProgressBarThread4最短剩余时间优先实现类

StatusTableModel状态区表格模型类

Timeing运行时间实现类

icon.gif一个图标

**三、 详细设计**

**MainFrame：**

public static void main(String[] args) throws Exception {

//程序的入口方法

//调用MainFrame()方法创建程序的窗体

}

public MainFrame() throws Exception {

//添加程序窗体的各个按钮、输入框等

//添加监听事件等

}

void createRamdonProcess(int number){

//用于根据输入的个数number生成随机进程，并添加到列表中

while (count < numberCount) {

// 创建进程标识符

}

for (i = 1; i <= numberCount; i++) {

PCB pcb = new PCB(array\_processName[i], 1, processPri, processTime,

processTime, new JProgressBar());

PCB pcb2 = new PCB(array\_processName[i], 1, processPri, processTime,

processTime);

statusTableModel.addProgressBar(object[i]);//设置列表的一行内容

//渲染进度条

if (!firstRun) {

//不是第一次运行创建进程，则将刚刚创建的进程添加到运行中的线程中

}

}

}

void isNum(String f) {

//判断用户输入的内容是否为正数

}

private void resetRow() {

//重置列表内容实现方法

}

private void clearRow() {

//清空列表内容实现方法

}

**ProgressBarThread1时间片轮转调度实现类:**

public ProgressBarThread1(JTable statusTable, ArrayList<PCB> arraylist) {

//时间片轮转调度算法构造函数

}

public void run() {

//线程初始化参数

while (end == 0) {

while (arraylist.get(number).time != 0) {

//跟新JTable列表的内容

try {

Thread.sleep(1000);

} catch (InterruptedException ex) {

System.out

.println("InterruptedException in ProgressBarThread");

}

if (arraylist.get(number).time != 0) {

statusTable.setValueAt("就绪态", number, 1);

} else {

statusTable.setValueAt("终止态", number, 1);

}

number++;

if (number >= arraylist.size()) {

// System.out.println("number reset=" + number);

number = 0;

}

}

stopCount = 0;

while (arraylist.get(number).time == 0) {

//检查是否全部进程的剩余时间都为0

}

}

}

}

}

其他3个调度算法，ProgressBarThread2优先数调度实现类、ProgressBarThread3最短进程优先实现类、ProgressBarThread4最短剩余时间优先实现类的结构与上面的类似，不再累赘。

**四、 调试分析**

在实现过程中，遇到比较复杂的问题就是在JTable单元格中实现JProgressBar的动态变化，最后通过添加三个类ProgressBarObservable（用于实现观察者观察的对象，进度条的显示值）ProgressBarObserver（用于实现观察者，进度条的显示值变化时更新statusTableModel）

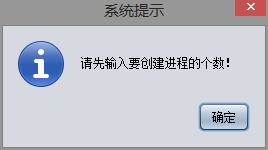
ProgressBarRenderer（进度条的渲染器）解决了该问题。还有就是对进程模拟过程中继续添加新进程后的进程处理，为了实现这个功能，我几乎把整个程序的框架都重构了，因为一开始没考虑到这一点。

**五、 用户使用说明**

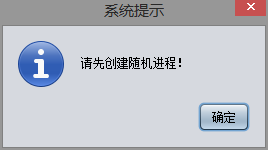


1. 在上图标号1处输入要创建随机进程的个数，仅可输入正数，非正数会有相关提示。然后点击标号2处的“创建进程”按钮，随进创建的进程显示在程序界面的中央窗口，如标号3所示。
2. 创建好随机进程后，在标号4的单选框选择将要模拟执行的调度算法，然后点击标号5处的“开始模拟”，程序开始执行。标号3的列表会显示相应的调度变化。
3. 模拟过程中，可以继续添加新的进程，操作同上。
4. 一个算法模拟执行完毕之后，可以点击标号6的“复位”按钮，可以重置列表的内容为程序模拟运行前的内容。复位成功后，可以继续选择其他调度算法进行模拟。
5. 标号7显示为程序模拟过程中的时间，从1秒开始累计。
6. 点击标号8的“清空”按钮，可以清空类别的进程，以便程序的下次执行。

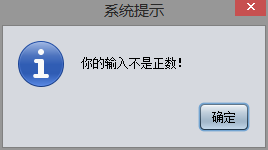
**六、 测试与运行结果**



一些提示框



一些提示框



一些提示框



按时间片轮转调度



按优先数调度



按最短进程优先调度



按最短剩余时间优先调度