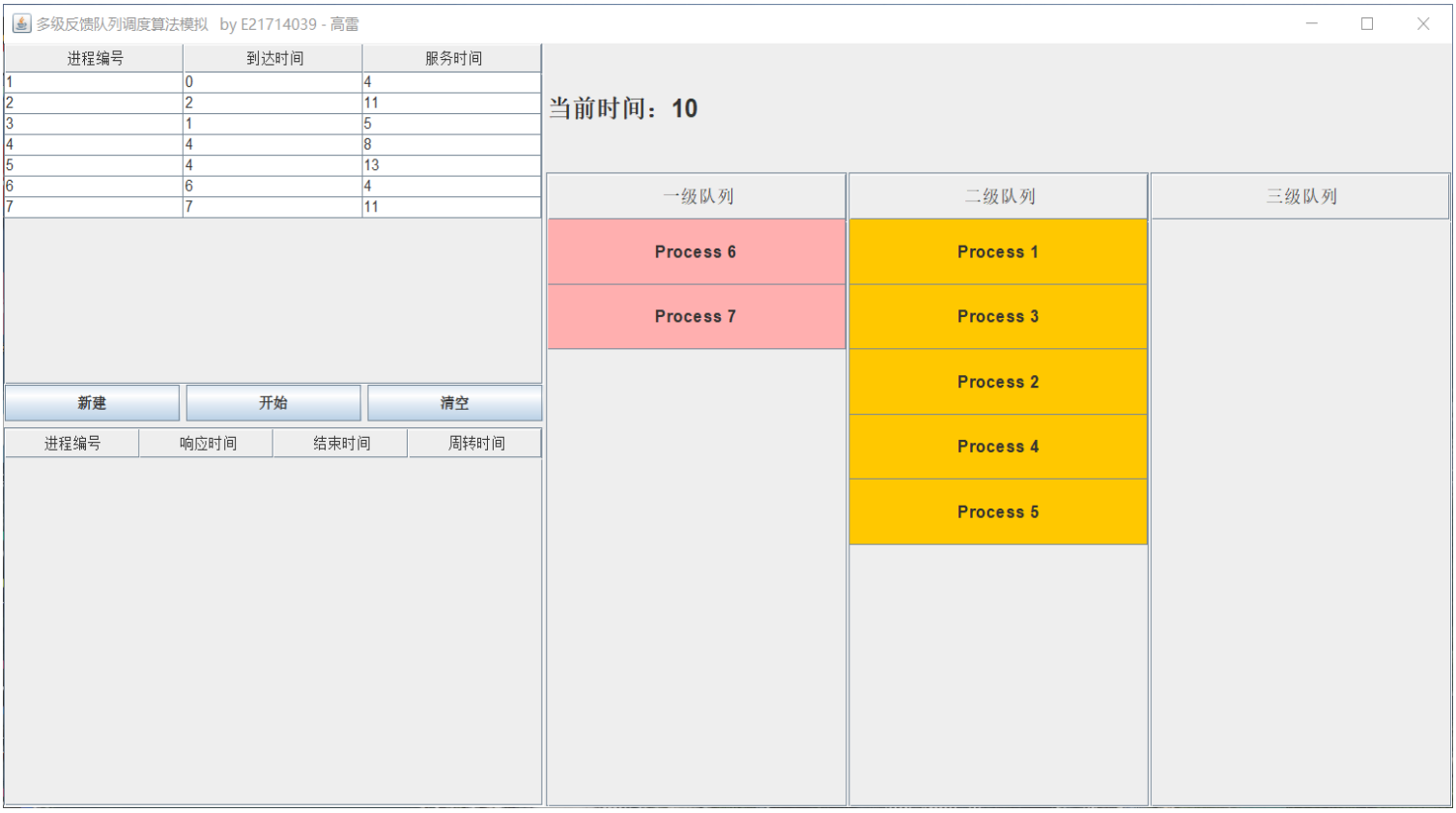
操作系统课程设计实验报告

E21714039 高雷

一、项目概述与操作说明

本项目主要利用 Java Swing 图形界面开发工具来对操作系统中的多级反馈队列算法进行模拟与分析。项目界面如下图，界面中主要包含“创建进程”、“多级队列状态模拟”和“运行结果”三个功能区。



1. 创建进程区

进程编号	到达时间	服务时间
1	0	4
2	2	11
3	1	5
4	4	8
5	4	13
6	6	4
7	7	11

新建　开始　清空

(1) 新建进程

点击“新建”按钮，输入进程的到达时间和服务时间两项基本信息后，自动按创建先后顺序为进程分配编号。

注：到达时间和服务时间必须为整数。

(2) 开始模拟

点击“开始”按钮，将开始在界面右侧显示进程和队列的实时运行状态。

注：开始模拟后不能再创建添加新进程，需等待这一轮模拟结束。

(3) 清空进程

点击“清空”按钮，将把所有创建的进程信息和调度结束后的执行信息清空，之后可以开始进行新一轮的模拟

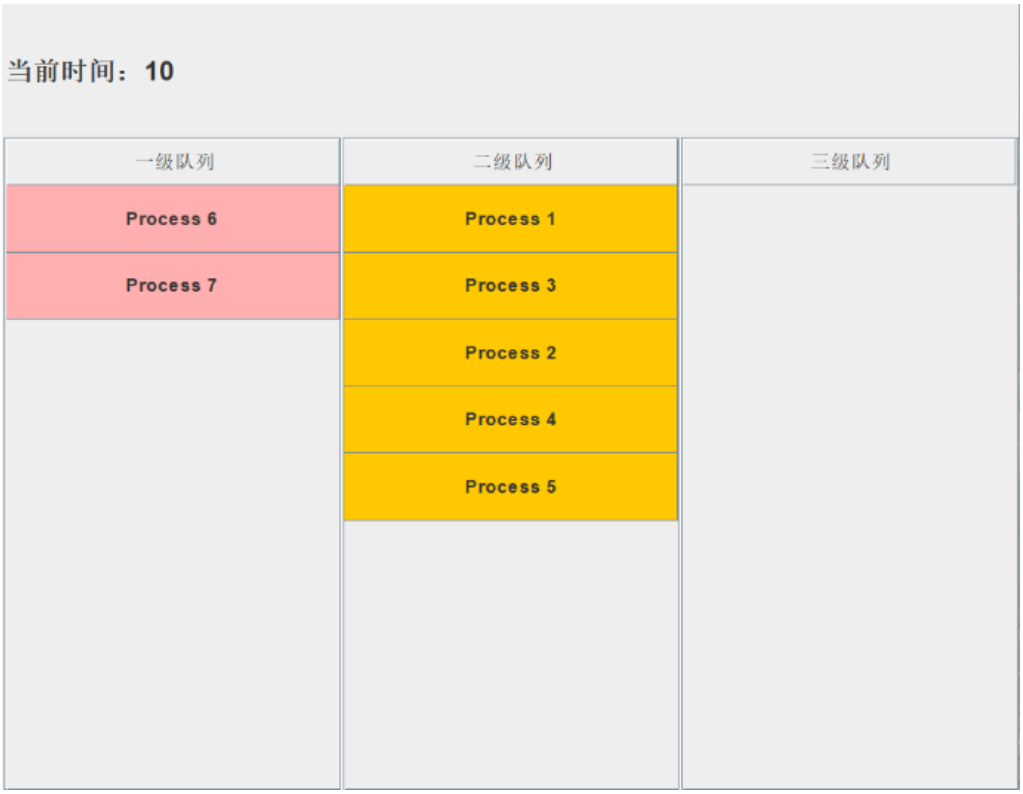
注：开始模拟的过程中不能点击清空按钮，需等待这一轮模拟结束。

2. 运行结果区

进程编号	响应时间	结束时间	周转时间
1	0	16	16
3	1	19	18
6	4	33	27
2	2	42	40
4	2	44	40
5	4	51	47
7	5	56	49

进程执行结束后，执行信息将显示在该区域

3. 多级队列状态模拟区



该区域上方为计时器，点击“开始”按钮后从 0 开始计时，直至最后一个进程执行结束。下方为三级反馈队列，队列优先级从左向右依次递减。每经过一秒，队列内的进程状态信息动态刷新。界面的动态刷新采用 Java 提供的多线程机制实现。

二、算法设计与实现

1. PCB 进程控制块设计

Process.java

```java
package MLFQ;

public class Process {

    private int pid;                //进程编号
    private int arriveTime;         //进程到达时间
    private int serviceTime;        //进程所需服务时间

    private int startTime;          //进程开始时间
    private int runTime;            //进程运行时间
    private int endTime;            //结束时间
    private int turnaround;         //周转时间

    private boolean execute = false;//是否头次执行
```

源码中的 Process.java 包含了进程控制块信息，进程控制块中主要包含两部分

(1) 进程创建信息：进程编号、到达时间、服务时间

(2) 进程执行信息：

开始时间：进程结束在一级队列内空等，被处理机响应的时间点

运行时间：进程未结束前，从进程开始到当前计时器时间的时间段

结束时间：进程结束调度并被释放的时间点

周转时间：从进程开始到进程结束的时间段

注：由于队列内部采用的是非抢占式的时间片轮转调度算法，因此没有在进程控制信息内设置“优先级”这一属性

2. Multilevel Feedback Queue 多级反馈队列设计

MultiLevelQueue.java

```java
package MLFQ;

import java.util.LinkedList;

public class MultiLevelQueue {
    private int priority;
    private int time_slice;
    private LinkedList<Process> queue = new LinkedList<Process>();

    public MultiLevelQueue(int priority, int time_slice) {…
    }
```

多级反馈队列包含了优先级和时间片两项属性。

本次项目中模拟了三级优先队列因此为多级队列设置了一二三 3 个优先级，一级队列时间片大小为 2，二级队列时间片大小为 4，三级队列时间片大小为 8

3. 模拟调度算法设计

```
15  public class MLFQScheduling extends Thread{
16
17      static int timeSlice=2;                          //最小时间片为2
18      static int queuesize=10;
19      static int processnum=0;                         //进程编号
20      static int currentTime = 0;
21      static Process[] newprocess;
22      static Process[] pool=new Process[queuesize];    //存储待排序处理的所有输入进程信息
23
24      static DefaultTableModel res = new DefaultTableModel();
25      static DefaultTableModel firstqueue = new DefaultTableModel();
26      static JTable jt_first=new JTable(firstqueue);
27      static DefaultTableModel secondqueue = new DefaultTableModel();
28      static JTable jt_second=new JTable(secondqueue);
29      static DefaultTableModel thirdqueue = new DefaultTableModel();
30      static JTable jt_third=new JTable(thirdqueue);
31
32      static JLabel clock = new JLabel();
33      static JPanel picture=new JPanel(new BorderLayout());
34
35      static LinkedList<Process> firstQueue = new LinkedList<>();
36      static LinkedList<Process> secondQueue = new LinkedList<>();
37      static LinkedList<Process> thirdQueue = new LinkedList<>();
38
39      static JFrame f = new JFrame("多级反馈队列调度算法模拟   by E21714039 - 高雷");
40
41      public static void createProcess(int pid,int arrivetime,int servicetime) {···
47      }
48
49      public static void startScheduling() throws InterruptedException{···
52      }
53
54      public static void doneScheduling() throws InterruptedException {···
73      }
74
75      public static Process[] sortbyarrival(Process[] pool) {···
93      }
94
95      public static Process[] Schedule(Process[] process) throws InterruptedException{···
240     }
241
242     public static void initFrame() {    ···
397     }
398
399     public static void main(String[] args) {     ···
403     }
404 }
```

(1) createProcess 函数用于从图形界面接收所创建的进程信息，暂存在 pool 数组中，等待排序后被调度

(2) startScheduling 函数绑定了图形界面中的“开始”按钮，用于响应开始调度的请求

(3) doneScheduling 函数用于将调度结束后的执行信息显示在图形界面中的“运行结果”区域内

(4) sortbyarrival 函数用于对图形界面接收的进程信息进行按到达时间排序的预处理

(5) Schedule 函数用于执行进程的调度，具体说明已在程序注释中给出，这里不再赘述

(6) InitFrame 函数用于搭建用户图形界面