**程序设计实训报告**



**作者：何炜（独立完成）**

**班级：计科1501**

**学号：2015014278**

**完成日期：2017年7月1日星期日**

目录

[一、 课题介绍 3](#_Toc487143467)

[二、 进程调度模拟程序设计 3](#_Toc487143468)

[（一）运行截图 3](#_Toc487143469)

[（二）注意事项 3](#_Toc487143470)

[（三）原理分析 4](#_Toc487143471)

[三、 模拟实现动态分区存储管理 4](#_Toc487143472)

[（一） 运行截图 5](#_Toc487143473)

[（二） 原理分析 5](#_Toc487143474)

[四、 分页管理方式下存储分配情况模拟 8](#_Toc487143475)

[（一） 运行截图 9](#_Toc487143476)

[（二） 原理分析 9](#_Toc487143477)

[作业分配和回收部分 9](#_Toc487143478)

[作业的分页及块号显示 10](#_Toc487143479)

[主存的物理块的分区情况 10](#_Toc487143480)

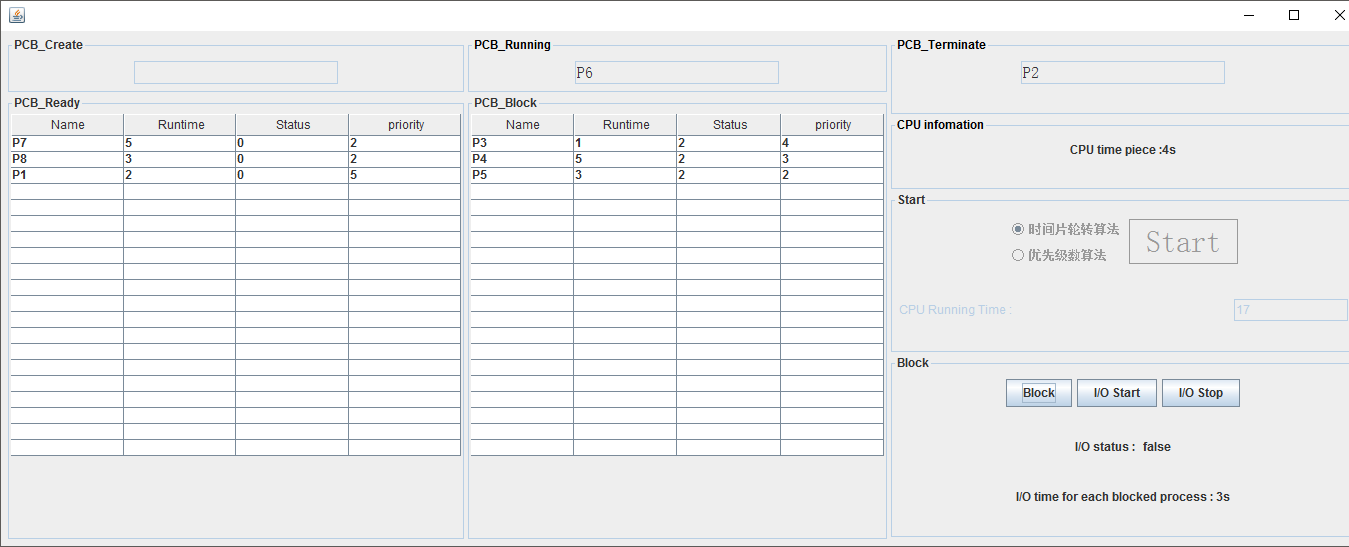
[五、 收获及体会 10](#_Toc487143481)

1. 课题介绍

本次课题由本人独立完成。一共完成了以下三个课题：

1. 进程调度模拟程序设计（进程管理）
2. 模拟实现动态分区存储管理（存储管理）
3. 分页管理方式下存储分配情况模拟（存储管理）
4. 进程调度模拟程序设计

## （一）运行截图



## （二）注意事项

1. 由于windows和Linux的目录结构 ，

在Linux下，PCB.java的120行：

File file=**new** File("/" + xmlpath.toString().replace("file:/", "")+"process/JOB.txt");

在windows下面，需要改成：

File file=**new** File(xmlpath.toString().replace("file:/", "")+"process/JOB.txt");

2. 该Java程序由于需要读写文件功能，所以必须放置在C 盘之外的盘符，或者C 盘允许读写权限。

## （三）原理分析

分为四个线程，同步运行：

* CheckIODevice：从进程创建开始到所有进程运行结束。负责一直检测I/0设备是否开启。以便于让阻塞队列中的进程以3s为时间周期回到就绪队列。
* CreatPCBQueue：创建pcbs数组的进程。从JOB.txt中导入进程信息到PCBs2的数组中，并显示在界面中。
* RereshData：刷新界面中的组件的数据
* ScanningReadyQueue：从就绪队列中根据算法选择合适的进程，使其运行

分为两个算法：

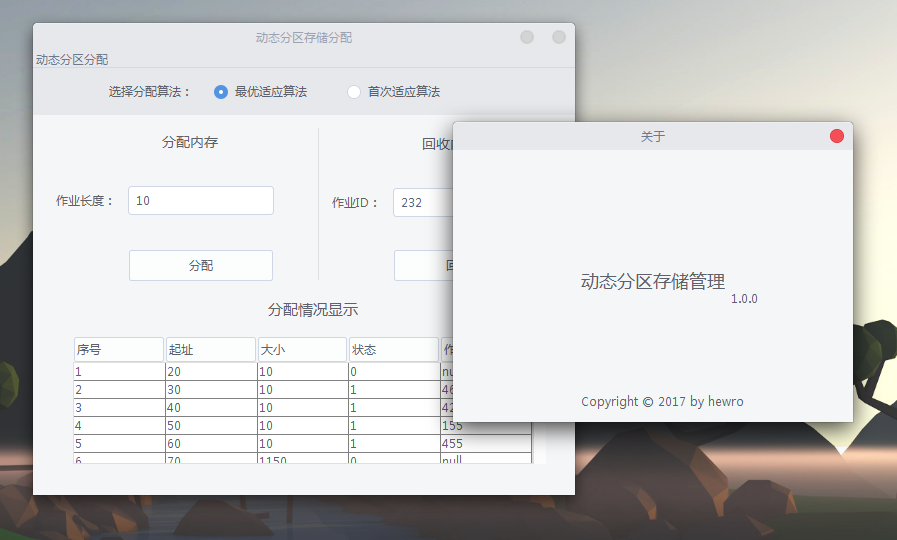
* 时间片轮转算法：本质是先来先服务 + 4s的运行周期
* 优先数算法：本质是4s的运行周期 + 按优先数排序运行进程

这两个算法非常简单。

优先数算法是每经过一个运行周期，当前进程的优先数会减1。

1. 模拟实现动态分区存储管理

## 运行截图



## 原理分析

1. **DynamicPartitioned类**

这个是整个程序最核心的类。

**主要的成员变量有：**

1. MainMemorySize：是主存的剩余空间大小
2. MainMemoryStart：是主存的起始地址
3. ArrayList<Task> tasks：用来存储所有的作业的链表
4. ArrayList<Partition>：用来存储所有的分区链表（包括空闲分区和已经占用的分区）
5. MyTableModel dtm：存储分区的表格的表格模型

**主要的方法有：**

1. partionAcllocation（）：分配分区算法

作业分配有两种分配分区的算法：

* 1. 最优适应算法

最优使用算法，每次分配时候是按照空闲分区的大小进行分配。所以每次分配之前，将主存的分区***按照分区大小进行排序***，然后选第一个能够分配的分区分配给作业即可。

* 1. 首次适应算法

首次适应算法，是在分配之前***先将所有分区按照起始地址排序***一遍，然后直接选择第一个能够分配的分区分配给作业即可。

1. partionRecovery（）：回收作业分区算法

回收作业相关的分区的时候，有四种情况，定义了两个变量***front=0*** ***back=0***：

1. 该作业的前面分区是空闲分区，后面分区是空闲分区，**front = 1,back =1**,回收的时候，三个分区同时合并
2. 该作业的前面分区是空闲分区，后面分区不是空闲分区，**front = 1,back =0**,回收的时候，前面分区和当前分区一起合并
3. 该作业的前面分区不是空闲分区，后面分区是空闲分区，**front = 0,back =1**,回收的时候，当前分区和后面分区一起合并
4. 该作业的前面分区不是空闲分区，后面分区不是空闲分区，**front = 0,back =0**,回收的时候，只把当前的分区的状态修改为空闲分区
5. **GeneratingRandomNumbers类**

生成随机数的类。单独写出来，为了能够复用（在下一个程序中也用到了该类）。该随机数的生成的数不会重复。

思路很简单，把生成的数都存储在一个链表中，每次返回生成数之前先判断是否有过该随机数，如果有，则重新生成一个数，直到输出一个不重复的随机数。

1. **GlobalObject类**

用来将某些变量设置为全局变量，方面所有的类都能够**获取和设置变量。**

1. **MyTableModel类**

是用来表示存储主存所有分区的分区表的抽象模型。

最主要的函数是reloadJTable（），用来重载表格中的数据。

每次重载表格数据的时候，先获取mainInstance.partitions的数据，然后修改表格的数据模型，这样就能够重载表格了。

1. **Partition类**

用来描述每个分区的类。主要的成员变量为：

1. startAddress：起始地址
2. size：分区大小
3. status：分区状态
4. task：分区装的作业
5. **Task类**

用来描述每个用户输入的作业的类。主要成员变量为：

1. taskName：作业名称（随机生成唯一数字）
2. size：作业的大小
3. **SortByPartitionSize类**

这个类继承了Comparator的接口。用来按照分区的大小进行排序，在最优分配算法中使用到。

1. **sortByPatitonAdress类**

这个类继承了Comparator的接口。用来按照分区的起始地址进行排序。在首次适应分配算法中使用到。

1. 分页管理方式下存储分配情况模拟

## 运行截图



## 原理分析

分为三个部分：

### 作业分配和回收部分

**分配主存：**

从作业大小的输入框中获取到作业的页面数。，然后在主存的所有物理块中，寻找空闲物理块。

如果当前主存的物理块空闲，则将该物理块加入到该作业的块号数组中，并把该物理块的状态设置为已经占用。直到寻找到所有数目的符合要求的物理块数。

**回收主存：**

通过唯一的作业ID回收主存。每个作业都绑定了一个相关的物理块数。遍历所有的物理块，将其的状态设置为空间，并在作业链表中将该作业删除。

### 作业的分页及块号显示

显示当前作业的页号及其对应的块号。

最左边的一列显示所有已经分配的作业ID 。

### 主存的物理块的分区情况

主存的物理块是一共是64个。所以用9\*8的表格存储数据。第一列存储的是表头的数据。

块号、字、位都是从0开始的。

已知块号i，其行号是：i/8。其列号是：i%8 + 1。

1. 收获及体会

北京化工大学计算机科学与技术学院至今一共有了三次课设。第一次课设使用c语言写的简单的一个计算器，没有任何界面，但是能够实现任一大小的加减乘除。第二次课设用c语言写的一个简单学生考试系统，没有任何GUI界面，是在控制台输出。这次，第三次，由于学习了java的相关的知识，使用了Swing组件写了一些简单的GUI界面，自己也感到了一定的进步。但是深知和班级一些同学差距还是很远。自己对java的掌握仍然停留在比较浅的层面，今后还必须加强Java开发的深入学习。

在这次课设过程中，题目的难度不大，但是要重复的考虑到每一个用户可能的输出 的值而导致的错误的结果，也就是程序的健壮性，才是我们应该重点考虑的。除此之外，除了算法，用户体验以及界面优化对于程序的开发也非常的重要。

1. 源代码下载