**武汉大学计算机学院**

**本科生课程设计报告**

**操作系统实验**

专 业 名 称 ：软件工程

课 程 名 称 ：操作系统课程设计

指 导 教 师 ：宋伟 副教授

学 生 学 号 ：2018302110400

学 生 姓 名 ：谭攀

二○二〇年五月

**郑 重 声 明**

本人呈交的设计报告，是在指导老师的指导下，独立进行实验工作所取得的成果，所有数据、图片资料真实可靠。尽我所知，除文中已经注明引用的内容外，本设计报告不包含他人享有著作权的内容。对本设计报告做出贡献的其他个人和集体，均已在文中以明确的方式标明。本设计报告的知识产权归属于培养单位。

本人签名： 谭攀 日期：2020 5.16

摘 要

操作系统实验的实验目的是为了了解处理器调度，存储空间的分配与回收，进程与程序的区别以及死锁的避免。

实验内容主要包括：

1、选择一个调度算法，实现处理器调度。

2、主存储器空间的分配和回收。

3、模拟磁盘空闲空间的表示方法，以及模拟实现磁盘空间的分配和回收。

4、利用fork()系统调用创建进程。

5、设计实现银行家算法。

**目 录**

**1** **实验目的和意义**

1.1 实验目的 ………………………………………………………………………1

1.2 实验意义 ………………………………………………………………………1

**2** **实验一**

2.1 概述…………………………………………………………………………… 2

2.2 实验原理……………………………………………………………………… 2

2.3 实验方案……………………………………………………………………… 2

2.4 结果…………………………………………………………………………… 2

**3** **实验二**

3.1 概述…………………………………………………………………………… 4

3.2 实验原理……………………………………………………………………… 4

3.2 实验方案……………………………………………………………………… 4

3.4 结果…………………………………………………………………………… 5

**4** **实验三**

4.1 概述……………………………………………………………………………… 6

4.2 实验原理……………………………………………………………………… 6

4.3 实验方案……………………………………………………………………… 6

4.4 结果…………………………………………………………………………… 7

**5** **实验四**

5.1 概述……………………………………………………………………………… 8

5.2 实验原理………………………………………………………………………… 8

5.3 实验方案………………………………………………………………………… 8

5.4 结果……………………………………………………………………………… 8

**6** **实验五**

6.1 概述…………………………………………………………………………… 10

6.2 实验原理……………………………………………………………………… 10

6.3 实验方案……………………………………………………………………… 11

6.4 结果…………………………………………………………………………… 12

**结果** …………………………………………………………………………………… 13

**1 实验目的和意义**

**1.1** **实验目的**

本实验分为五个小实验，实验一的目的是模拟在单处理器环境下的处理器调度，加深了解处理器调度的工作。实验二的目的是理解在不同的存储管理方式下应怎样进行存储空间的分配和回收。实验三的目的是掌握磁盘存储空间的分配和回收算法。实验四的目的是了解进程的创建过程，进一步理解进程的概念，明确进程和程序的区别。实验五的目的是了解死锁的避免，掌握银行家算法。

**1.2** **实验意义**

通过完成本实验，深入了解操作系统的处理器调度，存储调度，进程线程调度，从而了解操作系统对各类资源的分配方式以及是如何利用各类算法提高系统对资源的利用率。

**2 实验一**

**2.1** 概述

本实验有两个选题，我选择了第一题：设计一个按优先数调度算法实现处理器调度的程序。

**2.2** 实验原理

每个进程用一个PCB来代表，PCB有进程名；要求运行时间：假设进程需要运行的单位时间数；优先数：赋予进程的优先数，调度时总是选取优先数大的进程先执行；状态：就绪和结束，用R表示就绪，用E表示结束。初始状态都为就绪状态。

**2.3** 实验方案

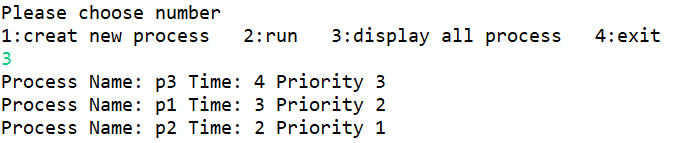
采用java来完成本实验，设计了四个类分别为Service, Main, PCB及PCBCreator. Main类为输入输出相关的类。PCB类定义了每个进程的数据结构，包含进程名，要求运行时间，优先级及状态，并且包含一个将进程信息打印出来的方法。PCBCreator类是用来创建PCB对象的类。

Service类包含一个用来表示就绪队列的存储PCB对象的链表，包含一个进程运行的方法，每次运行将就绪队列的第一个进程的优先级和要求运行时间减1，再判断该进程的要求运行时间是否为0，为零则从就绪队列删除。运行一次后调用排序方法根据优先级进行排序。

**2.4** 结果

随机使用一组数据进行测试，创建了如图2.3.1的进程，运行结果如图2.3.2。

进程运行时是遵循优先级高的先运行。可以看出，实验代码完成了要求。



图**2.3.1**



图2.3.2

**3 实验二**

**3.1** 概述

本实验有两个选题，我选择了第一题：可变分区管理方式下采用首次适应算法实现主存分配和回收。

**3.2** 实验原理

按作业需要的主存空间大小来分割分区的。当要装入一个作业时，根据作业需要的主存容量查看是否有足够的空闲空间，若有，则按需分配，否则，作业无法装入。

空闲分区说明表格式：分区号——表示是第几个空闲分区；起始地址——指出空闲区的起始地址；长度——一个连续空闲区的长度。

采用首次适应算法分配回收内存。

**3.3** 实验方案

采用java来完成本实验，设计了三个类分别为Memory, Main及Partition. Main类为输入输出相关的类。Partition类定义了每个分区的数据结构，包含分区起始地址，分区长度以及分区状态。

Memory类包含两个链表，分别用来存放已使用的分区和空闲分区，包含申请，回收，显示内存状态，以及另外三个辅助的方法。申请方法从空闲分区的链表中遍历，第一个分区长度大于申请长度的用于分配主存，分配时新建一个Partition对象，将原空闲分区的起始地址及长度修改继续作空闲分区，新对象作为被分配的空间加入已分配空间的链表。加入后再调用排序方法，按照起始地址的顺序进行排序。

回收方法将要回收的Partition对象的状态修改修改为未分配，再从已分配链表里移除，加入空闲分区链表。再调用排序方法，按照起始地址的顺序进行排序。最后调用空闲分区拼接方法，将相邻的空间分区拼接。

**3.4** 结果

随机使用一组数据进行测试，运行结果如图3.3。

申请，回收都能正常进行，特殊情况需要拼接的也能正常完成。可以看出，代码完成了实验要求。

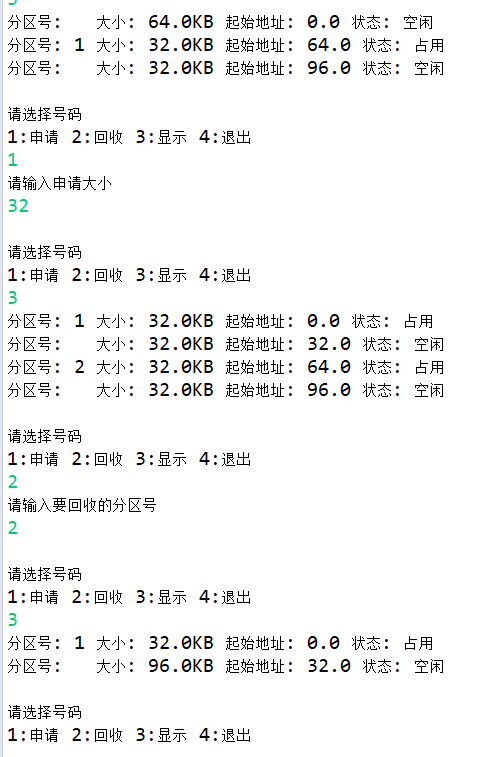


图3.3

**4 实验三**

**4.1** 概述

本实验有三个选题，我选择了第二题：用位示图管理磁盘存储空间。

**4.2** 实验原理

为了提高磁盘存储空间的利用率，可在磁盘上组织成链接文件、索引文件，这类文件可以把逻辑记录存放在不连续的存储空间。为了表示哪些磁盘空间已被占用，哪些磁盘空间是空闲的，可用位示图来指出。位示图由若干字节构成，每一位与磁盘上的一块对应，“1”状态表示相应块已占用，“0”状态表示该块为空闲。

申请一块磁盘空间时，由分配程序查位示图，找出一个为“0”的位，计算出这一位对应块的磁盘物理地址，且把该位置成占用状态“1”。假设现在有一个盘组共8个柱面，每个柱面有2个磁道（盘面），每个磁道分成4个物理记录。那么，当在位示图中找到某一字节的某一位为“0”时，这个空闲块对应的磁盘物理地址为：柱面号=字节号 磁道号= 位数 / 4 物理记录号= 位数 % 4

归还一块磁盘空间时，由回收程序根据归还的磁盘物理地址计算出归还块在位示图中的对应位，把该位置成“0”。按照（2）中假设的盘组，归还块在位示图中的位置计算如下：字节号=柱面号 位数=磁道号4+物理记录号

**4.3** 实验方案

采用java来完成本实验，设计了三个类分别为Map, Main及Service. Main类为输入输出相关的类。Map类定义了位视图的数据结构，包含总字节数，总位数以及用来表示位示图的二维数组。

Service类包含显示位示图，申请，回收三个方法，申请时遍历位示图，计算剩余空间，如果满足条件，可分配；同时利用公式将物理地址计算处理出来，将分配的空间对应的物理地址打印出来。

回收方法利用公式计算出在位示图中的坐标，将该位置0。

**4.4** 结果

随机使用一组数据进行测试，运行结果如图4.3.1及4.3.2。

申请，回收都能正常操作，可见代码完成了实验要求。

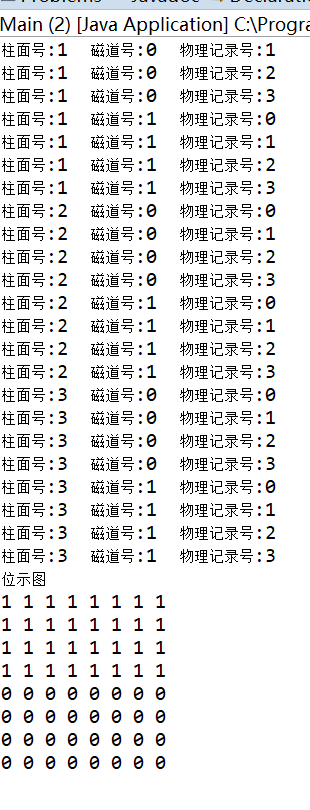
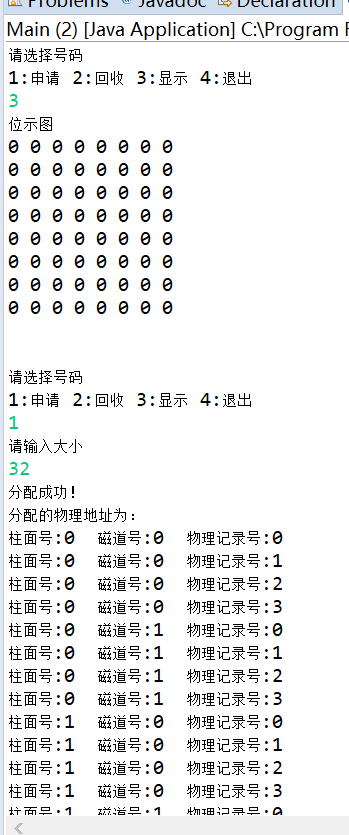


图4.3.1

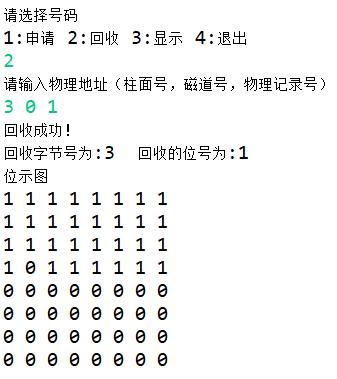


图4.3.2

**5 实验四**

**5.1** 概述

利用fork()系统调用创建进程。

**5.2** 实验原理

可用fork()系统调用来创建一个新进程。

系统调用格式：pid=fork()

fork()返回值意义如下：

=0：若返回值为0，表示当前进程是子进程。

>0：若返回值大于0，表示当前进程是父进程，返回值为子进程的pid值。

-1：若返回值小于0，表示进程创建失败。

如果fork()调用成功，它向父进程返回子进程的pid，并向子进程返回0，即fork()被调用了一次，但返回了两次。此时OS在内存中建立一个新进程，所建的新进程是调用fork()父进程的副本，称为子进程。子进程继承了父进程的许多特性，并具有与父进程完全相同的用户级上下文。父进程与子进程并发执行。

**5.3** 实验方案

利用fork()创建两个子进程，在每个进程里都循环10次打印字符和进程号。

**5.4** 结果

编译后多次运行可发现，三个进程运行的顺序是不定，符合题目中描述的父进程与子进程并发执行。

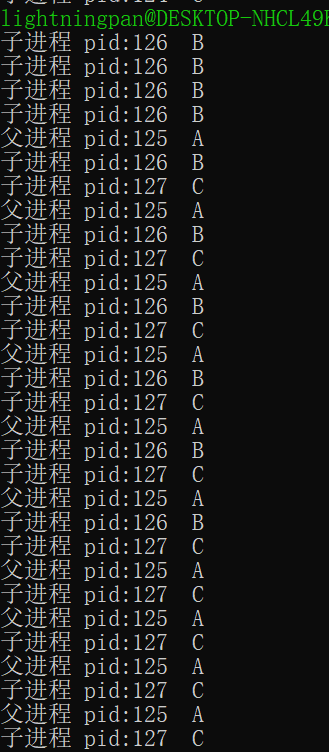
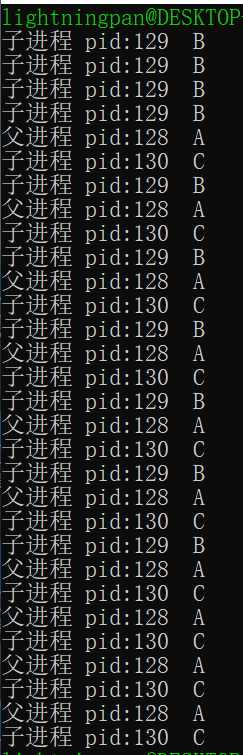


图5.3

**6 实验五**

**6.1** 概述

设计、实现银行家算法。

**6.2** 实验原理

银行家算法。假定系统中有n个进程，m类资源，数据结构如下：

1. 可利用资源向量Available。是个含有m个元素的数组，其中的每一个元素代表一类可利用的资源数目。如果Available[j]=K，则表示系统中现有Rj类资源K个。
2. 最大需求矩阵Max。这是一个n×m的矩阵，它定义了系统中n个进程中的每一个进程对m类资源的最大需求。如果Max[i, j]=K，则表示进程i需要Rj类资源的最大数目为K。
3. 分配矩阵Allocation。这也是一个n×m的矩阵，它定义了系统中每一类资源当前已分配给每一进程的资源数。如果Allocation[i, j]=K，则表示进程i当前已分得Rj类资源的 数目为K。
4. 需求矩阵Need。这也是一个n×m的矩阵，用以表示每一个进程尚需的各类资源数。如果Need[i, j]=K，则表示进程i还需要Rj类资源K个，方能完成其任务。

三矩阵关系: Need[i, j]=Max[i, j]-Allocation[i, j]

设Requesti 是进程Pi的请求向量，Requesti[j]=k表示进程Pi请求分配Rj类资源k个。系统按如下步骤检查：

1. 如果Requesti<= Needi，则转(2)；否则，出错。
2. 如果Requesti<= Allocationi，则转(3)；否则，等待。
3. 系统试探分配资源，修改相关数据：

Available=Available - Request;

Allocation=Allocation + request;

Need=Need-Request;

1. 系统执行安全性检查，如安全，则分配成立；否则试探险性分配作废，系统恢复原状，进程等待。

安全性检查算法：

1. 设置两个工作向量Work=Available; Finish:Finish[i]=false;
2. 从进程集合中找到一个满足下述条件的进程，

Finish[i]==false;

Needi<=Work;

如找到，执行（3)；否则，执行（4)。

(3)设进程获得资源，可顺利执行，直至完成，从而释放资源。

Work=Work+Allocation;

Finish[i]=true;

转向第二步。

(4) 如所有的进程Finish[i]为true，则表示安全；否则系统不安全。

**6.3** 实验方案

采用java来完成本实验，设计了四个类分别为AllocSystem, Main, Process以及Array. Main类为输入输出相关的类。Process类定义了进程的数据结构，包含该进程的分配数组，最大需求数组以及需求数组。Array类定义了与数组相关的计算的方法。

AllocSystem类包含一个可利用资源向量，一个元素为进程的链表。包含打印各进程情况及系统可利用资源的方法，请求分配资源的方法以及辅助的安全性检查方法。

分配资源方法首先检测需求数组是否小于进程需求数组以及可利用资源数组。再尝试进行分配，利用分配的结果进行安全性检查，若安全，则分配，否则取消分配。

安全性检测方法，新建临时链表和work数组，将进程链表克隆过去，work数组初始值设为可用资源数组，不断遍历临时链表，满足元素进程的需求数组小于等于work数组，则将该元素移除链表，若最终临时链表能将所有元素移除，则说明是安全的。

**6.4** 结果

利用教材书中数据，运行结果如图6.3.1。

三次申请的结果符合实际情况。

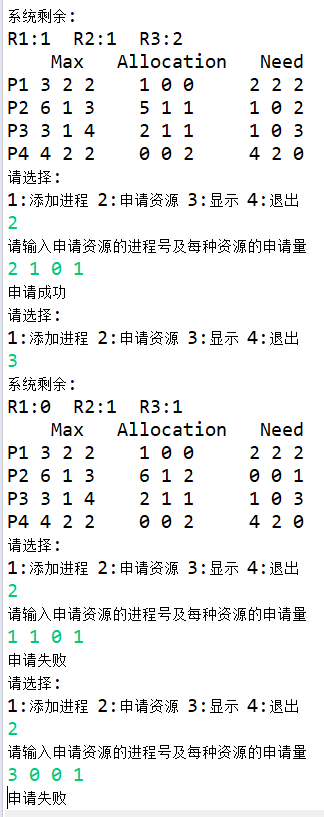


图6.3

结果

所有代码都能顺利运行，所有实验结果都能满足实验要求和实验预期。