

实验	一	二	三	四	五	六	七	八	九	十	总评
成绩											

武汉大学计算机学院

课程实验(设计)报告

专业(班): 16 级弘毅计算机

学 号: 2016300030006

姓 名: 关焕康

课程名称: 操作系统设计

2018 年 7 月 5 日

目录

实习一 处理器调度.....	3
1.实验内容.....	3
2.实验目的.....	3
3.实验要求.....	3
4.实验过程.....	3
5.实验结果.....	3
6.实验结论:	4
实习二 主存空间的分配和回收.....	5
1.实验内容.....	5
2.实验目的.....	5
3.实验要求.....	5
4.实验过程.....	5
5.实验结果.....	5
6.实验结论:	6
实习三 磁盘存储空间的分配和回收.....	7
1.实验内容.....	7
2.实验目的.....	7
4.实验过程.....	7
5.实验结果.....	8
6.实验结论:	8
选做实验一 进程创建.....	9
1.实习内容.....	9
2.实习目的.....	9
3.实习要求.....	9
4.实验过程.....	9
6.实验结论.....	9
参考文献.....	9

源码上传至

Github: [git@github.com:guanhuankang/2018OS3labs.git](https://github.com/guanhuankang/2018OS3labs.git)

实习一 处理器调度

1.实验内容

选择一个调度算法，实现处理器调度。

2.实验目的

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。

3.实验要求

设计一基于优先数或者时间片轮转的算法完成进程调度的模拟实验。

4.实验过程

我选择基于优先数的调度算法，优先数越大，优先级越高，优先数控制台输入，优先数和剩余时间更新公式如下：

$$\text{Nice} = (\text{is Running})? \text{Nice} - 1 : \text{Nice};$$
$$\text{Time} = (\text{is Running})? \text{Time} - 1 : \text{Time};$$

也就是说，每运行一次 Nice 减一，Time 减一，知道 Time=0 时退出队列。

本实验我才用两种方法完成。

方法一是利用 C++ STL 模块优先队列 `priority_queue` 实验。程序框图如图 1。

方法二是利用线段树查找最大 Nice 值，这样每次只需要 $O(\log n)$ 的时间复杂度找到最高优先级的进程。

两个方法的代码都已经上传到 Github 托管代码平台，分别对应 A.cpp 和 AA.cpp 文件，。

5.实验结果

样例输入是 3 个进程，优先数和运行时间如表：

程序名称	初始优先数	估计运行时间
1	4	2
2	2	2
3	6	3

结果显示，进程调度一次是(进程名字)3-3-1-3-1-2-2。结果正确（详细输出见 A.out）。

6.实验结论:

该实验较为简单，主要是如何设计优先数更新使得尽可能公平，本实验中进程每运行一次优先数减一，实质是一种动态优先级，这样可以即体现高优先级的进程，又有效降低进程出现饥饿可能。

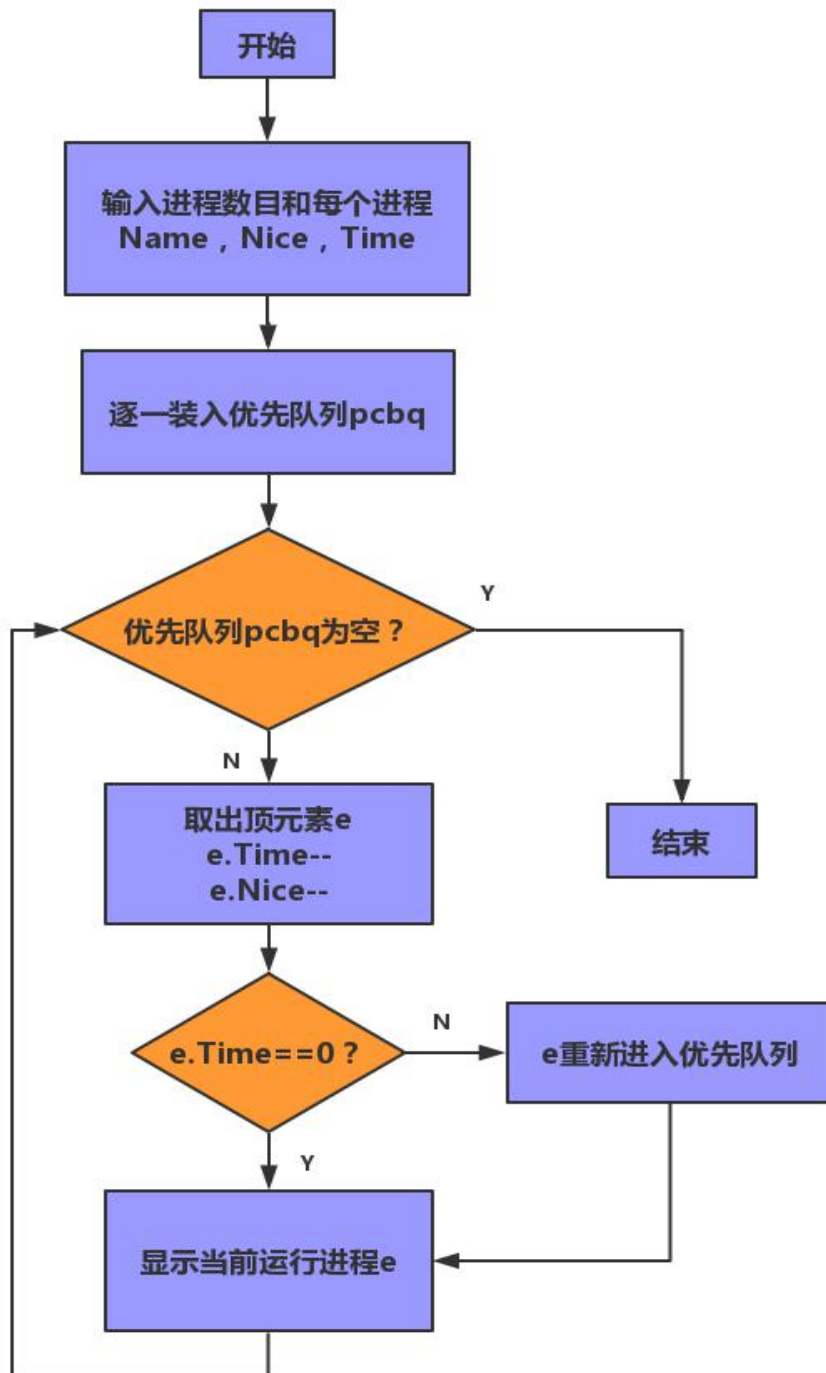


图1.优先队列实现进程基于优先数调度

实习二 主存空间的分配和回收

1.实验内容

主存储器空间的分配和回收。

2.实验目的

通过本实习帮助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。

3.实验要求

模拟主存储空间分配和释放过程。

4.实验过程

我选择首次适应算法完成模拟实验。实验思路采用链表法。维护一个链表，链表的每个节点是一块连续空间，记录该块是否空闲，起始结束地址，块大小，占用进程信息等信息。合并时检查前后节点是否空闲，改变起始地址，块大小等，释放合并的块，只保留一块。分配时，从链表头部往后找到第一个满足块，划分该块，该块多余空间用新建的节点表示，并完成新节点的插入。实验思路如图 2。

5.实验结果

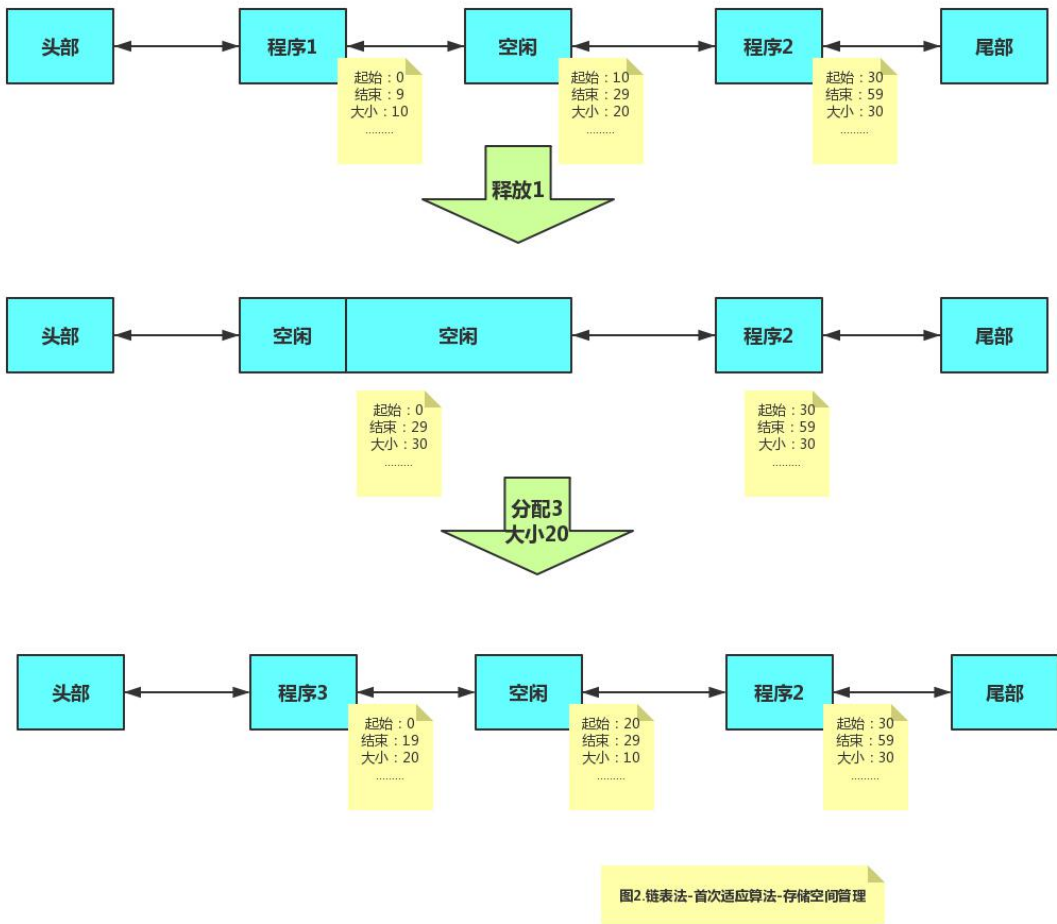
样例输入(格式说明<allocate 进程名字 x 大小 s>表示进程 x 请求分配 s 字节空间；<free 进程名称 x>表示释放进程 x；<query>表示查询当前主存储器存储情况。)

```
allocate 1 10
allocate 2 20
allocate 3 30
allocate 4 40
allocate 5 1000000
query
free 2
query
free 1
query
free 3
query
free 4
query
```

实验结果显示一切正常，由于输出较长，详细输出见 B.out，实验代码 B.cpp。

6.实验结论:

该实验较为简单，主要是注意释放空间时需要检查前后节点是否需要合并，这样才可以形成一块较大的连续空间。首次适应算法是每次从头部开始查找最先满足分配要求的空间，若满足立即分配，与最优适应算法相比，它出现较多小碎块的几率要低，但是在保留大块连续空间方面可能不如最优适应算法。



实习三 磁盘存储空间的分配和回收

1.实验内容

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。

2.实验目的

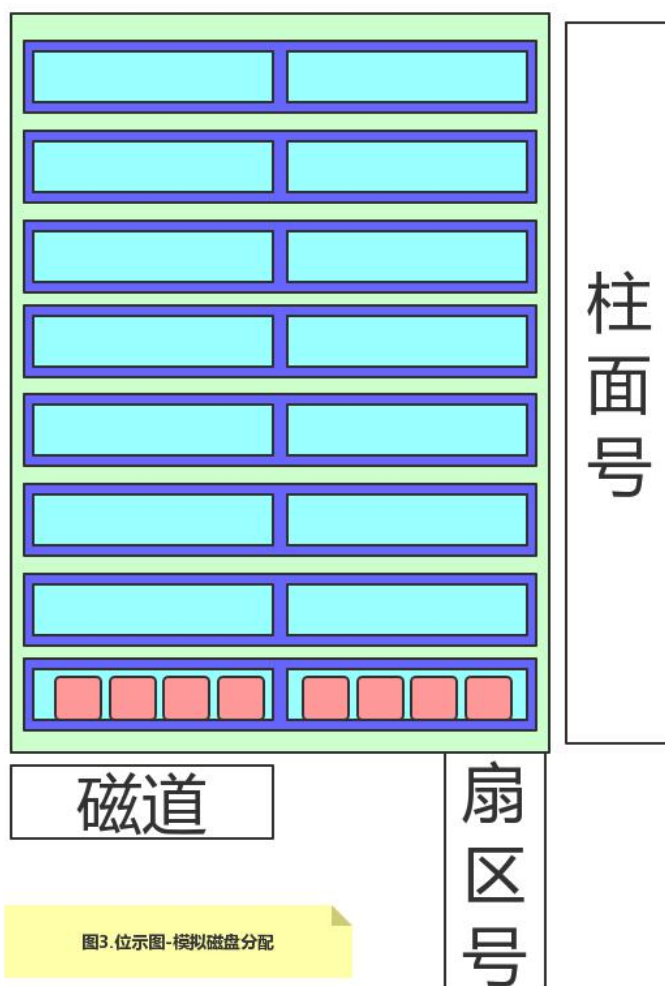
磁盘初始化时把磁盘存储空间分成许多块（扇区），这些空间可以被多个用户共享。用户作业在执行期间常常要在磁盘上建立文件或把已经建立在磁盘上的文件删去，这就涉及到磁盘存储空间的分配和回收。一个文件存放到磁盘上，可以组织成顺序文件（连续文件）、链接文件（串联文件）、索引文件等，因此，磁盘存储空间的分配有两种方式，一种是分配连续的存储空间，另一种是可以分配不连续的存储空间。怎样有效地管理磁盘存储空间是操作系统应解决的一个重要问题，通过本实习使学生掌握磁盘存储空间的分配和回收算法。

3.实验要求

使用位示图的方法完成模拟磁盘空间分配实验。当磁盘空间不足以分配时拒绝分配。分配空间应该从磁盘逻辑地址低地址开始分配，已经分配的不能重复分配，释放磁盘空间时，会给出柱面号，磁道，扇区号。

4.实验过程

使用一个一维数组 `disk[64]` 表示磁盘总共块数，总共 8 个柱面，每个柱面 2 个磁道，每个磁道有 4 个扇区，所以总共 $8*2*4=64$ 块（一个扇区是一块）。分配时先判断剩余扇区是否足够，若足够就从低柱面号，低磁道，低扇区号开始扫描，边扫描边分配，反之直接不分配。实验思路图解如图 3。



5.实验结果

样例输入(格式说明<allocate 大小 s>表示请求分配 s 个扇区；<free a b c>表示释放 a 柱面，b 磁道，扇区 c；<query>表示查询当前磁盘存储情况。)

```
allocate 5
query
free 0 0 0
query
allocate 1
query
allocate 10
query
allocate 60
query
```

实验结果显示一切正常，由于输出较长，详细输出见 C.out，实验代码 C.cpp。

6.实验结论：

模拟磁盘分配的实验中，使用位示图来表示块的空闲，空间效率非常高，比如一个扇区 512KB 时，空间开销接近 $1/2^{22}$ ，所以位示图表示块情况具有很高的空间效率，但是，时间复杂度是 $O(n)$ ，这个复杂度其实相当高。

选做实验一 进程创建

1.实习内容

利用 fork()系统调用创建进程。

2.实习目的

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。

3.实习要求

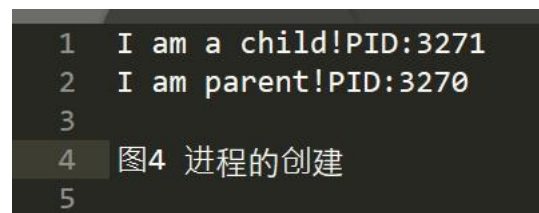
编制一段程序，使用系统调用 fork()创建两个子进程，这样在此程序运行时，在系统中就有一个父进程和两个子进程在活动。每一个进程在屏幕上显示一个字符，其中父进程显示字符 A，子进程分别显示字符 B 和字符 C。试观察、记录并分析屏幕上进程调度的情况。

4.实验过程

调用 fork()创建进程，fork 函数一次调用，返回两次，父进程返回子进程 PID，子进程返回 0，以此也可以区分父子进程，从而完成实验。

5.实验结果

见图 4，代码 D.cpp，输出结果 D.out。



```
1 I am a child!PID:3271
2 I am parent!PID:3270
3
4 图4 进程的创建
5
```

6.实验结论

Fork()函数可以返回新进程的 PID(父进程角度)，子进程返回 0，一次区分父子进程。另外，查询资料可以知道父子进程共享代码，采用写时拷贝技术，数据，堆栈为两份。

参考文献

参考文献：《OPERATING SYSTEM CONCEPT (Seventh Edition)》 Abraham Silberschatz, Peter Baer Galvin, Greg Gagne。