《操作系统分析与设计实习》

指导书

（第二版）

华南农业大学数学与信息学院，软件学院

孙微微主编，张丽霞，王金凤参编

目　录

第一部分 《操作系统分析与设计实习》要求

一、《操作系统分析与设计实习》教学概述

1、教学基本情况

课程总学时数：2 周；　　课程总学分：2 学分

适用专业：数学与信息学院计算机科学与技术、软件工程、网络工程专业，软件学院软件工程专业

考核方式及方法：实际操作＋程序运行＋课程设计报告。

成绩评定：在参考“难度系数”的基础上

>=90——功能完善，编程风格好，人机接口界面好，报告结构清晰，总结和分析详尽；

80~90——功能完善，编程风格良好，人机接口界面良好，报告结构完整；

70~80——功能基本完善，编程风格良好，报告结构完整；

60~70——能完成所选题目，并提交报告；

<60——未按时完成，或者抄袭（含雷同者）。

2、教学指导思想和教学目的

1）指导思想：通过由浅入深、循序渐进、精讲多练，培养学生对计算机操作系统的熟练使用，使学生全面了解操作系统的特点，熟练掌握操作系统的基本设计方法和系统工作原理。

2）教学目的：使学生通过实践来验证课堂教学的理论，并学会设计一些简单的综合应用程序或小型的模拟操作系统。

3、项目表

一共提供六个题目供大家选择，题目难度逐级递增，即第一个题目难度最小，第 6 个题目难度最大。前四个题目需要单独完成，后两个题目可组队完成，其中第五个题目建议 2 到 3 人一队，第 6 个题目建议 3 到 5 人一队。每位同学根据自己的能力选择一个题目。

不要求对真实操作系统的各种数据结构进行操作，而是对你自己所设置的一些数据结构（如数组、链表或队列）进行操作，来模拟实现操作系统中的算法或调度行为。你可以采用任何你熟悉的语言编写。

二、《操作系统分析与设计实习》教学规范

1、课程意义

本课程是对学生的一种全面综合训练，是与课堂听讲、自学和练习相辅相成的必不可少的一个教学环节。通常，实验所涉及到的问题比平时的习题复杂得多，也更接近实际。本课程着眼于原理与应用的结合点，使学生学会如何把书上学到的知识用于解决实际问题，培养软件工作所需要的动手能力；另一方面，能使书上的知识变"活"，起到深化理解和灵活掌握教学内容的目的。平时的练习较偏重于如何编写功能单一的"小"算法，而本课程涉及到的实验题目是软件设计的综合训练，包括问题分析、总体结构设计、用户界面设计、程序设计基本技能和技巧，多人合作，以至一整套软件工作规范的训练和科学作风的培养。此外，还有很重要的一点是：机器是比任何教师都严厉的检查者。

2、实验步骤

常用的软件开发方法，是将软件开发过程划分为分析、设计、实现和维护四个阶段。虽然本课程所涉及到的实验题目远不如实际问题中的复杂程度高，但为了培养一个软件工作者所应具备的科学工作的方法和作风，也应遵循以下五个步骤来完成实验题目：

1）问题分析和任务定义

在进行设计之前，首先应该充分地分析和理解问题，明确问题要求做什么，限制条件是什么。本步骤强调的是做什么，而不是怎么做。对问题的描述应避开算法和所涉及的数据类型，而是对所需完成的任务作出明确的回答。例如：输入数据的类型、值的范围以及输入的形式；输出数据的类型、值的范围及输出的形式；若是会话式的输入，则结束标志是什么，是否接受非法的输入，对非法输入的回答方式是什么等。还应该为调试程序准备好测试数据，包括合法的输入数据和非法形式的输入数据。

2）逻辑设计和详细设计

在设计这一步骤中需分逻辑设计和详细设计两步实现。逻辑设计指的是，对问题描述中涉及的操作对象定义相应的数据类型，并按照以数据结构为中心的原则划分模块，定义主程序模块和各抽象数据类型；详细设计则为定义相应的存储结构并写出各函数的伪码算法。在这个过程中，要综合考虑系统功能，使得系统结构清晰、合理、简单和易于调试，抽象数据类型的实现尽可能做到数据封装，基本操作的规格说明尽可能明确具体。作为逻辑设计的结果，应写出每个抽象数据类型的定义(包括数据结构的描述和每个基本操作的功能说明)，各个主要模块的算法，并画出模块之间的调用关系图。详细设计的结果是对数据结构和基本操作作出进一步的求精，写出数据存储结构的类型定义，写出函数形式的算法框架。在求精的过程中，应尽量避免陷入语言细节，不必过早表述辅助数据结构和局部变量。

3）编码实现和静态检查

编码是把详细设计的结果进一步求精为程序设计语言程序。如果基于详细设计的伪码算法就能直接在键盘上输入程序的话，则可以不必用笔在纸上写出编码，而将这一步的工作放在上机准备之后进行，即在上机调试之前直接用键盘输入。

然而，不管你是否写出编码的程序，在上机之前，认真的静态检查是必不可少的。静态检查主要有两种方法，一是用一组测试数据手工执行程序(通常应先分模块检查)；二是通过对程序深

入全面地理解程序逻辑，在这个过程中再加入一些注解和断言。如果程序中逻辑概念清楚，后者将比前者有效。

4）上机准备和上机调试

上机准备包括以下几个方面：

(1)注意同一高级语言文本之间的差别。

(2)熟悉机器的操作系统和语言集成环境的用户手册，尤其是最常用的命令操作，以便顺利进行上机的基本活动。

(3)掌握调试工具，考虑调试方案，设计测试数据并手工得出正确结果。应该能够熟练运用高级语言的程序调试器 DBBUG 调试程序。

(4)上机调试程序时要带一本高级语言教材或手册。调试最好分模块进行，自底向上，即先调试低层函数。在调试过程中可以不断借助 DEBUG 的各种功能，提高调试效率。调试中遇到的各种异常现象往往是预料不到的，此时应动手确定疑点，通过修改程序来证实它或绕过它。调试正确后，认真整理源程序及其注释，形成格式和风格良好的源程序清单和结果。

5）总结和整理实验报告

3、课程设计报告(文档)规范

报告的开头应首先包括如下成绩单表格，并填写班级、学号、姓名、题目等信息。在成绩单之后另起一新页，开始你的报告主体。

华 南 农 业 大 学 数 学 与 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

<table>
<tr><td colspan="10">小组成员、组内分工、工作量比例、各成员个人成绩</td></tr>
<tr><td>学号</td><td></td><td>姓名</td><td></td><td>分工</td><td></td><td>工作量比例</td><td></td><td>成绩</td><td></td></tr>
<tr><td>学号</td><td></td><td>姓名</td><td></td><td>分工</td><td></td><td>工作量比例</td><td></td><td>成绩</td><td></td></tr>
<tr><td>学号</td><td></td><td>姓名</td><td></td><td>分工</td><td></td><td>工作量比例</td><td></td><td>成绩</td><td></td></tr>
<tr><td>学号</td><td></td><td>姓名</td><td></td><td>分工</td><td></td><td>工作量比例</td><td></td><td>成绩</td><td></td></tr>
<tr><td>实验题目</td><td colspan="9">若有更多小组成员可加行。
这是小组的实验报告成绩单，作为实验报告的第一页</td></tr>
<tr><td>自我评价</td><td colspan="9">（实验体会和心得）</td></tr>
<tr><td>教师评语</td><td colspan="9">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 □</td></tr>
<tr><td></td><td colspan="4"></td><td colspan="2">教师签名</td><td colspan="3"></td></tr>
</table>

然后，在报告主体中包括以下六个内容：

一、需求分析

明确陈述说明程序设计的任务，强调的是程序要做什么，主要包括：

(1)输入的形式和输入值的范围；

(2)输出的形式；

(3)程序所能达到的功能；

(4)测试数据：包括正确的输入及其输出结果和含有错误的输入及其输出结果。

二、概要设计

说明本程序中用到的所有抽象数据类型的定义、主程序的流程以及各程序模块之间的层次(调用)关系。

三、详细设计

实现概要设计中定义的所有数据类型，对每个操作只需要写出伪码算法；对主程序和其他模块也都需要写出伪码算法(伪码算法达到的详细程度应能够按照伪码算法在计算机键盘上直接输入高级程序设计语言程序)；画出函数的调用关系图。

四、调试分析

内容包括：

(1) 调试过程中遇到的问题是如何解决的以及对设计与实现的讨论和分析；

(2) 算法的时间复杂性(包括基本操作和其他算法的时间复杂性的分析)和改进设想；

(3) 设计过程的经验和体会；

(4) 实现过程中出现的主要问题及解决方法。

五、用户使用说明

说明如何使用你编写的程序，详细列出每一步的操作步骤。

六、测试与运行结果

列出你的测试结果和运行情况（即运行时的关键画面），包括输入和输出。这里的测试数据应该完整和严格，最好多于需求分析中所列。

值得注意的是，报告的各种文档资料，要在程序开发的过程中逐渐充实形成，而不是最后补写。必要时可在报告中附部分关键源代码，但不需要附全部源代码。

4、课程考核

1）考核点：编程(50 分)、测试分析(20 分)、报告(30 分)；

2）**上交内容（电子版）：**

①源代码；

②可执行程序；

③课程设计报告。

上交时间与地点由任课教师指定。**请各班学习委员注意：上交电子版文件时，每位或每组同学的内容各自放在单独的一个文件夹中（此文件夹不要压缩），文件夹的名字格式形如：202131000101 陈陈，即只包含学号姓名，且学号和姓名之间没有任何空格及其它符号。小组文件夹以组内成员“学号姓名列表”作为文件夹名称。**

最后整个班压缩为一个文件发送给任课教师。

第二部分 《操作系统分析与设计实习》内容

题目一、二、三、四只能由单人完成。题目五、六可由多人组成小组合作完成，每组上交**一份**报告即可，在报告中各小组成员需说明自己的分工情况和完成体会，以便评阅教师对小组成员分别评分（百分制）。

如果想摆脱“数据结构大作业”的感觉，想深入理解文件系统和操作系统各模块的原理，那么建议选择题目五或题目六。

题目一 页面置换算法的模拟实现及命中率对比

一、课程设计目的

通过请求页式管理方式中页面置换算法的模拟设计，了解虚拟存储技术的特点，掌握请求页式存储管理中的页面置换算法。

二、课程设计内容

模拟实现 OPT（最佳置换）、FIFO 和 LRU 算法，并计算缺页率。

三、要求及提示

本题目必须单人完成。

1、首先用随机数生成函数产生一个“指令将要访问的地址序列”，然后将地址序列变换成相应的页地址流（即页访问序列），再计算不同算法下的命中率。

2、通过随机数产生一个地址序列，共产生 400 条。其中 50%的地址访问是顺序执行的，另外 50%就是非顺序执行。且地址在前半部地址空间和后半部地址空间均匀分布。具体产生方法如下：

1) 在前半部地址空间，即[0，199]中随机选一数 m，记录到地址流数组中（这是非顺序执行）；
2) 接着“顺序执行一条指令”，即执行地址为 m+1 的指令，把 m+1 记录下来；
3) 在后半部地址空间，[200，399]中随机选一数 m’，作为新指令地址；
4) 顺序执行一条指令，其地址为 m’+1；
5) 重复步骤 1~4，直到产生 400 个指令地址。

3、将指令地址流变换成页地址（页号）流，简化假设为：

1) 页面大小为 1K（这里 K 只是表示一个单位，不必是 1024B）；
2) 用户虚存容量为 40K；

3) 用户内存容量为 4 个页框到 40 个页框；

4) 用户虚存中，每 K 存放 10 条指令，所以那 400 条指令访问地址所对应的页地址（页号）流为：指令访问地址为[0，9]的地址为第 0 页；指令访问地址为[10，19]的地址为第 1 页；……。按这种方式，把 400 条指令组织进“40 页”，并将“要访问的页号序列”记录到页地址流数组中。

4、循环运行，使用户内存容量从 4 页框到 40 页框。计算每个内存容量下不同页面置换算法的命中率，命中率=1-缺页率。输出结果可以为：

页框数	OPT 命中率	FIFO 命中率	LRU 命中率
[4]	OPT：0.5566	FIFO：0.4455	LRU：0.5500
[5]	OPT：0.6644	FIFO：0.5544	LRU：0.5588
……	……		
……	……		
[39]	OPT：0.9000	FIFO：0.9000	LRU：0.9000
[40]	OPT：1.0000	FIFO：1.0000	LRU：1.0000

注 1：在某一次实验中，可能 FIFO 比 LRU 性能更好，但足够多次的实验表明 LRU 的平均性能比 FIFO 更好。

注 2：计算缺页率时，以总缺页次数计算。

题目二　磁盘调度算法的模拟实现及对比

一、课程设计目的

通过磁盘调度算法的模拟设计，了解磁盘调度的特点。

二、课程设计内容

模拟实现 FCFS、SSTF、电梯 LOOK、C-SCAN 算法，并计算及比较磁头移动道数。

三、要求及提示

本题目必须单人完成。

1、首先假设磁盘磁道数为 1500，磁头初始位置可任意设置。

2、用随机数生成函数产生“磁道号”序列（即磁盘请求的位置），共产生 400 个。其中 50%位于 0～499，25%分布在 500～999，25%分布在 1000～1499。具体的产生方法可参考“题目一 页面置换算法的模拟实现及命中率对比”。

3、计算及比较每种磁盘调度算法下的磁头移动道数。

注：本题目要求给出图形可视化界面，并且能够动态模拟每个算法的调度过程，可采用从上一个请求到下一个请求进行连线的方式。

题目三 进程同步与互斥

一、课程设计目的

掌握基本的同步与互斥算法，掌握进程并发执行的原理，及其所引起的同步、互斥问题的方法。

二、课程设计内容

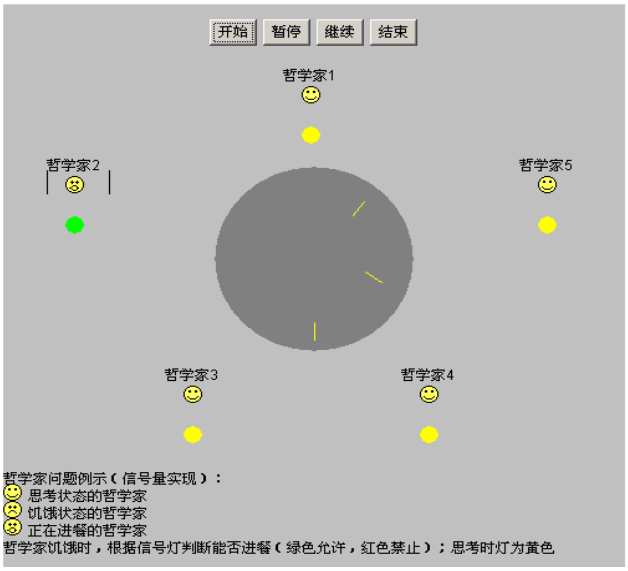
自己编写信号量和 wait、signal 操作的模拟程序，然后用它们解决不死锁的哲学家问题或者读者-写者问题。

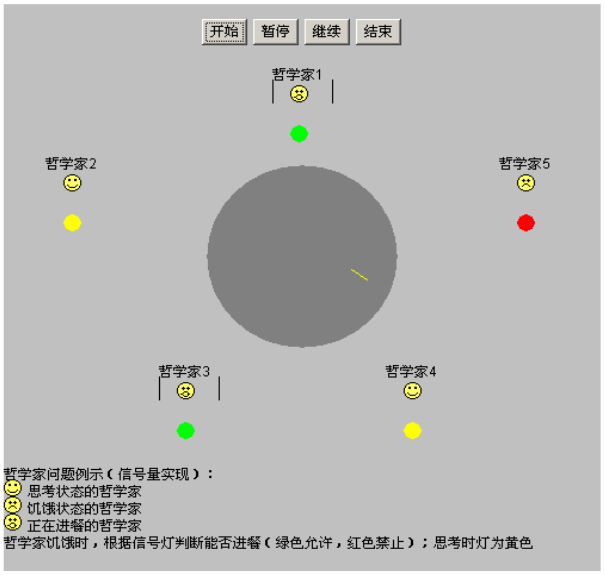
三、要求及提示

本题目必须单人完成。

1、解决不死锁的哲学家问题，要求把哲学家们的活动过程用文字或图形可视化形式表示出来。

提示：首先设置一个“PCB”数组或队列，其中一个字段表示“阻塞原因兼阻塞标志”，本实验中，该数组有 5 个元素表示 5 个哲学家即可。它们随机提出申请以及进行“思考”“吃”的行为。再设一个“筷子”数组。还需要设置哪些数据结构以及需要哪些字段自己考虑。示例图如下，仅供参考。





2、解决读者-写者问题，仅要求解决读者优先的情况。

提示：创建一个控制台进程，此进程包含 n 个线程。用这 n 个线程来表示 n 个读者或写者。每个线程按相应测试数据文件的要求进行读写操作。

读者-写者问题的读写操作限制：

1)写-写互斥，即不能有两个写者同时进行写操作。

2)读-写互斥，即不能同时有一个线程在读，而另一个线程在写。

3)读-读允许，即可以有一个或多个读者在读。

读者优先的附加限制：如果一个读者申请进行读操作时已有另一个读者正在进行读操作，则该读者可直接开始读操作。但任何写者必须等到没有读者时才能开始写操作。

运行结果显示要求：要求在每个线程创建、发出读写操作申请、开始读写操作和结束读写操作时分别显示一行提示信息，以确定所有处理都遵守相应的读写操作限制。

测试数据文件包括 n 行测试数据，分别描述创建的 n 个线程是读者还是写者，以及读写操作的开始时间和持续时间。每行测试数据包括四个字段，各个字段间用空格分隔。第一字段为一个正整数，表示线程序号。第二字段表示相应线程角色，R 表示读者，W 表示写者。第三字段为一个正数，表示读写操作的开始时间：线程创建后，延迟相应时间(单位为秒)后发出对共享资源的读写申请。第四字段为一个正数，表示读写操作的持续时间。当线程读写申请成功后，开始对共享资源的读写操作，该操作持续相应时间后结束，并释放共享资源。测试数据随机生成，为简化起见，假设每个线程只执行一次读或写操作，之后运行结束。

下面是一个测试数据文件的例子：

线程序号	角色	何时开始读写	读写持续时间
1	R	3	5
2	R	4	5
3	W	5	2
4	R	10	3
5	W	11	3
6	R	13	4
……	……	……	

可能用到的 API 函数：

CreateThread()	在调用进程的地址空间上创建一个线程
ExitThread()	用于结束当前线程
Sleep()	可在指定的时间内挂起当前线程
CreateMutex()	创建一个互斥对象，返回对象句柄
OpenMutex()	打开并返回一个已存在的互斥对象句柄，用于后续访问

ReleaseMutex()	释放对互斥对象的占用，使之成为可用
WaitForSingleObject()	可在指定的时间内等待指定对象为可用状态
InitializeCriticalSection()	初始化临界区对象
EnterCriticalSection()	等待指定临界区对象的所有权
LeaveCriticalSection()	释放指定临界区对象的所有权
CreateSemaphore()	创建一个信号量对象
ReleaseSemaphore()	将所指信号量加上指定大小的一个量

题目四 单处理器系统的进程调度

一、课程设计目的

1. 加深对进程概念的理解，明确进程和程序的区别。

2. 深入了解系统如何组织进程、创建进程。

3. 进一步认识如何实现处理器调度。

二、课程设计内容

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。

三、要求及提示

本题目必须单人完成。要求能够动态地随机生成新进程添加到就绪队列中。

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。

1、组织进程

考虑如何组织进程，首先要设定进程控制块的内容。进程控制块 PCB 记录各个进程执行时的情况。不同的操作系统，进程控制块记录的信息内容不一样。操作系统功能越强，软件也越庞大，进程控制块的内容也就越多。这里只使用必不可少的信息。一般操作系统中，无论进程控制块中信息量多少，信息都可以大致分为以下四类：

（1）标识信息

每个进程都要有一个唯一的标识符，用来标识进程的存在和区别于其他进程。这个标识符是必不可少的，可以用符号或编号实现，它必须是操作系统分配的。例如采用编号方式，也就是为每个进程依次分配一个不相同的正整数。

（2）说明信息

用于记录进程的基本情况，例如进程的状态、等待原因、进程程序存放位置、进程数据

存放位置等等。实验中，因为进程没有数据和程序，仅使用模拟的进程控制块，所以这部分内容仅包含进程状态。

进程状态可假设只有就绪、运行、终止三种。如果要设置“等待（阻塞）”状态，需要随机生成“阻塞时间”，阻塞时间到时转为就绪态。

（3）现场信息

现场信息记录各个寄存器的内容。当进程由于某种原因让出处理器时，需要将现场信息记录在进程控制块中，当进行进程调度时，从选中进程的进程控制块中读取现场信息进行现场恢复。现场信息就是处理器的相关寄存器内容，包括通用寄存器、程序计数器和程序状态字寄存器等。在本实验中，本部分可忽略。

（4）管理信息

管理信息记录进程管理和调度的信息。例如进程优先数、进程队列指针等。

另外，本实验为了模拟进程的不同运行时间，再添加一个“剩余运行时间”属性。

因此可将进程控制块结构定义如下：

```
struct pcb
{ int name;              //进程标识符
  int status;            //进程状态
  int pri;               //进程优先数
  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
}
```

确定进程控制块内容后，要考虑的就是如何将进程控制块组织在一起。多道程序设计系统中，往往同时创建多个进程。在单处理器的情况下，每次只能有一个进程处于运行态，其它的进程处于就绪状态或者等待状态。为了便于管理，通常把处于相同状态的进程的进程控制块链接在一起组成就绪队列和等待队列。

由于实验模拟的是进程调度，没有对等待队列的操作，所以实验中只有一个指向正在运行进程的进程控制块指针和一个就绪进程的进程控制块队列指针。操作系统实现中，系统往往在主存中划分出一个连续的专门区域存放系统的进程控制块，实验中应该用数组模拟这个专门的进程控制块区域，定义如下：

```
#define   n   10                //假定系统允许进程个数为 n
struct pcb pcbarea[n];        //模拟进程控制块区域的数组
```

这样，进程控制块的链表实际上是数据结构中使用的静态链表。实验中，进程控制块队列采用单向不循环静态链表。为了管理空闲进程控制块，还应该将空闲控制块链接成一个队列。

进程调度其实就是一个排队的过程，不同的算法区别在于按照什么样的次序将就绪队列里面的进程进行排序，比如时间片轮转调度算法，是将进程控制块按照进入就绪队列的先后次序排队。关于就绪队列的操作就是从队头摘下一个进程控制块和从队尾挂入一个进程控制块。因此为就绪队列定义两个指针，一个头指针，指向就绪队列的第一个进程控制块；一个尾指针，指向就绪队列的最后一个进程控制块。

实验中指向运行进程的进程控制块指针、就绪队列指针和空闲进程控制块队列指针定义如下：

```
int run;            //定义指向正在运行进程的进程控制块的指针
struct
{   int head;
    int tail;       //定义指向就绪队列的头指针 head 和尾指针 tail
}ready;
int pfree;          //定义指向空闲进程控制块队列的指针
```

2、创建进程

进程创建是一个原语，因此在实验中应该用一个函数实现，进程创建的过程应该包括：

（1）申请进程控制块

进程控制块的数量是有限的，如果没有空闲进程控制块，则进程不能创建，如果申请成功才可以执行第二步。

（2）填写进程控制块

将该进程信息写入进程控制块内。进程标识符应该随机生成并且是唯一，优先数和剩余运行时间随机生成，刚刚创建的进程为就绪态，然后转去执行第三步。

（3）挂入就绪队列

如果原来就绪队列不为空，则将该进程挂入就绪队列尾部，并修改就绪队列尾部指针；如果原来就绪队列为空，则将就绪队列的头指针、尾指针均指向该进程控制块，进程创建完成。

多道程序设计的系统中，处于就绪状态的进程往往是多个，它们都要求占用处理器，可是单处理器系统的处理器只有一个，进程调度就是解决这个处理器竞争问题的。进程调度的

任务就是按照某种算法从就绪进程队列中选择一个进程，让它占有处理器。因此进程调度程序就应该包括两部分，一部分是在进程就绪队列中选择一个进程，并将其进程控制块从进程就绪队列中摘下来，另一部分工作就是分配处理器给选中的进程，也就是将指向正在运行进程的进程控制块指针指向该进程的进程控制块，并将该进程的进程控制块信息写入处理器的各个寄存器中。

提醒注意的是：在实际的系统中，当一个进程被选中运行时，必须恢复进程的现场，让它占有处理器运行，直到出现等待事件或运行结束。在本实验里省去了这些工作。

以时间片轮转调度算法为例说明如何挂入就绪队列。时间片轮转调度算法让就绪进程按就绪的先后次序排成队列，每次总是选择就绪队列中的第一个进程占有处理器，但是规定只能使用一个“时间片”。时间片就是规定进程一次使用处理器的最长时间。实验中采用每个进程都使用相同的不变时间片。每被调度 1 次，将其剩余运行时间－1。

进程运行一次后，若剩余运行时间不等于 0，则再将它加入就绪队列尾；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、调度

完成上述功能后，编写主函数进行测试：首先建立一个就绪队列，随机生成信息建立若干个进程；然后进行进程调度；将正在运行进程指针指向的进程控制块的内容以及调度一次后进程队列的现状输出，查看结果。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”。

题目五　模拟磁盘文件系统实现

一、课程设计目的

了解磁盘文件系统的结构、功能和实现。并可练习合作完成系统的团队精神和提高程序设计能力。

二、小组人数

建议 3～5 人一组共同完成模拟磁盘文件系统的实现。

选择题目“模拟磁盘文件系统实现”的小组在最终提交时须公开演示及讲解。由于这个题目较复杂，难度和工作量远大于前面几个题目，故小组成员最后得分也酌情高于选择前面四个题目的同学的分数（高 5~10 分）。

三、编程语言

建议使用一些 Windows 环境下的程序设计语言如 VC、Java，以借助这些语言的多线程来模拟并行发生的行为。要求图形界面。

四、课程设计内容

设计一个简单的文件系统，用文件模拟磁盘，用数组模拟缓冲区，要求：

（1）　支持多级目录结构，支持文件的绝对读路径；

（2）　文件的逻辑结构采用流式结构，物理结构采用链接结构中的显式链接方式；

（3）　采用文件分配表 FAT；

（4）　实现的命令包括建立目录、列目录、删除空目录、建立文件、删除文件、显示文件内容、打开文件、读文件、写文件、关闭文件、改变文件属性。可以采用命令行界面执行这些命令，也可以采用“右击快捷菜单选择”方式执行命令。

（5）　最后编写主函数对所作工作进行测试。

五、课程设计具体内容和要求

为了正确地实现文件的存取，文件系统设计了一组与存取文件有关的功能模块，用户可以用“访管指令”调用这些功能模块，以实现文件的存取要求。我们把文件系统设计的这一组功能模块称为“文件操作”，实验就是要模拟实现一些文件操作。文件操作不是独立的，它和文件系统的其它部分密切相关，若要实现文件操作就离不开文件的目录结构、文件的组织结构和磁盘空间的管理。因此，这个实验虽然是文件操作的模拟实现，但是还必须模拟一部分文件的组织结构、目录结构和磁盘空间管理的实现。

（1）　文件的组织结构

文件的逻辑结构有两种形式：流式文件和记录式文件。实验中只支持流式文件，采用称为显式链接的物理文件结构，把磁盘中每一块的指针部分提出来，组织在一起，形成文件分配表（FAT）。文件分配表的作用不仅如此，其它的作用下面将提到。

磁盘有多少块，文件分配表就有多少项，若某文件的一个磁盘块号为 i，则这个文件的下一个磁盘块号应该记录在文件分配表的第 i 项。例如，某系统文件分配表的前几项值如下图所示。某个文件的起始盘块号为 3，则该文件的磁盘块号依次为 3、4、9、12、13。

项	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
内容	-1	-1	-1	4	9	0	7	8	-1	12	11	-1	13	-1	0	0	

（2） 磁盘空间的管理

首先要模拟一个磁盘。因为是实验，不使用真正的磁盘，所以实验中用一个文件模拟一个小磁盘。假设模拟磁盘有 128 个物理块，每个物理块大小为 64 字节。盘块的块号从 0 编起。

将前面所讲的文件分配表放在磁盘的开始处，因为盘块有 128 块，所以文件分配表有 128 项，每项占用一个字节。这样文件分配表占用了磁盘的 0 块和 1 块，这两块就不能作其它用处。若一个盘块是某个文件的最后一块，填写“-1”表示文件结束。

文件的建立和删除就需要对磁盘的空间进行分配和回收，所以要建立一定的数据表格来记录磁盘的使用情况。用文件分配表的第 i 项表示第 i 个盘块的使用情况。磁盘的第 0 块一定会被系统数据占用，所以任何一个文件的某个盘块块号都不可能是“0”，因而可以用“0”表示磁盘盘块空闲，若这个盘块已经分配出去，即是某个文件的一块，由上面我们知道文件分配表中对应项记录的是文件下一块的块号或结束标志都不是“0”。这样非“0”值表示盘块已分。像前图那张文件分配表中，块号为 5、14 和 15 的盘块是空闲的，其余是已分配的。在文件分配表中可以用一个超过盘块编号的正整数表示文件结束，在此实验中采用 255 代替 -1 表示文件结束。

如果磁盘中某些部分损坏，只要不是系统区（引导扇区、文件分配表或根目录等），不分配那些坏的盘块，磁盘可以继续使用。在文件分配表对应坏盘块的项不能是“0”，一般需要特定的数值表示（这个数值应该是盘块编号以外并且不是结束标志的数值，例如，实验中采用 128～254 之间的某个数值）。假设实验中模拟磁盘的第 23 块和第 49 块已经损坏，不能使用，则在文件分配表的第 23 项和第 49 项写入“254”表示该盘块损坏不能使用。

由于磁盘分配时，有时不能预定文件的大小，例如建立文件时并不知道文件的大小。因

而磁盘的分配有时是一块一块申请的。磁盘空间回收时，整个文件删除时回收很多，但有时文件修改时可能会删除某些内容，造成归还磁盘块，这时是一块一块回收的。注意：分配一个磁盘块时，不应该从文件分配表第一项查起，因为磁盘中最开始的几块为系统数据区，可假定系统区域占用了 x 个盘块，则应该从第 x 块（0～x-1 块为系统区域）开始查询。回收一个磁盘块很简单，比如回收磁盘块的块号为 x，只要找到文件分配表中第 x 项，将第 x 项的值改为 0 即可。

（3） 目录结构

文件目录是用于检索文件的，它是文件系统实现按名存取的主要手段。文件目录由若干目录项组成，每一个目录记录一个文件的有关信息。一般的说，目录项应该包括如下内容：

①有关文件的控制信息。例如，用户名、文件名。文件类型、文件属性。实验模拟个人计算机上的文件操作，这部分内容仅包括文件名、文件类型和文件属性；

②有关文件结构的信息。例如，文件的逻辑结构、文件的物理结构、记录个数、文件在存储介质的位置等。实验中，仅仅支持流式文件，不支持记录式文件，所以这部分内容仅仅包括文件在存储介质的位置（分给文件第一个盘块的块号，即起始盘块号）、文件的长度；

③有关文件管理的信息。例如，文件的建立日期、文件被修改的日期、文件保留期限和记帐信息等。实验中为了简单起见，这部分内容都不采用。

因此，实验中文件的目录项包括：文件名、文件类型、文件属性、文件的起始盘块号、文件的长度，每个目录项占用 8 个字节，具体结构如下所示：

文件名：3 个字节（实验中合法文件名仅可以使用字母、数字和除“$”、“.”、“/”以外的字符，第一个字节的值为“$”时表示该目录为空目录项，文件名和类型名之间用“.”分割，用“/”作为路径名中目录间分隔符）；

文件类型名：2 个字节；

文件属性：1 个字节；

起始盘块号：1 个字节；

文件长度：1 个字节（为了实验的简单，假设文件长度单位为盘块）

有了文件目录后，当用户要求使用某个文件时，文件系统可以顺序查找目录项，并比较文件名，就可以找到指定文件的目录项，根据目录行中有关内容核对使用权限、并读出文件供用户使用。因此文件目录的组织和管理要便于检索和防止冲突。

在操作系统中目录就有根目录和子目录两种。除了文件需要登记形成目录外，还要登记子目录的情况。实验中，根目录固定位置、固定大小（可以登记有限个文件或子目录项），

子目录像文件一样，可使用任何一个空闲磁盘块。为了实验简单，实验中根目录占用一个盘块，子目录的长度不采用可以任意长的方法，而是采用定长的方法，每个子目录的长度也是一个盘块，只能放 8 个目录项。文件和目录的登记项是混在一起的，登记项的结构应该和文件目录一样，每个目录项占用 8 个字节，结构如下：

目录名：3 个字节（实验中合法目录名仅可以使用字母、数字和除“$”、“.”、“/”以外的字符，第一个字节的值为“$”时表示该目录为空目录项）；

保留 2 字节未使用（在实验中填写空格）；

目录属性：1 个字节；

起始盘块号：1 个字节；

保留 1 字节未使用（在实验中填写“0”）。

在目录登记项中，系统为目录名后 2 个字节（对应文件类型名位置）填写空格，目录起始盘块号后 1 字节（对应文件长度位置）填写 0。目录属性和文件属性占用同一个字节，为了区别目录和文件，该字节每一位代表不同的含义（为“1”表示“是”，为“0”表示否），如下图所示，第 0 位表示文件为只读文件，第 1 位表示文件为系统文件，第 2 位表示文件为可读、可写的普通文件，第 3 位表示该登记项不是文件的登记项，而是目录的登记项，其余几位闲置未用。如该字节为 8（00001000），表示该目录是一个目录的登记项，该字节为 3（00000011），表示该目录是一个只读系统文件的登记项，该字节为 4（000000100），表示该目录是一个可读可写的普通文件。

第 7 位	第 6 位	第 5 位	第 4 位	第 3 位	第 2 位	第 1 位	第 0 位
未使用	未使用	未使用	未使用	目录属性	普通文件	系统文件	只读文件

目录检索的方法常用的是顺序检索，根据绝对路径名查找文件的方法一般如下：先找到根目录的起始盘块，一般根目录位置是固定的，实验中就是模拟磁盘的第 2 块，将该盘块读出；取出路径名中根目录后的目录名或文件名，和根目录中目录项依次比较，比较完一块，再根据文件分配表找到下一块，再读入比较，直到找到名字一致的目录项或根目录登记项均已查完为止；若没有找到，则查找失败，结束；若查找的是文件，结束；若查找的是目录，从查找到的目录项中，取出目录的起始盘块号，读入此盘块，然后用上述相同的查找方法继续查找，直到找到该文件（或目录）或查找失败结束。

查找文件除了绝对路径名外，还可以使用相对路径名。相对路径名是从当前目录出发到指定文件的路径。如果文件（或目录）在当前目录下，使用相对路径名查找速度比较快。和

绝对路径的查找方法一样，只是查找的起点是当前目录，不是根目录。实验中只使用绝对路径名。

（4） 文件操作

确定文件组织结构、目录结构和磁盘空间管理的方法后，就可以模拟文件操作的实现。实验中文件操作包括建立文件、打开文件、关闭文件、读文件、写文件、删除文件、显示文件内容和改变文件属性，目录命令包括建立目录、显示目录内容和删除空目录。在实验中没有程序调用这些命令，为了看到它们的模拟情况，从键盘输入选择指令来模拟用户程序的调用。

首先要建立一个“已打开文件表”，用来记录打开或建立文件的相关内容，结构如下图所示：

文件路径名	文件属性	起始盘块号	文件长度	操作类型	读指针		写指针	
					块号	块内地址	块号	块内地址

用数组模拟已打开文件表，数据结构定义如下：

```
#define   n   5          //实验中系统允许打开文件的最大数量
typedef struct
{ int dnum;              //磁盘盘块号
  int bnum;              //磁盘盘块内第几个字节
}pointer;                //已打开文件表中读、写指针的结构
typedef struct
{ char name[20];         //文件绝对路径名
  char attribute;;       //文件的属性，用 1 个字节表示，所以此用 char 类型
  int number;            //文件起始盘块号
  int length;            //文件长度，文件占用的字节数
  int flag;              //操作类型，用“0”表示以读操作方式打开文件，用“1”表示以写操作方式打开文件
  pointer read;      //读文件的位置，文件打开时 dnum 为文件起始盘块号，bnum 为“0”
  pointer write;     //写文件的位置，文件刚建立时 dnum 为文件起始盘块号，bnum 为“0 ，打开文件时 dnum 和 bnum 为文件的末尾位置
```

}OFTLE;　　//已打开文件表项类型定义

struct

{ OFILE file[n];　//已打开文件登记表

　int length;　　//已打开文件登记表中登记的文件数量

}openfile;　　//已打开文件登记表定义

无论上述哪种文件操作都会涉及到已打开文件表，对于已打开文件表主要是查找、删除和插入操作。下面分析所有常用的文件操作。

①建立文件（create_file）

用户要把一个新文件放到存储介质上前，首先调用文件系统的“建立”操作。

建立文件的主要工作就是检查文件目录，确认无重名文件后，寻找空闲登记项进行登记；寻找空闲存储块（至少一块）以备存储文件信息或存放索引表，最后应该填写已打开文件表。

实验中需要的参数比较少，只要有文件名和文件属性就可以，create_file（文件名，文件属性）。

实验中，建立文件时给出文件名和文件属性，文件属性如果是只读性质则不能建立；文件建立时根据给定的文件路径名进行查找，如果父目录不存在，建立文件失败；如果存在，查看有无重名文件，如果有，则提示该文件已存在，建立文件失败；如无重名文件，则为该文件建立文件目录，并分配给它一个磁盘块，最后填写目录和已打开文件表。

②打开文件（open_file）

用户要求使用一个已经存在的文件时，首先执行“打开文件”操作。可以在读或写一个文件时，默认打开该文件，将“打开”和“读或写”合二为一，不必非得先“打开”再“读或写”。

实验中，所需参数有文件名、操作类型（读或写），open_file（文件名，操作类型）。

实验中，打开文件首先要检查该文件是否存在，不存在，打开失败；如果文件存在，还要检查打开方式，确保不能以写方式打开只读文件；最后填写已打开文件表，若文件已经打开则不需要填写已打开文件表。

③读文件（read_file）

用户要求读文件信息时调用文件系统的“读文件”操作。

实验中，读文件的参数只需要文件名和读取长度，read_file（文件名，读取长度）。因为采用的是流式文件结构，所以读的长度用字节表示。

实验中，读文件操作的主要工作是查找已打开文件表中是否存在该文件；如果不存在，

则打开后再读；然后检查是否是以读方式打开文件，如果是以写方式打开文件，则不允许读；最后从已打开文件表中读出读指针，从这个位置上读出所需要长度，若所需长度没有读完已经遇到文件结束符，就终止操作。实验中用“#”表示文件结束。

④写文件（write_file）

用户要求存取文件信息时调用文件系统的“写文件”操作。实验中，写文件的参数只需要文件名、存放准备写入磁盘信息的缓冲和写的长度，write_file（文件名，缓冲，写长度）。因为采用流式文件结构，所以写长度用字节表示。

实验中，写文件操作的主要工作是查找已打开文件表中是否存在该文件，如果不存在，则打开后再写；如果存在，还要检查是否以写方式打开文件；如果不是写方式打开文件，不能写；最后从已打开文件表中读出写指针，从这个位置上写入缓冲中的数据。

写文件有两种情况，一种情况是建立文件后的写入，这种写比较简单，一边写一边申请空间即可完成；另一种情况是文件打开后的写入，这个比较复杂，存在着文件中间修改的问题。实验中，第二种情况只要求完成从文件末尾向后追加的功能。

⑤关闭文件（close_file）

用户对文件读写完毕后需要调用文件系统的“关闭文件”操作。

实验中，关闭文件的参数只需要文件名，close_file（文件名）。

实验中关闭文件，首先要看该文件是否打开，如果没有打开，就不用关闭；如果已经打开，则检查打开方式，如果是写方式打开的，要追加文件结束符，修改目录项；最后从已打开文件表中删除对应项。

⑥删除文件（delete_file）

用户认为文件没有必要保存时需要调用文件系统的“删除文件”操作。实验中，删除文件时参数只要文件名，delete_file（文件名）。

实验中，删除文件操作的主要工作是检查文件是否存在；不存在，操作失败；如存在，查找该文件是否打开，如果打开不能删除；如果没有打开，则删除文件目录项并归还文件所占磁盘空间。

⑦显示文件内容（typefile）

显示文件内容首先要找到该文件的目录登记项，如果文件不存在，指令执行失败；如果存在，查看文件是否打开，打开则不能显示文件内容；若没有打开，从目录中取出文件的起始盘块号，一块一块显示文件内容。

⑧改变文件属性（change）

改变文件属性，首先查找该文件，如果不存在，结束；如果存在，检查文件是否打开，打开不能改变属性；没有打开，根据要求改变目录项中属性值。

实验中，首先要系统初始化，包括建立文件模拟磁盘、初始化磁盘、初始化根目录为空目录项；然后，可以选择一项功能执行。

目录的操作命令：

①建立目录（md）

建立目录首先要找到建立目录的位置（父目录），然后查找该目录是否存在，如果父目录不存在，不能建立；如果存在，查找是否存在同名目录，存在，不能建立；不存在，则查找一个空目录项，为该目录申请一个盘块，并填写目录内容。

②显示目录内容（dir）

显示目录内容首先要找到该目录，如果目录不存在，指令执行失败；如果存在，一项一项显示目录内容。

③删除空目录（rd）

删除空目录首先要找到该目录，如果目录不存在，指令执行失败；如果存在，但是根目录或非空目录，显示不能删除，操作失败；若是非空子目录，则删除其目录项并回收对应空间。删除空目录的过程和删除文件的过程相似。

另外注意，对磁盘文件进行读操作时，需要磁盘的一个盘块读入主存后才能进行处理，对磁盘文件进行写操作时，要写满缓冲后才写入磁盘。所以模拟文件操作时，不能将整个模拟磁盘的内容同时读入主存，应该当需要模拟磁盘的某个盘块内容时，从对应文件中读出；修改后需要写回模拟磁盘。实验中就是用这种方法模拟磁盘的输入输出。

实验中需要定义两个数组 buffer1 和 buffer2 模拟缓冲。

实验中首先系统初始化，包括建立文件模拟磁盘，初始化磁盘和根目录初始化为空目录项，然后选择各个命令进行测试。

有能力的同学可在上述基础上，将磁盘文件系统改进为支持多级树型目录，支持相对路径，子目录可以任意长的文件系统。

题目六　模拟操作系统实现

一、课程设计目的

通过模拟操作系统的实现，加深对操作系统工作原理理解，进一步了解操作系统的实现方法，并可练习合作完成系统的团队精神和提高程序设计能力。

二、小组人数

建议 4～6 人一组共同完成模拟系统的实现。

选择题目“模拟操作系统实现”的小组在最终提交时须公开演示及讲解。由于这个题目最复杂，难度和工作量最大，故小组成员最后得分也酌情高于选择其它题目的同学的分数（高 5~15 分）。

三、编程语言

建议使用一些 Windows 环境下的程序设计语言如 VC、Java，以借助这些语言的多线程来模拟并行发生的行为。要求图形界面。

四、课程设计内容

模拟一个采用多道程序设计方法的单用户操作系统，该操作系统包括进程管理、存储管理、设备管理、文件管理和用户接口四部分。

五、课程设计具体内容和要求

1、文件管理和用户接口

文件管理和用户接口部分实现的主要是单用户的磁盘文件管理部分，包括文件的逻辑结构、物理结构、目录、磁盘分配回收、文件的保护和用户接口的实现。这一部分可参考题目五的说明。

（1） 文件的逻辑结构

文件的逻辑结构采用流式结构；文件均采用文本文件。

假设系统中只有两种文件，一种是存放任意字符的普通文本文件，一种是可执行文件。**可执行文件的程序内容手工输入，事先创建约 10 个可执行文件，将来用这些可执行文件进行后续的进程创建、内存分配、进程执行/调度和设备分配。**

这里，“可执行文件”中的“指令”只有 5 种，包括：

x=?　给 x 赋值（数值不用太大，一位数、两位数即可。不要求存储或任意指定变量名，只是实现“赋值”即可）。

x++　x 加 1 （设 x 值总是小于等于 255、大于等于 0）。

x--　　x 减 1。

!? ?　　!是“特殊命令（I/O）的前缀”，第一个?为 A,B,C 中的某个设备，第二个?为一位整数，表示使用设备的时间（例如假定一个数，这个数随着系统时间增加而递减（时间单位自定，例如：秒）。减到 0 时，认为设备工作完成）。

end　表示“可执行文件”结束。

每个可执行文件中可以包含多条同一类指令，假设每条指令在文件中占 1 字节（自己思考如何把前述 5 种指令用一个字节表示并存储在后述的 disk 磁盘块里，其实这是一个“汇编/编译”的模拟过程）。如果可执行文件中的指令超过 64 条，就再多分配一个磁盘块，以此类推。

假设每种指令都是原子性的（即不考虑各指令的汇编实现细节，每种指令都是原子执行的），且执行时间都是一个时间单位，后面进程调度的时间片是 n 个时间单位，例如 n=6，就表示本进程执行 6 条指令之后将发生进程调度。

（2） 磁盘模拟

用一个文件 disk 模拟磁盘，设磁盘的每个盘块 64 字节，模拟磁盘共有 256 块。第 0、1 块存放文件分配表，第 2 块存放根目录，其余存放子目录和文件。（所以你创建的目录和流式文件不能太大太多，但至少要包含 5 个目录和 15 个文件。注意：文件对磁盘块是独占的，即每个文件至少占据一个磁盘块，不会让两个文件共栖于同一磁盘块。）

（3） 目录结构

目录结构采用树型目录结构。

(a) 目录项内容：

每个目录项 8 个字节，其中：

目录名或文件名：3 个字节；

扩展名：1 个字节（可执行文件扩展名为 e，目录没有扩展名）；

目录属性、文件属性：1 字节；

起始盘号：1 字节；

文件长度：2 字节。

(b) 根目录

根目录位置固定，为磁盘第 2 块，大小固定，共可包含 8 个目录项，占用模拟磁盘第 2 块；

(c) 子目录

位置不固定，大小不固定。

（4） 磁盘分配

磁盘的分配采用链接结构（显式链接）的分配方式。系统采用文件分配表方式记录磁盘空间的使用情况和链接结构的指针。（参考 MSDOS 的 FAT，题目五中也有 FAT 的提示说明）

文件分配表中一项需要 1 字节，而磁盘有 256 块，因而有 256 项，模拟磁盘空间中的第 0、1 块用来存放文件分配表。

（5） 用户接口

用户接口提供用户命令接口，接收用户从键盘键入的命令。如果不使用键盘输入命令的方式，那么就模拟 windows 操作方式，采用“右击快捷菜单”方式提供“创建删除文件/目录，修改属性”等命令，拖动来移动/复制文件等。

要求实现以下命令：

(下面例子中的 $ 只是命令提示符而已，和 UNIX 无关。除 aa 是目录名外，其余均为文件名。你可以根据自己实现方便或喜欢而自定义命令参数。)

需要实现的命令包括：

创建文件：create　　例如 $ create \aa\bb.e

删除文件：delete　　例如 $ delete \aa\yy

显示文件：type　　例如 $ type \zz

拷贝文件：copy　　例如 $ copy \xx \aa\yy

建立目录：mkdir　　例如 $ mkdir \dd

删除空目录：rmdir　　目录非空时，要报错。

可选实现的命令包括：

改变目录路径：chdir

删除目录：deldir（既可删除空目录又可删除非空目录）

移动文件：move

改变文件属性：change

磁盘格式化：format

磁盘分区命令：fdisk

上述命令在实际系统中都是需要建立进程才可以运行的，这里为简单起见，这些命令执行时不必在模拟系统中建立进程，可直接让 Windows 执行你编写的相应函数。

（6） 屏幕显示

如图 1，屏幕显示要求包括：

用户命令接口:用于系统运行时用户用键盘输入或模拟 win 命令；

磁盘目录显示：要求显示磁盘的目录结构；

磁盘使用情况：显示磁盘每一个磁盘块的空间是占用还是空闲。

2、存储管理

存储管理部分主要实现内存空间的分配和回收、存储保护。

用链表模拟内存空间分配表。存储管理采用动态分区存储管理方式，采用首次适配、下次适配、最佳适配均可。

（1）模拟系统中，主存部分分为两部分，一部分是系统区，这里只存放进程控制块和内存分配表，一部分是用户区，存放可执行文件。

系统区包括 PCB 区域（最多容纳 10 个 PCB）、内存空间分配表；（长度自己定）

用户区用数组模拟，大小为 512 字节。例如要执行一个 36 字节大的 bb.e，则分配 36 字节给这个 bb 进程。如果 cc.e 需要 110 字节，

太大而无法分配，则 cc.e 无法载入“内存”而等待。

（2）屏幕显示

如图 1，屏幕显示要求包括：内存使用情况示意图，以不同的颜色表示哪些区域未分配或已分配（已分配给哪个进程）。

3、设备管理

设备管理主要包括设备的分配和回收。（可设一张“设备分配表”和设备等待队列）

（1）模拟系统中有 A、B、C 三种独占型设备，A 设备 2 个，B 设备 3 个，C 设备 3 个。（同一种设备的不同实体被认为是相同的）

（2）设备的申请是由于前述可执行文件中的!? ?指令引起，有空闲设备时分配设备，然后进程阻塞，同时设备使用倒计时至 0 后释放设备（不考虑设备具体的 I/O 操作）并唤醒进程继续运行；无空闲设备时阻塞进程，直至其它进程释放设备时才分配设备并使用，设备使用完后唤醒进程。

注意：设备使用倒计时期间，本进程阻塞，需要调度另外一个进程去占用 CPU 执行；假设设备使用完后立即释放该设备，后续指令需再次使用该设备时重新分配。

（3）不考虑死锁。

（4）屏幕显示如图 1 所示，屏幕显示要求包括：每个设备是否被使用，哪个进程在使用该设备，哪些进程在等待使用该设备。

4、进程管理

进程管理主要包括进程调度，进程的创建和撤销、进程的阻塞和唤醒，中断作用的实现。

（1）硬件工作的模拟

（a） 中央处理器的模拟

用函数 CPU()（该函数没有参数）模拟单个中央处理器。

该函数主要负责解释“可执行文件”中的指令。（为简单计，用

户命令接口部分实现的命令不必用 CPU()解释执行）

设一个“程序计数器”，跟踪现在执行到哪条指令。

如果是赋值或加减指令，那么就赋值或加减，并在屏幕上显示中间结果；

如果是!?设备申请指令，那么要和设备管理联系起来，看设备是否可分配，然后采取不同动作。

如果是 end 指令，则释放内存，结束本进程。

考虑到 CPU()函数执行、系统时钟递增、相对时钟（时间片）递减、设备倒计时递减、随机选择新进程诞生等多种任务的同时存在，可能需要多线程编程，才能保证这几种任务的并发。

（b） 主要寄存器的模拟

用全局变量或数组模拟重要寄存器，如数据寄存器（这里可以只设一个 AX，放 x 的值），程序状态寄存器 PSW，指令寄存器 IR，程序计数器 PC 等。

（c） 中断的模拟

I、 中断的发现应该是硬件的工作，但在这里，用在函数 CPU()中检测 PSW 的方式来模拟。

在 CPU()函数中，每执行一条指令之前，先检查 PSW，判断有无中断，若有则先进行中断处理，然后再解释运行指令。

CPU 函数应该不断循环执行。

II、 模拟中断的种类和中断处理方式

发生下述 3 种模拟中断时，分别将 PSW 中的 3 个 bit 设置为 1，处理完中断后将相应 bit 设置为 0。

① 程序结束（执行指令 end 形成的软中断）：在屏幕上输出 x 的值，调用“进程撤销原语”撤销进程，然后进行进程调度；

② 时间片结束（当相对时钟减到 0 时）：将正在运行进程的 CPU 现场（寄存器值即可，暂不考虑“栈”）保存在进程

控制块中，然后进行进程调度；

③ I/O 中断发生（设备使用时间变量倒计时至 0，完成输入输出）：将输入输出完成的进程唤醒，同时将等待该设备的另一个进程唤醒。

（d） 时钟的模拟

系统时钟和相对时钟用全局变量模拟。系统时钟用来记录开机以后的单位时间，相对时钟用来存放进程可运行的时间片（如 6 个单位时间），在进程调度时设置初值，随系统时间的增值 1 而减值 1，减到 0 时，发出时钟中断。

这里的系统时钟并不是计算机的真正时钟，而是单位时间，时间单位长度自定（例如：秒，0.5 秒等），可以使用程序语言里面的一些有关时间的函数来实现。

（2）进程控制块 PCB

进程控制块内容包括进程标识符、主要寄存器、进程状态、阻塞原因等。本模拟系统最多容纳 10 个进程块。PCB 区域用数组模拟。

进程控制块根据内容的不同组成三个队列：空白 PCB 队列，就绪队列和阻塞队列。正在运行的进程只有一个，系统初始时只有空白 PCB 队列。

（3）进程调度

(a) 首先随机选择前面创建的 10 个可执行文件之一，创建进程 PCB，分配内存，然后逐条执行其中的指令；然后经过随机时间后，再选择一个可执行文件，创建进程……，如此往复，模拟操作系统中进程随机到达的过程。

采用时间片轮转调度算法，时间片长度为 6。

(b) 进程调度函数的主要工作是：

将正在运行进程的现场（寄存器组）保存在该进程的 PCB 中；（保存现场）

从就绪队列中选择一个进程；

将这个进程的 PCB 中记录的各寄存器内容恢复到 CPU 各个寄存器内（恢复现场），根据程序计数器 PC 继续执行后续指令。

(c) 闲逛进程

建立一个闲逛进程，当就绪队列为空时，系统调用该进程运行。当有进程就绪时，就调用就绪进程运行。闲逛进程什么有用的事也不做，只是起到系统能正常运转的作用。

（4）进程控制

建立 4 个函数模拟进程创建、撤销、阻塞和唤醒四个原语

(a) 进程创建 create()，主要工作是：

第一步，申请空白进程控制块；

第二步，申请主存空间，申请成功，装入主存；

第三步，初始化进程控制块；

第四步，在屏幕上显示进程执行结果，进程撤销。

(b) 进程撤销 destroy()，主要工作是：

第一步，回收进程所占内存；

第二步，回收进程控制块；

第三步，在屏幕上显示进程执行结果，进程撤销。

(c) 进程阻塞 block()，主要工作是：

第一步，保存运行进程的 CPU 现场；

第二步，修改进程状态；

第三步，将进程链入对应的阻塞队列，然后转向进程调度。

(d) 进程唤醒 awake()

进程唤醒的主要工作是将进程由阻塞队列中摘下，修改进程状态为就绪，然后链入就绪队列。

（5）屏幕显示

如图 1，屏幕显示要求包括：

显示系统时钟；

显示正在运行进程的进程 ID、运行的指令、中间结果（x 的当前

值）、相对时钟（时间片剩余值）；

显示就绪队列中进程 ID；

显示阻塞队列中进程 ID。

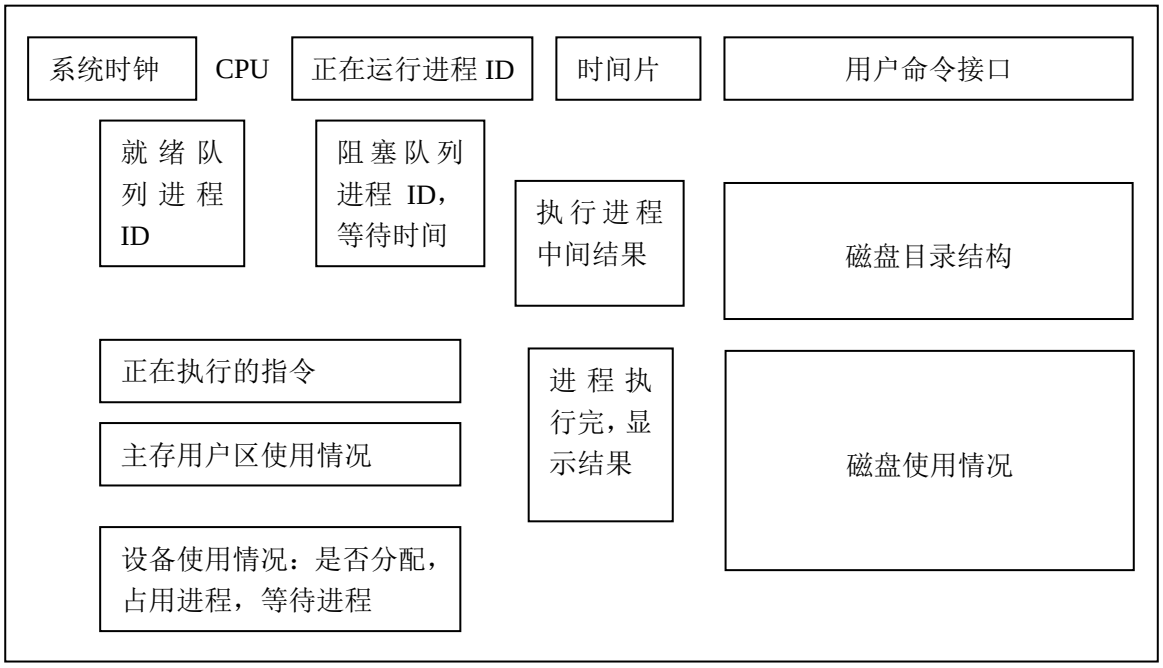


图 1 模拟操作系统的屏幕显示布局和内容